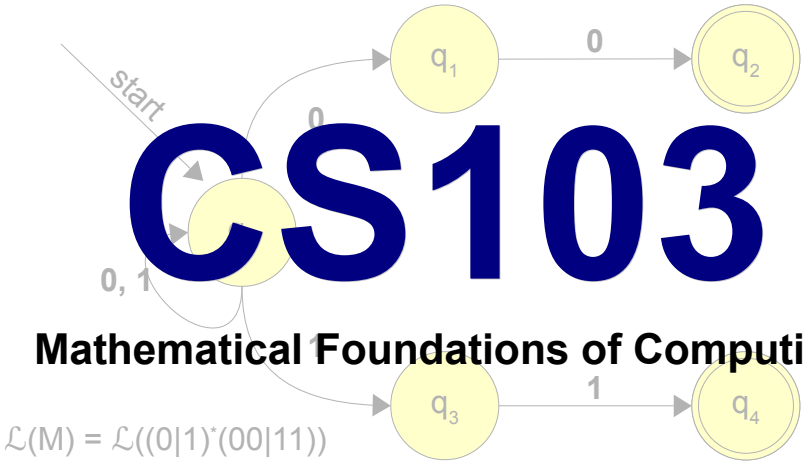



# CS103

## Mathematical Foundations of Computing

**Preliminary Course Notes**

**Keith Schwarz**

**Fall 2014**

# Table of Contents

# Chapter 0 Introduction

Computer science is the art of solving problems with computers. This is a broad definition that encompasses an equally broad field. Within computer science, we find software engineering, bioinformatics, cryptography, machine learning, human-computer interaction, graphics, and a host of other fields.

Mathematics underpins all of these endeavors in computer science. We use graphs to model complex problems, and exploit their mathematical properties to solve them. We use recursion to break down seemingly insurmountable problems into smaller and more manageable problems. We use topology, linear algebra, and geometry in 3D graphics. We study computation itself in computability and complexity theory, with results that have profound impacts on compiler design, machine learning, cryptography, computer graphics, and data processing.

This set of course notes serves as a first course in the mathematical foundations of computing. It will teach you how to model problems mathematically, reason about them abstractly, and then apply a battery of techniques to explore their properties. It will teach you to prove mathematical truths beyond a shadow of a doubt. It will give you insight into the fundamental nature of computation and what can and cannot be solved by computers. It will introduce you to complexity theory and some of the most important problems in all computer science.

This set of course notes is intended to give a broad and deep introduction to the mathematics that lie at the heart of computer science. It begins with a survey of discrete mathematics – basic set theory and proof techniques, mathematic induction, graphs, relations, functions, and logic – then explores computability and complexity theory. No prior mathematical background is necessary.

These course notes are designed to provide a secondary treatment of the material from Stanford's CS103 course. The topic organization roughly follows the presentation from CS103, albeit at a much deeper level. It is designed to supplement lectures from CS103 with additional exposition on each topic, broader examples, and more advanced applications and techniques. My hope is that there will be something for everyone. If you're interested in brushing up on definitions and seeing a few examples of the various techniques, feel free to read over the relevant parts of each chapter. If you want to see the techniques applied to a variety of problems, look over the examples that catch your interest. If you want to see just how deep the rabbit hole goes, read over the entire chapter and work through all the starred exercises.

## 0.1 How These Notes are Organized

The organization of this book into chapters is as follows:

- Chapter One motivates our discussion of mathematics by giving a brief outline of set theory, concluding with *Cantor's theorem*, a profound and powerful result about the nature of infinity.
- Chapter Two lays the groundwork for formal proofs by exploring several key proof techniques that we will use throughout the rest of these notes.
- Chapter Three explores mathematical induction, a proof technique that we will use extensively when proving results about discrete structures, programs, and computation.
- Chapter Four investigates mathematical structures called *graphs* that are useful for modeling complex problems. These structures will form the basis for many of the models of computation we will investigate later.

- Chapter Five explores different ways in which objects often relate to one another and the mathematical structures of these relations.
- Chapter Six revisits the material on set theory from the first chapter with a mathematically rigorous exploration of the properties of infinity.
- Chapter Seven introduces a proof technique called the *pigeonhole principle* that superficially appears quite simple, but which has profound implications for the limits of computation.
- Chapter Eight explores formal logic and gives a more rigorous basis for the proof techniques developed in the preceding chapters.
- Chapter Nine introduces key concepts from *computability theory*, the study of what problems can and cannot be solved by a computer.
- Chapter Ten introduces *finite automata*, *regular expressions*, and the *regular languages*. It serves as an introduction to the study of formal models of computations. These models have applications throughout computer science
- Chapter Eleven introduces *pushdown automata* and *context-free grammars*, more powerful formalisms for describing computation.
- Chapter Twelve introduces the *Turing machine*, an even more powerful model of computation that we suspect is the most powerful feasible computing machine that could ever be built.
- Chapter Thirteen explores the power of the Turing machine by exploring what problems it can solve.
- Chapter Fourteen provides examples of problems that are provably beyond our capability to solve with any type of computer.
- Chapter Fifteen introduces *reductions*, a fundamental technique in computability and complexity theory for reasoning about the relative difficulties of problems.
- Chapter Sixteen introduces *complexity theory*, the study of what problems can be computed given various resource constraints.
- Chapter Seventeen introduces the complexity classes **P** and **NP**, which contain many problems of great practical and theoretical importance.
- Chapter Eighteen explores the connection between the classes **P** and **NP**, which (as of 2012) is the single biggest open problem in all of theoretical computer science.
- Chapter Nineteen goes beyond **P** and **NP** and explores alternative models of computation and their complexities.
- Chapter Twenty concludes our treatment of the mathematical foundations of computing and sets the stage for further study on the subject.

Some of the sections of these notes are marked with a ★ symbol. These sections contain more advanced material that, while interesting, is not essential to understanding the course material. If you are interested in learning more about the topic, I would suggest reading through it. However, if you're pressed for time, feel free to skip these sections.

Similarly, some of the chapter exercises are marked with the ★ symbol. These exercises are more difficult than the other exercises. I highly recommend attempting at least one or two of the starred exercises from each chapter. If you're up for a real challenge, try working through all of them!

This is a work-in-progress draft of what I hope will become a full set of course notes for CS103. Right now, the notes only cover up through the first six or seven lectures. I am hoping to expand that over the course of the upcoming months. Since this is a first draft, there are *definitely* going to be errors in here, whether they're typoz, grammatically problems, layout issues, and logic errors. If you find any errors, please don't hesitate to get in touch with me at **htiek@cs.stanford.edu** – I genuinely want these notes to be as good as they can be, so any feedback would be most appreciated.

# 0.2 Acknowledgements

These course notes represent the product of several years of hard work. I received invaluable feedback on the content and structure from many of my friends, students, and colleagues. In particular, I'd like to thank Leonid Shamis, Sophia Westwood, and Amy Nguyen for their comments. Their feedback has dramatically improved the quality of these notes, and I'm very grateful for the time and effort they put into reviewing drafts at every stage.

I'd also like to thank everyone who found typos or other errors in this set of course notes. Thanks to Rob Patrick, Emily Tang, Vignesh Venkataraman, Akshay Agrawal, Stephanie Tsai, Clara Fannjiang, Andrew Dahlstrom, William Lewis, Arzav Jain, Luke Pappas, Mark Cuson, Greg Gibson, Zak Butler, Katherine Kuang, Anshul Samar, Yitao Zhang, Karen Gomez, Emily Alsentzer, Sophie Ye, Jessica Xu, Linda Thompson, AJ Hollopeter, Joel Aguero, Gary Beene, Angela Ellington, Mitchell Douglass, Lloyd Lucin, Lidan Cao, Doug Vargha, and Craig Smail. If you submitted a correction in the past and your name doesn't appear on this list, please let me know; I want to give credit where credit is due!

# Chapter 1 Sets and Cantor's Theorem

Our journey into the realm of mathematics begins with an exploration of a surprisingly nuanced concept: the *set*. Informally, a set is just a collection of things, whether it's a set of numbers, a set of clothes, a set of nodes in a network, or a set of other sets. Amazingly, given this very simple starting point, it is possible to prove a result known as *Cantor's Theorem* that provides a striking and profound limit on what problems a computer program can solve. In this introductory chapter, we'll build up some basic mathematical machinery around sets. Then, we will see how the simple notion of asking how big a set is can lead to incredible and shocking results.

## 1.1 What is a Set?

Let's begin with a simple definition:

> A ***set*** is an unordered collection of distinct elements.

What exactly does this mean? Intuitively, you can think of a set as a group of things. Those things must be distinct, which means that you can't have multiple copies of the same object in a set. Additionally, those things are unordered, so there's no notion of a "first" thing in the group, a "second" thing in a group, etc.

We need two additional pieces of terminology to talk about sets:

> An ***element*** is something contained within a set.

To denote a set, we write the elements of the set within curly braces. For example, the set {1, 2, 3} is a set with three elements: 1, 2, and 3. The set { cat, dog } is a set containing the two elements "cat" and "dog."

Because sets are *unordered* collections, the order in which we list the elements of a set doesn't matter. This means that the sets {1, 2, 3}, {2, 1, 3}, and {3, 2, 1} are all descriptions of exactly the same set. Also, because sets are unordered collections of *distinct elements*, no element can appear more than once in the same set. In particular, this means that if we write out a set like {1, 1, 1, 1, 1}, it's completely equivalent to writing out the set {1}, since sets can't contain duplicates. Similarly, {1, 2, 2, 2, 3} and {3, 2, 1} are the same set, since ordering doesn't matter and duplicates are ignored.

When working with sets, we are often interested in determining whether or not some object is an element of a set. We use the notation $x \in S$ to denote that $x$ is an element of the set $S$. For example, we would write that $1 \in \{1, 2, 3\}$, or that cat $\in$ { cat, dog }. If spoken aloud, you'd read $x \in S$ as "$x$ is an element of $S$." Similarly, we use the notation $x \notin S$ to denote that $x$ is not an element of $S$. So, we would have $1 \notin \{2, 3, 4\}$, dog $\notin \{1, 2, 3\}$, and ibex $\notin$ {cat, dog}. You can read $x \notin S$ as "$x$ is not an element of $S$."

Sets appear almost everywhere in mathematics because they capture the very simple notion of a group of things. If you'll notice, there aren't any requirements about what can be in a set. You can have sets of integers, sets of real numbers, sets of lines, sets of programs, and even sets of other sets. Through the remainder of your mathematical career, you'll see sets used as building blocks for larger and more complicated objects.

As we just mentioned, it's possible to have sets that contain other sets. For example, the set $\{ \{1, 2\}, 3 \}$ is a set containing two elements – the set $\{1, 2\}$ and the number 3. There's no requirement that all of the elements of a set have the same "type," so a single set could contain numbers, animals, colors, and other sets without worry. That said, when working with sets that contain other sets, it's important to note what the elements of that set are. For example, consider the set

$$\{\{1, 2\}, \{2, 3\}, 4\}$$

which has just three elements: $\{1, 2\}$, $\{2, 3\}$, and 4. This means that

$$\{1, 2\} \in \{\{1, 2\}, \{2, 3\}, 4\}$$

$$\{2, 3\} \in \{\{1, 2\}, \{2, 3\}, 4\}$$

$$4 \in \{\{1, 2\}, \{2, 3\}, 4\}$$

However, it is **not** true that $1 \in \{\{1, 2\}, \{2, 3\}, 4\}$. Although $\{\{1, 2\}, \{2, 3\}, 4\}$ contains the set $\{1, 2\}$ which in turn contains 1, 1 itself is not an element of $\{\{1, 2\}, \{2, 3\}, 4\}$. In a sense, set containment is "opaque," in that it just asks whether the given object is directly contained within the set, not whether it is contained with that set, a set contained within that set, etc. Consequently, we have that

$$1 \notin \{\{1, 2\}, \{2, 3\}, 4\}$$

$$2 \notin \{\{1, 2\}, \{2, 3\}, 4\}$$

$$3 \notin \{\{1, 2\}, \{2, 3\}, 4\}$$

But we do have that

$$4 \in \{\{1, 2\}, \{2, 3\}, 4\}$$

because 4 is explicitly listed as a member of the set.

In the above example, it's fairly time-consuming to keep writing out the set $\{\{1, 2\}, \{2, 3\}, 4\}$ over and over again. Commonly, we'll assign names to mathematical objects to make it easier to refer to them in the future. In our case, let's call this set "S." Mathematically, we can write this out as

$$S = \{\{1, 2\}, \{2, 3\}, 4\}$$

Given this definition, we can rewrite all of the above discussion much more compactly:

$$\{1, 2\} \in S \qquad 1 \notin S$$

$$\{2, 3\} \in S \qquad 2 \notin S$$

$$4 \in S \qquad 3 \notin S$$

Throughout this book, and in the mathematical world at large, we'll be giving names to things and then manipulating and referencing those objects through those names. Hopefully you've used variables before when programming, so I hope that this doesn't come as too much of a surprise.

Before we move on and begin talking about what sorts of operations we can perform on sets, we need to introduce a very special set that we'll be making extensive use of: the **empty set**.

> The ***empty set*** is the set that does not contain any elements.

It may seem a bit strange to think about a collection of no things, but that's precisely what the empty set is. You can think of the empty set as representing a group that doesn't have anything in it. One way that we

could write the empty set is as { }, indicating that it's a set (the curly braces), but that this set doesn't contain anything (the fact that there's nothing in-between them!) However, in practice this notation is not used, and we use the special symbol Ø to denote the empty set.

The empty set has the nice property that there's nothing in it, which means that for any object x that you ever find anywhere, $x \notin \emptyset$. This means that the statement $x \in \emptyset$ is always false.

Let's return to our earlier discussion of sets containing other sets. It's possible to build sets that contain the empty set. For example, the set { Ø } is a set with one element in it, which is the empty set. Thus we have that $\emptyset \in \{ \emptyset \}$. More importantly, note that Ø and { Ø } are **not** the same set. Ø is a set that contains no elements, while { Ø } is a set that does indeed contain an element, namely the empty set. Be sure that you understand this distinction!

# 1.2 Operations on Sets

Sets represent collections of things, and it's common to take multiple collections and ask questions of them. What do the collections have in common? What do they have collectively? What does one set have that the other does not? These questions are so important that mathematicians have rigorously defined them and given them fancy mathematical names.

First, let's think about finding the elements in common between two sets. Suppose that I have two sets, one of US coins and one of chemical elements. This first set, which we'll call *C*, is

$$C = \{ \text{penny, nickel, dime, quarter, half-dollar, dollar} \}$$

and the second set, which we'll call *E*, contains these elements:

$$E = \{ \text{hydrogen, helium, lithium, beryllium, boron, carbon, …, ununseptium} \}$$

(Note the use of the ellipsis here. It's often acceptable in mathematics to use ellipses when there's a clear pattern present, as in the above case where we're listing the elements in order. Usually, though, we'll invent some new symbols we can use to more precisely describe what we mean.)

The sets *C* and *E* happen to have one element in common: nickel, since

$$\text{nickel} \in C$$

and

$$\text{nickel} \in E$$

However, some sets may have a larger overlap. For example, consider the sets {1, 2, 3} and {1, 3, 5}. These sets have both 1 and 3 in common. Other sets, on the other hand, might not have anything in common at all. For example, the sets {cat, dog, ibex} and {1, 2, 3} have no elements in common.

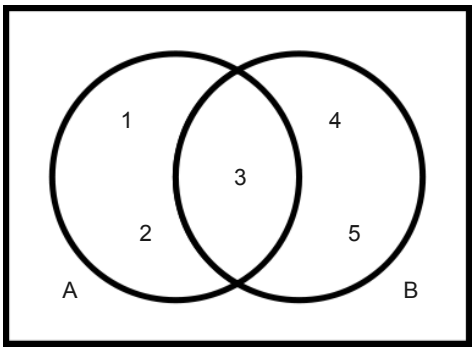
Since sets serve to capture the notion of a collection of things, we might think about the set of elements that two sets have in common. In fact, that's a perfectly reasonable thing to do, and it goes by the name **intersection**:

> The ***intersection*** of two sets *S* and *T*, denoted $S \cap T$, is the set of elements contained in both *S* and *T*.

For example, $\{1, 2, 3\} \cap \{1, 3, 5\} = \{1, 3\}$, since the two sets have exactly 1 and 3 in common. Using the set *C* of currency and *E* of chemical elements from earlier, we would say that $C \cap E = \{\text{nickel}\}$.

But what about the intersection of two sets that have nothing in common? This isn't anything to worry about. Let's take an example: what is {cat, dog, ibex} ∩ {1, 2, 3}? If we consider the set of elements common to both sets, we get the empty set Ø, since there aren't any common elements between the two sets.

Graphically, we can visualize the intersection of two sets by using a *Venn diagram*, a pictorial representation of two sets and how they overlap. You have probably encountered Venn diagrams before in popular media. These diagrams represent two sets as overlapping circles, with the elements common to both sets represented in the overlap. For example, if $A = \{1, 2, 3\}$ and $B = \{3, 4, 5\}$, then we would visualize the sets as




Given a Venn diagram like this, the intersection $A \cap B$ is the set of elements in the intersection, which in this case is {3}.

Just as we may be curious about the elements two sets have in common, we may also want to ask what elements two sets contain collectively. For example, given the sets $A$ and $B$ from above, we can see that, collectively, the two sets contain 1, 2, 3, 4, and 5. Mathematically, the set of elements held collectively by two sets is called the **union**:

> The ***union*** of two sets $A$ and $B$, denoted $A \cup B$, is the set of all elements contained in either of the two sets.

Thus we would have that {**1**, **2**, **3**} ∪ {**3**, **4**, **5**} = {**1**, **2**, **3**, **4**, **5**} (here, I'm color-coding the numbers to indicate which sets they're from; we'll treat all numbers as the same regardless of their color). Since sets are unordered collections of *distinct* elements, that it would also have been correct to write {**1**, **2**, **3**, **3**, **4**, **5**} as the union of the two sets, since {**1**, **2**, **3**, **4**, **5**} and {**1**, **2**, **3**, **3**, **4**, **5**} describe the same set. That said, to eliminate redundancy, typically we'd prefer to write out {1, 2, 3, 4, 5} since it gets the same message across in less space.

The symbols for union (∪) and intersection (∩) are similar to one another, and it's often easy to get them confused. A useful mnemonic is that the symbol for union looks like a U, so you can think of the ∪nion of two sets.

An important (but slightly pedantic) point is that ∪ can only be applied to two sets. This means that although

$$\{1, 2, 3\} \cup 4$$

might intuitively be the set {1, 2, 3, 4}, mathematically this statement isn't meaningful because 4 isn't a set. If we wanted to represent the set formed by taking the set {1, 2, 3} and adding 4 to it, we would represent it by writing

$$\{1, 2, 3\} \cup \{4\}$$

Now, since both of the operands are sets, the above expression is perfectly well-defined. Similarly, it's not mathematically well-defined to say

$$\{1, 2, 3\} \cap 3$$

because 3 is not a set. Instead, we should write

$$\{1, 2, 3\} \cap \{3\}$$

Another important point to clarify when working with union or intersection is how they behave when applied to sets containing other sets. For example, what is the value of the following expression?

$$\{\{1, 2\}, \{3\}, \{4\}\} \cap \{\{1, 2, 3\}, \{4\}\}$$

When computing the intersection of two sets, all that we care about is what elements the two sets have in common. Whether those elements themselves are sets or not is irrelevant. Here, for example, we can list off the elements of the two sets $\{\{1, 2\}, \{3\}, \{4\}\}$ and $\{\{1, 2, 3\}, \{4\}\}$ as follows:

$$\{1, 2\} \in \{\{1, 2\}, \{3\}, \{4\}\} \qquad \{1, 2, 3\} \in \{\{1, 2, 3\}, \{4\}\}$$

$$\{3\} \in \{\{1, 2\}, \{3\}, \{4\}\} \qquad \{4\} \in \{\{1, 2, 3\}, \{4\}\}$$

$$\{4\} \in \{\{1, 2\}, \{3\}, \{4\}\}$$

Looking at these two lists of elements, we can see that the only element that the two sets have in common is the element $\{4\}$. As a result, we have that

$$\{\{1, 2\}, \{3\}, \{4\}\} \cap \{\{1, 2, 3\}, \{4\}\} = \{\{4\}\}$$

That is, the set of just one element, which is the set containing 4.

You can think about computing the intersection of two sets as the act of peeling off just the outermost braces from the two sets, leaving all the elements undisturbed. Then, looking at just those elements, find the ones in common to both sets, and gather all of those elements back together.

The union of two sets works similarly. So, if we want to compute

$$\{\{1, 2\}, \{3\}, \{4\}\} \cup \{\{1, 2, 3\}, \{4\}\}$$

We would "peel off" the outer braces to find that the first set contains $\{1, 2\}$, $\{3\}$, and $\{4\}$ and that the second set contains $\{1, 2, 3\}$ and $\{4\}$. If we then gather all of these together into a set, we get the result that

$$\{\{1, 2\}, \{3\}, \{4\}\} \cup \{\{1, 2, 3\}, \{4\}\} = \{\{1, 2\}, \{3\}, \{1, 2, 3\}, \{4\}\}$$

Given two sets, we can find what they have in common by finding their intersection and can find what they have collectively by using the union. But both of these operations are *symmetric*; it doesn't really matter what order the sets are in, since $A \cup B = B \cup A$ and $A \cap B = B \cap A$. (If this doesn't seem obvious, try out a couple of examples and see if you notice anything). In a sense, the union and intersection of two sets don't have a "privileged" set. However, at times we might be interested in learning about how one set differs from another. Suppose that we have two sets $A$ and $B$ and want to find the elements of $A$ that don't appear in $B$. For example, given the sets $A = \{1, 2, 3\}$ and $B = \{3, 4, 5\}$, we would note that the elements 1 and 2 are unique to $A$ and don't appear anywhere in $B$, and that 4 and 5 are unique to $B$ and don't appear in $A$. We can capture this notion precisely with the *set difference* operation:

> The ***set difference*** of $A$ and $B$, denoted $A - B$ or $A \setminus B$, is the set of elements contained in $A$ but not contained in $B$.

Note that there are two different notations for set difference. In this book we'll use the minus sign to indicate set subtraction, but other authors use the slash for this purpose. You should be comfortable working with both.

As an example of a set difference, $\{1, 2, 3\} - \{3, 4, 5\} = \{1, 2\}$, because 1 and 2 are in $\{1, 2, 3\}$ but not $\{3, 4, 5\}$. Note, however, that $\{3, 4, 5\} - \{1, 2, 3\} = \{4, 5\}$, because 4 and 5 are in $\{3, 4, 5\}$ but not $\{1, 2, 3\}$. Set difference is not symmetric. It's also possible for the difference of two sets to contain nothing at all, which would happen if everything in the first set is also contained in the second set. For instance, we have $\{1, 2, 3\} - \{1, 2, 3, 4\} = \varnothing$, since every element of $\{1, 2, 3\}$ is also contained in $\{1, 2, 3, 4\}$.

There is one final set operation that we will touch on for now. Suppose that you and I travel the world and each maintain a set of the places that we went. If we meet up to talk about our trip, we'd probably be most interested to tell each other about the places that one of us had gone but the other hadn't. Let's say that set $A$ is the set of places I have been and set $B$ is the set of places that you've been. If we take the set $A - B$, this would give the set of places that I have been that you hadn't, and if you take the set $B - A$ it would give the set of places that you have been that I hadn't. These two sets, taken together, are quite interesting, because they represent fun places to talk about, since one of us would always be interested in hearing what the other had to say. Using just the operators we've talked about so far, we could describe this overall set as $(B - A) \cup (A - B)$. For simplicity, though, we usually define one final operation on sets that makes this concept easier to convey: the *symmetric difference*.

> The ***set symmetric difference*** of two sets $A$ and $B$, denoted $A \Delta B$, is the set of elements that are contained in exactly one of $A$ or $B$, but not both.

For example, $\{1, 2, 3\} \Delta \{3, 4, 5\} = \{1, 2, 4, 5\}$, since 1 and 2 are in $\{1, 2, 3\}$ but not $\{3, 4, 5\}$ and 4 and 5 are in $\{3, 4, 5\}$ but not $\{1, 2, 3\}$.

# 1.3 Special Sets

So far, we have described sets by explicitly listing off all of their elements: for example, {cat, dog, ibex}, or $\{1, 2, 3\}$. But what if we wanted to consider a collection of things that is too big to be listed off this way? For example, consider all the integers, of which there are infinitely many. Could we gather them together into a set? What about the set of all possible English sentences, which is also infinitely huge? Can we make a set out of them?

It turns out that the answer to both of these questions is "yes," and sets can contain infinitely many elements. But how would we describe such a set? Let's begin by trying to describe the set of all integers. We could try writing this set as

$$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

which does indeed convey our intention. However, this isn't mathematically *rigorous*. For example, is this the set

$$\{\ldots, -11, -7, -5, -3, -2, -1, 0, 1, 2, 3, 5, 7, 11, \ldots\}$$

Or the set

$$\{\ldots, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, \ldots\}$$

Or even the set

$$\{\ldots -16, -8, -4, -2, -1, 0, 1, 2, 3, 4, 5 \ldots, \}$$

When working with complex mathematics, it's important that we be precise with our notation. Although writing out a series of numbers with ellipses to indicate "and so on and so forth" might convey our intentions well in some cases, we might end up accidentally being ambiguous.

To standardize terminology, mathematicians have invented a special symbol used to denote the set of all integers: the symbol $\mathbb{Z}$.

> The ***set of all integers*** is denoted $\mathbb{Z}$. Intuitively, it is the set $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$

For example, $0 \in \mathbb{Z}$, $-137 \in \mathbb{Z}$, and $42 \in \mathbb{Z}$, but $1.37 \notin \mathbb{Z}$, cat $\notin \mathbb{Z}$, and $\{1, 2, 3\} \notin \mathbb{Z}$. The integers are whole numbers, which don't have any fractional parts.

When reading a statement like $x \in \mathbb{Z}$ aloud, it's perfectly fine to read it as "$x$ in Z," but it's more common to read such a statement as "$x$ is an integer," abstracting away from the set-theoretic definition to an (equivalent) but more readable version.

Since $\mathbb{Z}$ really is the set of all integers, all of the set operations we've developed so far apply to it. For example, we could consider the set $\mathbb{Z} \cap \{1, 1.5, 2, 2.5\}$, which is the set $\{1, 2\}$. We can also compute the union of $\mathbb{Z}$ and some other set. For example, $\mathbb{Z} \cup \{1, 2, 3\}$ is the set $\mathbb{Z}$, since $1 \in \mathbb{Z}$, $2 \in \mathbb{Z}$, and $3 \in \mathbb{Z}$.

You might be wondering – why $\mathbb{Z}$? This is from the German word "Zahlen," meaning "numbers."* Much of modern mathematics has its history in Germany, so many terms that we'll encounter in the course of this book (for example, "Entscheidungsproblem") come from German. Much older terms tend to come from Latin or Greek, while results from the 8th through 13th centuries are often Arabic (for example, "algebra" derives from the title كتاب المختصر في حساب الجبر والمقابلة of a book written in the 9th century by Persian mathematician al-Khwarizmi, whose name is the source of the word "algorithm.") It's interesting to see how the languages used in mathematics align with major world intellectual centers.

While in mathematics the integers appear just about everywhere, in computer science they don't arise as frequently as you might expect. Most languages don't allow for negative array indices. Strings can't have a negative number of characters in them. A loop never runs -3 times. More commonly, in computer science, we find ourselves working with just the numbers 0, 1, 2, 3, …, etc. These numbers are called the ***natural numbers***, and represent answers to questions of the form "how many?" Because natural numbers are so ubiquitous in computing, the set of all natural numbers is particularly important:

> The ***set of all natural numbers***, denoted $\mathbb{N}$, is the set $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$

For example, $0 \in \mathbb{N}$, $137 \in \mathbb{N}$, but $-3 \notin \mathbb{N}$, $1.1 \notin \mathbb{N}$, and $\{1, 2\} \notin \mathbb{N}$. As with $\mathbb{Z}$, we might read $x \in \mathbb{N}$ as either "x in N" or as "$x$ is a natural number."

---

* A reader pointed out that the German words "Zahlen" (with an upper-case Z) and "zahlen" (with a lower-case z) actually have different meanings. "Zahlen" means "numbers," while "zahlen" is the infinitive "to pay."

The natural numbers arise frequently in computing as ways of counting loop iterations, the number of nodes in a binary tree, the number of instructions executed by a program, etc.

Before we move on, I should point out that while there is a definite consensus on what $\mathbb{Z}$ is, there is not a universally-accepted definition of $\mathbb{N}$. Some mathematicians treat 0 as a natural number, while others do not. Thus you may find that some authors consider

$$\mathbb{N} = \{0, 1, 2, 3, \ldots\}$$

while others treat $\mathbb{N}$ as

$$\mathbb{N} = \{1, 2, 3, \ldots\}$$

For the purposes of this course, **we will treat 0 as a natural number**, so

- the smallest natural number is 0, and (appropriately)
- $0 \in \mathbb{N}$.

In some cases we may want to consider the set of natural numbers other than zero. We denote this set $\mathbb{N}^+$.

> The ***set of positive natural numbers*** $\mathbb{N}^+$ is the set $\mathbb{N}^+ = \{1, 2, 3, \ldots\}$

Thus $137 \in \mathbb{N}$, but $0 \notin \mathbb{N}^+$.

There are other important sets that arise frequently in mathematics and that will appear from time to time in our exploration of the mathematical foundations of computing. We should consider the set of ***real numbers***, numbers representing arbitrary measurements. For example, you might be 1.7234 meters tall, or weigh 70.22 kilograms. Numbers like $\pi$ and $e$ are real numbers, as are numbers like the square root of two. The set of all real numbers is so important in mathematics that we give it a special symbol.

> The ***set of all real numbers*** is denoted $\mathbb{R}$.

The sets $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ are all quite different from the other sets we've seen so far in that they contain infinitely many elements. We will return to this topic later, but we do need one final pair of definitions:

> A ***finite set*** is a set containing only finitely many elements. An ***infinite set*** is a set containing infinitely many elements.

# 1.4 Set-Builder Notation

## 1.4.1 Filtering Sets

So far, we have seen how to use the primitive set operations (union, intersection, difference, and symmetric difference) to combine together sets into other sets. However, more commonly, we are interested in defining a set not by combining together existing sets, but by gathering together all objects that share some common property. It would be nice, for example, to be able to just say something like “the set of all even numbers”

or “the set of legal Java programs.” For this, we have a tool called *set-builder notation* which allows us to define a set by describing some property common to all of the elements of that set.

Before we go into a formal definition of set-builder notation, let's see some examples. First, here's how we might define the set of even natural numbers:

$$\{ n \mid n \in \mathbb{N} \text{ and } n \text{ is even} \}$$

You can read this aloud as “the set of all $n$, where $n$ is a natural number and $n$ is even.” Similarly, we could define the set of positive real numbers like this:

$$\{ x \mid x \in \mathbb{R} \text{ and } x > 0 \}$$

This would be read as “the set of all $x$, where $x$ is a real number and $x$ is greater than 0.”

Leaving the realm of pure mathematics, we could also consider a set like this one:

$$\{ p \mid p \text{ is a legal Java program} \}$$

If you'll notice, each of these sets is defined using the following pattern:

$$\{ \textit{variable} \mid \textit{conditions on that variable} \}$$

Let's dissect each of the above sets one at a time to see what they mean and how to read them. First, we defined the set of even natural numbers this way:

$$\{ n \mid n \in \mathbb{N} \text{ and } n \text{ is even} \}$$

Here, this definition says that this is the set formed by taking every choice of $n$ where $n \in \mathbb{N}$ (that is, $n$ is a natural number) and $n$ is even. Consequently, this is the set $\{0, 2, 4, 6, 8, \ldots\}$.

The set

$$\{ x \mid x \in \mathbb{R} \text{ and } x > 0 \}$$

can similarly be read off as “the set of all $x$ where $x$ is a real number and $x$ is greater than zero,” which filters the set of real numbers down to just the positive real numbers.

Of the sets listed above, this set was the least mathematically precise:

$$\{ p \mid p \text{ is a legal Java program} \}$$

However, it's a perfectly reasonable way to define a set: we just gather up all the legal Java programs (of which there are infinitely many) and put them into a single set.

When using set-builder notation, the name of the variable chosen does not matter. This means that all of the following are equivalent to one another:

$$\{ x \mid x \in \mathbb{R} \text{ and } x > 0 \}$$

$$\{ y \mid y \in \mathbb{R} \text{ and } y > 0 \}$$

$$\{ z \mid z \in \mathbb{R} \text{ and } z > 0 \}$$

Using set-builder notation, it's actually possible to define many of the special sets from the previous section in terms of one another. For example, we can define the set $\mathbb{N}$ as follows:

$$\mathbb{N} = \{ x \mid x \in \mathbb{Z} \text{ and } x \geq 0 \}$$

That is, the set of all natural numbers ($\mathbb{N}$) is the set of all $x$ such that $x$ is an integer and $x$ is nonnegative. This precisely matches our intuition about what the natural numbers ought to be. Similarly, we can define the set $\mathbb{N}^+$ as

$$\mathbb{N}^+ = \{ n \mid n \in \mathbb{N} \text{ and } n \neq 0 \}$$

Since this describes the set of all $n$ such that $n$ is a nonzero natural number.

So far, all of the examples above with set-builder notation have started with an infinite set and ended with an infinite set. However, set-builder notation can be used to construct finite sets as well. For example, the set

$$\{ n \mid n \in \mathbb{N}, n \text{ is even, and } n < 10 \}$$

has just five elements: 0, 2, 4, 6, and 8.

To formalize our definition of set-builder notation, we need to introduce the notion of a *predicate*:

> A ***predicate*** is a statement about some object $x$ that is either true or false.

For example, the statement "$x < 0$" is a predicate that is true if $x$ is less than zero and false otherwise. The statement "$x$ is an even number" is a predicate that is true if $x$ is an even number and is false otherwise. We can build far more elaborate predicates as well – for example, the predicate "$p$ is a legal Java program that prints out a random number" would be true for Java programs that print random numbers and false otherwise. Interestingly, it's not required that a predicate be checkable by a computer program. As long as a predicate always evaluates to either true or false – regardless of how we'd actually go about verifying which of the two it was – it's a valid predicate.

Given our definition of a predicate, we can formalize the definition of set-builder notation here:

> The set $\{ x \mid \mathbf{P}(x) \}$ is the set of all $x$ such that $\mathbf{P}(x)$ is true.

It turns out that allowing us to define sets this way can, in some cases, lead to *paradoxical sets*, sets that cannot possibly exist. We'll discuss this later on when we talk about **Russell's Paradox**. However, in practical usage, it's almost universally safe to just use this simple set-builder notation.

### 1.4.2 Transforming Sets

You can think of this version of set-builder notation as some sort of filter that is used to gather together all of the objects satisfying some property. However, it's also possible to use set-builder notation as a way of applying a transformation to the elements of one set to convert them into a different set. For example, suppose that we want to describe the set of all perfect squares – that is, natural numbers like $0 = 0^2$, $1 = 1^2$, $4 = 2^2$, $9 = 3^2$, $16 = 4^2$, etc. Using set-builder notation, we can do so, though it's a bit awkward:

$$\{ n \mid \text{there is some } m \in \mathbb{N} \text{ such that } n = m^2 \}$$

That is, the set of all numbers $n$ where, for some natural number $m$, $n$ is the square of $m$. This feels a bit awkward and forced, because we need to describe some property that's shared by all the members of the set, rather than the way in which those elements are generated. As a computer programmer, you would probably be more likely to think about the set of perfect squares more constructively by showing how to build the set of perfect squares out of some other set. In fact, this is so common that there is a variant of set-builder notation that does just this. Here's an alternative way to define the set of all perfect squares:

$$\{ n^2 \mid n \in \mathbb{N} \}$$

This set can be read as "the set of all $n^2$, where $n$ is a natural number." In other words, rather than building the set by describing all the elements in it, we describe the set by showing how to apply a transformation to all objects matching some criterion. Here, the criterion is "$n$ is a natural number," and the transformation is "compute $n^2$."

As another example of this type of notation, suppose that we want to build up the set of the numbers 0, ½, 1, $^3/_2$, 2, $^5/_2$, etc. out to infinity. Using the simple version of set-builder notation, we could write this set as

$$\{ x \mid \text{there is some } n \in \mathbb{N} \text{ such that } x = n / 2 \}$$

That is, this set is the set of all numbers $x$ where $x$ is some natural number divided by two. This feels forced, and so we might use this alternative notation instead:

$$\{ n / 2 \mid n \in \mathbb{N} \}$$

That is, the set of numbers of the form $n / 2$, where $n$ is a natural number. Here, we *transform* the set $\mathbb{N}$ by dividing each of its elements by two.

It's possible to perform transformations on multiple sets at once when using set-builder notation. For example, let's let the set $A = \{1, 2, 3\}$ and the set $B = \{10, 20, 30\}$. Then consider the following set:

$$C = \{ a + b \mid a \in A \text{ and } b \in B \}$$

This set is defined as follows: for any combination of an element $a \in A$ and an element $b \in B$, the set $C$ contains the number $a + b$. For example, since $1 \in A$ and $10 \in B$, the number $1 + 10 = 11$ must be an element of $C$. It turns out that since there are three elements of $A$ and three elements of $B$, there are nine possible combinations of those elements:

| | **10** | **20** | **30** |
|---|---|---|---|
| **1** | 11 | 21 | 31 |
| **2** | 12 | 22 | 32 |
| **3** | 13 | 23 | 33 |

This means that our set $C$ is

$$C = \{ a + b \mid a \in A \text{ and } b \in B \} = \{ 11, 12, 13, 21, 22, 23, 31, 32, 33 \}$$

# 1.5 Relations on Sets

## 1.5.1 Set Equality

We now have ways of describing collections and of forming new collections out of old ones. However, we don't (as of yet) have a way of comparing different collections. How do we know if two sets are equal to one another?

As mentioned earlier, a set is an unordered collection of distinct elements. We say that two sets are equal if they have exactly the same elements as one another.

If $A$ and $B$ are sets, then $A = B$ precisely when they have the same elements as one another. This definition is sometimes called the ***axiom of extensionality***.

For example, under this definition, $\{1, 2, 3\} = \{2, 3, 1\} = \{3, 1, 2\}$, since all of these sets have the same elements. Similarly, $\{1\} = \{1, 1, 1\}$, because both sets have the same elements (remember that a set either contains something or it does not, so duplicates are not allowed). This also means that

$$\mathbb{N} = \{ x \mid x \in \mathbb{Z} \text{ and } x \geq 0 \}$$

since the sets have the same elements. It is important to note that the manner in which two sets are described has absolutely no bearing on whether or not they are equal; all that matters is what the two sets contain. In other words, it's not what's on the outside (the description of the sets) that counts; it's what's on the inside (what those sets actually contain).

Because two sets are equal precisely when they contain the same elements, we can get a better feeling for why we call $\varnothing$ *the* empty set as opposed to *an* empty set (that is, why there's only one empty set, as opposed to a whole bunch of different sets that are all empty). The reason for this is that, by our definition of set equality, two sets are equal precisely when they contain the same elements. This means that if you take any two sets that are empty, they must be equal to one another, since they contain the same elements (namely, no elements at all).

## 1.5.2 Subsets and Supersets

Suppose that you're organizing your music library. You can think of one set $M$ as consisting of all of the songs that you own. Some of those songs are songs that you actually like to listen to, which we could denote $F$ for "Favorite." If we think about the relationship between the sets $M$ and $F$ you can quickly see that $M$ contains everything that $F$ contains, since $M$ is the set of all songs you own while $F$ is only your favorite songs. It's possible that $F = M$, if you only own your favorite songs, but in all likelihood your music library probably contains more songs than just your favorites. In this case, what is the relation between $M$ and $F$? Since $M$ contains everything that $F$ does, plus (potentially) quite a lot more, we say that $M$ is a superset of $F$. Conversely, $F$ is a subset of $M$. We can formalize these definitions below:

A set $A$ is a ***subset*** of another set $B$ if every element of $A$ is also contained in $B$. In other words, $A$ is a subset of $B$ precisely if every time $x \in A$, then $x \in B$ is true. If $A$ is a subset of $B$, we write $A \subseteq B$.

If $A \subseteq B$ (that is, $A$ is a subset of $B$), then we say that $B$ is a ***superset*** of $A$. We denote this by writing $B \supseteq A$.

For example, $\{1, 2\} \subseteq \{1, 2, 3\}$, since every element of $\{1, 2\}$ is also an element of $\{1, 2, 3\}$; specifically, $1 \in \{1, 2, 3\}$ and $2 \in \{1, 2, 3\}$. Also, $\{4, 5, 6\} \supseteq \{4\}$ because every element of $\{4\}$ is an element of the set$\{4, 5, 6\}$, since $4 \in \{4, 5, 6\}$. Additionally, we have that $\mathbb{N} \subseteq \mathbb{Z}$, since every natural number is also an integer, and $\mathbb{Z} \subseteq \mathbb{R}$, since every integer is also a real number.

Given any two sets, there's no guarantee that one of them must be a subset of the other. For example, consider the sets $\{1, 2, 3\}$ and $\{\text{cat, dog, ibex}\}$. In this case, neither set is a subset of the other, and neither set is a superset of the other.

By our definition of a subset, any set $A$ is a subset of itself, because it's fairly obviously true that every element of $A$ is also an element of $A$. For example, {cat, dog, ibex} ⊆ {cat, dog, ibex} because cat ∈ {cat, dog, ibex}, dog ∈ {cat, dog, ibex}, and ibex ∈ {cat, dog, ibex}. Sometimes when talking about subsets and supersets of a set $A$, we want to exclude $A$ itself from consideration. For this purpose, we have the notion of a *strict subset* and *strict superset*:

> A set $A$ is a ***strict subset*** of $B$ if $A \subseteq B$ and $A \neq B$. If $A$ is a strict subset of $B$, we denote this by writing $A \subset B$.
>
> If $A \subset B$, we say that $B$ is a ***strict superset*** of $A$. In this case, we write $B \supset A$.

For example, $\{1, 2\} \subset \{1, 2, 3\}$ because $\{1, 2\} \subseteq \{1, 2, 3\}$ and $\{1, 2\} \neq \{1, 2, 3\}$. However, $\{1, 2, 3\}$ is not a strict subset of itself.

#### 1.5.2.1 The Empty Set and Vacuous Truths

How does the empty set Ø interact with subsets? Consider any set $S$. Is the empty set a subset of $S$? Recall our definition of subset:

*$A \subseteq B$ precisely when every element of $A$ is also an element of $B$.*

The empty set doesn't contain any elements, so how does it interact with the above claim? If we plug Ø and the set $S$ into the above, we get the following:

Ø ⊆ $S$ if every element of Ø is an element of $S$.

Take a look at that last bit - "if every element of Ø is an element of $S$." What does this mean here? After all, there aren't any elements of Ø, because Ø doesn't contain any elements! Given this, is the above statement true or false? There are two ways we can think about this:

1. Since Ø contains no elements, the claim "every element of Ø is an element of $S$" is **false**, because we can't find a single example of an element of Ø that **is** contained in S.
2. Since Ø contains no elements, the claim "every element of Ø is an element of $S$" is **true**, because we can't find a single example of an element of Ø that **isn't** contained in $S$.

So which line of reasoning is mathematically correct? It turns out that it's the second of these two approaches, and indeed it is **true** that Ø ⊆ $S$. To understand why, we need to introduce the idea of a **vacuous truth**. Informally, a statement is vacuously true if it's true simply because it doesn't actually assert anything. For example, consider the statement "if I am a dinosaur, then the moon is on fire." This statement is completely meaningless, since the statement "I am a dinosaur" is false. Consequently, the statement "if I am a dinosaur, then the moon is on fire" doesn't actually assert anything, because I'm not a dinosaur. Similarly, consider the statement "if 1 = 0, then 3 = 5." This too doesn't actually assert anything, because we know that 1 ≠ 0.

Interestingly, mathematically speaking, the statements "if I am a dinosaur, then the moon is on fire" and "if 1 = 0, then 3 = 5" are both considered true statements! They are called *vacuously true* because although they're considered true statements, they're *meaningless* true statements that don't actually provide any new information or insights. More formally:

> The statement "if *P*, then *Q*" is ***vacuously true*** if *P* is always false.

There are many reasons to argue in favor of or against vacuous truths. As you'll see later on as we discuss formal logic, vacuous truth dramatically simplifies many arguments and makes it possible to reason about large classes of objects in a way that more naturally matches our intuitions. That said, it does have its idiosyncrasies, as it makes statements that might seem meaningless, such as "if 1 = 0, then 5 = 3" true. (That said, that previous statement actually does make sense – if you believe that 1 = 0, then you can multiply both sides by two to get that 2 = 0, and can then add three to both sides to get that 5 = 3.)

Let's consider another example: Are all unicorns pink? Well, that's an odd question – there aren't any unicorns in the first place, so how could we possibly know what color they are? But, if you think about it, the statement "all unicorns are pink" should either be true or false.* Which one is it? One option would be to try rewriting the statement "all unicorns are pink" in a slightly different manner – instead, let's say "if *x* is a unicorn, then *x* is pink." This statement conveys exactly the same idea as our original statement, but is phrased as an "if … then" statement. When we write it this way, we can think back to the definition of a vacuous truth. Since the statement "*x* is a unicorn" is never true – there aren't any unicorns! – then the statement "if *x* is a unicorn, then *x* is pink" ends up being a true statement because it's vacuously true. More generally:

> The statement "Every *X* has property *Y*" is (vacuously) true if there are no *X*'s.

Let's return to our original question: is Ø a subset of any set *S*? Recall that Ø is a subset of *S* if every element of Ø is also an element of *S*. But the statement "every element of Ø is an element of *S*" is vacuously true, because there are no elements of Ø! As a result, we have that

> For any set *S*, $\varnothing \subseteq S$.

This means that $\varnothing \subseteq \{1, 2, 3\}$, $\varnothing \subseteq \{\text{cat, dog, ibex}\}$, $\varnothing \subseteq \mathbb{N}$, and even $\varnothing \subseteq \varnothing$.

## 1.6 The Power Set

Given any set *S*, we know that some sets are subsets of *S* (there's always at least Ø as an option), while others are not. For example, the set {1, 2} has four subsets:

- Ø, which is a subset of every set,
- {1},
- {2}, and
- {1, 2}, since every set is a subset of itself.

---

* In case you're thinking "but it could be neither true nor false!," you are not alone! At the turn of the 20th century, a branch of logic arose called ***intuitionistic logic*** that held as a tenet that not all statements are true or false – some might be neither. In intuitionistic logic, there is no concept of a vacuous truth, and statements like "if I am a dinosaur, then the moon is on fire" would simply neither be true nor false. Intuitionistic logic has many applications in computer science, but has generally fallen out of favor in the mathematical community.

We know that sets can contain other sets, so we may want to think about the set that contains all four of these subsets as elements. This is the set

$\{\varnothing, \{1\}, \{2\}, \{1, 2\}\}$

More generally, we can think about taking an arbitrary set $S$ and listing off all its subsets. For example, the subsets of $\{1, 2, 3\}$ are

$$\varnothing \quad \{1\} \quad \{2\} \quad \{3\} \quad \{1, 2\} \quad \{1, 3\} \quad \{2, 3\} \quad \{1, 2, 3\}$$

Note that there are eight subsets here. The subsets of $\{1, 2, 3, 4\}$ are

$$\varnothing \quad \{1\} \quad \{2\} \quad \{3\} \quad \{4\} \quad \{1, 2\} \quad \{1, 3\} \quad \{1, 4\} \quad \{2, 3\} \quad \{2, 4\} \quad \{3, 4\} \quad \{1, 2, 3\} \quad \{1, 2, 4\} \quad \{1, 3, 4\} \quad \{2, 3, 4\} \quad \{1, 2, 3, 4\}$$

Note that there are 16 subsets here. In some cases there may be infinitely many subsets – for instance, the set $\mathbb{N}$ has subsets $\varnothing$, then infinitely many subsets with just one element ($\{0\}$, $\{1\}$, $\{2\}$, etc.), then infinitely many subsets with just two elements ($\{0, 1\}$, $\{0, 2\}$, …, $\{1, 2\}$, $\{1, 3\}$, etc.), etc., and even an infinite number of subsets with infinitely many elements (this is a bit weird, so hold tight... we'll get there soon!) In fact, there are so many subsets that it's difficult to even come up with a way of listing them in any reasonable order! We'll talk about why this is toward the end of this chapter.

Although a given set may have a lot of subsets, for any set $S$ we can talk about the set of all subsets of $S$. This set, called the *power set*, has many important applications, as we'll see later on. But first, a definition is in order.

> The ***power set*** of a set $S$, denoted $\wp(S)$, is the set of all subsets of $S$.

For example, $\wp(\{1, 2\}) = \{\varnothing, \{1\}, \{2\}, \{1, 2\}\}$, since those four sets are all of the subsets of $\{1, 2\}$.

We can write out a formal mathematical definition of $\wp(S)$ using set-builder notation. Let's see how we might go about doing this. We can start out with a very informal definition, like this one:

$\wp(S)$ = "The set of all subsets of $S$"

Now, let's see how we might rewrite this in set-builder notation. First, it might help to rewrite our informal statement "The set of all subsets of $S$" in a way that's more amenable to set-builder notation. Since set-builder notation has the form { *variable* | *conditions* }, we might want to take the information statement "The set of all subsets of $S$" like this:

$\wp(S)$ = "The set of all $T$, where $T$ is a subset of $S$."

Introducing this new variable $T$ makes the English a bit more verbose, but makes it easier to convert the above into a nice statement in set-builder notation. In particular, we can translate the above from English to set-builder notation as

$$\wp(S) = \{ T \mid T \text{ is a subset of } S \}$$

Finally, we can replace the English "is a subset of" with our $\subseteq$ symbol, which means the same thing but is more mathematically precise. Consequently, we could define the power set in set-builder notation as follows:

$$\wp(S) = \{ T \mid T \subseteq S \}$$

What is $\wp(\varnothing)$? This would be the set of all subsets of $\varnothing$, so if we can determine all these subsets, we could gather them together to form $\wp(\varnothing)$. We know that $\varnothing \subseteq \varnothing$, since the empty set is a subset of every set. Are there any other subsets of $\varnothing$? The answer is no. Any set $S$ other than $\varnothing$ has to have at least one element in it, meaning that it can't be a subset of $\varnothing$, which has no elements in it. Consequently, the only subset of $\varnothing$ is $\varnothing$. Since the power set of $\varnothing$ is the set of all of $\varnothing$'s subsets, it's the set containing $\varnothing$. In other words, we have $\wp(\varnothing) = \{\varnothing\}$.

Note that $\{\varnothing\}$ and $\varnothing$ are not the same set. The first of these sets contains one element, which is the empty set, while the latter contains nothing at all. This means that $\wp(\varnothing) \neq \varnothing$.

The power set is a mathematically interesting object, and its existence leads to an extraordinary result called Cantor's Theorem that we will discuss at the end of this chapter.

# 1.7 Cardinality

## 1.7.1 What is Cardinality?

When working with sets, it's natural to ask how many elements are in a set. In some cases, it's easy to see: for example, $\{1, 2, 3, 4, 5\}$ contains five elements, while $\varnothing$ contains none. In others, it's less clear – how many elements are in $\mathbb{N}$, $\mathbb{Z}$, or $\mathbb{R}$ for example? How about the set of all perfect squares? In order to discuss how "large" a set is, we will introduce the notion of set cardinality:

> The ***cardinality*** of a set is a measure of the size of the set. We denote the cardinality of set $A$ as $|A|$.

Informally, the cardinality of a set gives us a way to compare the relative sizes of various sets. For example, if we consider the sets $\{1, 2, 3\}$ and {cat, dog, ibex}, while neither set is a subset of the other, they do have the same size. On the other hand, we can say that $\mathbb{N}$ is a much, *much* bigger set than either of these two sets.

The above definition of cardinality doesn't actually say how to find the cardinality of a set. It turns out that there is a very elegant definition of cardinality that we will introduce in a short while. For now, though, we will consider two cases: the cardinalities of finite sets, and the cardinalities of infinite sets.

For finite sets, the cardinality of the set is defined simply:

> The cardinality of a finite set is the number of elements in that set.

For example:

- $|\varnothing| = 0$
- $|\{7\}| = 1$
- $|\{\text{cat, dog, ibex}\}| = 3$
- $|\{ n \mid n \in \mathbb{N}, n < 10 \}| = 10$

Notice that the cardinality of a finite set is always an integer – we can't have a set with, say, three-and-a-half elements in it. More specifically, the cardinality of a finite set is a natural number, because we also will never have a negative number of elements in a set; what would be an example of a set with, say, negative four elements in it?

The natural numbers are often used precisely because they can be used to count things, and when we use the natural numbers to count how many elements are in a set, we often refer to them as "*finite cardinalities*," since they are used as cardinalities (measuring how many elements are in a set), and they are finite. In fact, one definition of $\mathbb{N}$ is as the set of finite cardinalities, highlighting that the natural numbers can be used to count.

When we work with infinite cardinalities, however, we can't use the natural numbers to count up how many elements are in a set. For example, what natural number is equal to $|\mathbb{N}|$? It turns out that saying "infinity" would be mathematically incorrect here. Mathematicians don't tend to think of "infinity" as being a number at all, but rather a limit toward which a series of numbers approaches. As you count up 0, 1, 2, 3, etc. you tend toward infinity, but you can never actually reach it.

If we can't assign a natural number to the cardinality of $\mathbb{N}$, then what can we use? In order to speak about the cardinality of an infinite set, we need to introduce the notion of an *infinite cardinality*. The infinite cardinalities are a special class of values that are used to measure the size of infinitely large sets. Just as we can use the natural numbers to measure the cardinalities of finite sets, the infinite cardinalities are designed specifically to measure the cardinality of infinite sets.

"Wait a minute... infinite cardinalit*ies*? You mean that there's more than one of them?" Well... yes. Yes there are. In fact, there are many different infinite cardinalities, and not all infinite sets are the same size. This is a hugely important and incredibly counterintuitive result from set theory, and we'll discuss it later in this chapter.

So what are the infinite cardinalities? We'll introduce the very first one here:

> The cardinality of $\mathbb{N}$ is $\aleph_0$, pronounced "aleph-nought," "aleph-zero," or "aleph-null." That is, $|\mathbb{N}| = \aleph_0$.

In case you're wondering what the strange $\aleph$ symbol is, this is the letter "aleph," the first letter of the Hebrew alphabet. The mathematician who first developed a rigorous definition of cardinality, Georg Cantor, used this and several other Hebrew letters in the study of set theory, and the notation persists to this day.*

To understand the sheer magnitude of the value implied by $\aleph_0$, you must understand that this infinite cardinality is bigger than all natural numbers. If you think of the absolute largest thing that you've ever seen, it is smaller than $\aleph_0$. $\aleph_0$ is bigger than anything ever built or that ever could be built.

---

* The Hebrew letter $\aleph$ is one of the oldest symbols used in mathematics. It predates Hindu-Arabic numerals – the symbols 0, 1, 2, …, 9 – by at least five hundred years.

## 1.7.2 The Difficulty With Infinite Cardinalities

With $\aleph_0$, we have a way of talking about $|\mathbb{N}|$, the number of natural numbers. However, we still don't have an answer to the following questions:

- How many integers are there (what is $|\mathbb{Z}|$)?
- How many real numbers are there (what is $|\mathbb{R}|$)?
- How many natural numbers are there that are squares of another number (that is, what is $|\{ n^2 \mid n \in \mathbb{N} \}|$)?

All of these quantities are infinite, but are they all equal to $\aleph_0$? Or is the cardinality of these sets some other value?

At first, it might seem that the answer to this question would be that all of these values are $\aleph_0$, since all of these sets are infinite! However, the notion of infinity is a bit trickier than it might initially seem. For example, consider the following thought experiment. Suppose that we draw a line of some length, like this one below:



How many points are on this line? There are infinitely many, because no matter how many points you pick, I can always pick a point in-between two adjacent points you've drawn to get a new point. Now, consider this other line:



How many points are on this line? Well, again, it's infinite, but it seems as though there should be "more" points on this line than on the previous one! What about this square:

It seems like there ought to be more points in this square than on either of the two lines, since the square is big enough to hold infinitely many copies of the longer line.

So what's going on here? This question has interesting historical significance. In 1638, Galileo Galilei published *Two New Sciences*, a treatise describing a large number of important results from physics and a few from mathematics. In this work, he looked at an argument very similar to the previous one and concluded that the only option was that it makes no sense to talk about infinities being greater or lesser than any other infinity. [Gal] About 250 years later, Georg Cantor revisited this topic and came to a different conclusion – that there is no one "infinity," and that there are infinite sets that are indeed larger or smaller than one another! Cantor's argument is now part of the standard mathematical canon, and the means by which he arrived at this conclusion have been used to prove numerous other important and profoundly disturbing mathematical results. We'll touch on this line of reasoning later on.

### 1.7.3 A Formal Definition of Cardinality

In order for us to reason about infinite cardinalities, we need to have some way of formally defining cardinality, or at least to rank the cardinalities of different sets. We'll begin with a way of determining whether two sets have the same number of elements in them.

Intuitively, what does it mean for two sets to have the same number of elements in them? This seems like such a natural concept that it's actually a bit hard to define. But in order to proceed, we'll have to have some way of doing it. The key idea is as follows – if two sets have the same number of elements in them, then we should be able to pair up all of the elements of the two sets with one another. For example, we might say that {1, 2, 3} and {cat, dog, ibex} have the same number of elements because we can pair the elements as follows:

$$1 \leftrightarrow \text{cat}$$

$$2 \leftrightarrow \text{dog}$$

$$3 \leftrightarrow \text{ibex}$$

However, the sets {1, 2, 3} and {cat, dog, ibex, llama} do not have the same number of elements, since no matter how we pair off the elements there will always be some element of {cat, dog, ibex, llama} that isn't paired off. In other words, if two sets have the same cardinality, then we can indeed pair off all their elements, and if one has larger cardinality than the other, we cannot pair off all of the elements. This gives the following sketch of how we might show that two sets are the same size:

> Two sets have the same cardinality if the elements of the sets can be paired off with one another with no elements remaining.

Now, in order to formalize this definition into something mathematically rigorous, we'll have to find some way to pin down precisely what "pairing the elements of the two sets" off means. One way that we can do this is to just pair the elements off by hand. However, for large sets this really isn't feasible. As an example, consider the following two sets:

$$Even = \{ n \mid n \in \mathbb{N} \text{ and } n \text{ is even} \}$$

$$Odd = \{ n \mid n \in \mathbb{N} \text{ and } n \text{ is odd} \}$$

Intuitively, these sets should be the same size as one another, since half of the natural numbers are even and half are odd. But using our idea of pairing up all the elements, how would we show that the two have the same cardinality? One idea might be to pair up the elements like this:

$$0 \leftrightarrow 1$$

$$2 \leftrightarrow 3$$

$$4 \leftrightarrow 5$$

$$6 \leftrightarrow 7$$

$$\dots$$

More generally, given some even number $n$, we could pair it with the odd number $n + 1$. Similarly, given some odd number $n$, we could pair it with the even number $n - 1$. But does this pair off all the elements of both sets? Clearly each even number is associated with just one odd number, but did we remember to cover every odd number, or is some odd number missed? It turns out that we have covered all the odd numbers,

since if we have the odd number $n$, we just subtract one to get the even number $n - 1$ that's paired with it. In other words, this way of pairing off the elements has these two properties:

1. Every element of *Even* is paired with a different element of *Odd*.
2. Every element of *Odd* has some element of *Even* paired with it.

As a result, we know that all of the elements must be paired off – nothing from *Even* can be uncovered because of (1), and nothing from *Odd* can be uncovered because of (2). Consequently, we can conclude that the cardinality of the even numbers and odd numbers are the same.

We have just shown that $|Even| = |Odd|$, but we still don't actually know what either of these values happen to be! In fact, we only know of one infinite cardinality so far: $\aleph_0$, the cardinality of $\mathbb{N}$. If we can try finding some way of relating $\aleph_0$ to $|Even|$ or $|Odd|$, then we would know the cardinalities of these two sets.

Intuitively, we would have that there are twice as many natural numbers as even numbers and twice as many natural numbers as odd numbers, since half the naturals are even and half the naturals are odd. As a result, we would think that, since there are "more" natural numbers than even or odd numbers, that we would have that $|Even| < |\mathbb{N}| = \aleph_0$. But before we jump to that conclusion, let's work out the math and see what happens. We either need to find a way of pairing off the elements of *Even* and $\mathbb{N}$, or prove that no such pairing exists.

Let's see how we might approach this. We know that the set of even numbers is defined like this:

$$Even = \{ n \mid n \in \mathbb{N} \text{ and } n \text{ is even} \}$$

But we can also characterize it in a different way:

$$Even = \{ 2n \mid n \in \mathbb{N} \}$$

This works because every even number is two times some other number – in fact, some authors define it this way. This second presentation of *Even* is particularly interesting, because it shows that we can construct the even numbers as a transformation of the natural numbers, with the natural number $n$ mapping to $2n$. This actually suggests a way that we might try pairing off the even natural numbers with the natural numbers – just associate $n$ with $2n$. For example:

$$0 \leftrightarrow 0$$

$$1 \leftrightarrow 2$$

$$2 \leftrightarrow 4$$

$$3 \leftrightarrow 6$$

$$4 \leftrightarrow 8$$

$$\ldots$$

Wait a minute... it looks like we've just provided a way to pair up all the natural numbers with just the even natural numbers! That would mean that $|Even| = |\mathbb{N}|$! This is a pretty impressive claim, so before we conclude this, let's double-check to make sure that everything works out.

First, do we pair each natural number with a unique even number? In this case, yes we do, because the number $n$ is mapped to $2n$, so if we take any two natural numbers $n$ and $m$ with $n \neq m$, then they map to $2n$ and $2m$ with $2n \neq 2m$. This means that no two natural numbers map to the same even number.

Second, does every even number have some natural number that maps to it? Absolutely – just divide that even number by two.

At this point we're forced to conclude the seemingly preposterous claim that there are the same number of natural numbers and even numbers, even though it feels like there should be twice as many! But despite our intuition rebelling against us, this ends up being mathematically correct, and we have the following result:

> ***Theorem:*** $|\text{Even}| = |\text{Odd}| = |\mathbb{N}| = \aleph_0$

This example should make it clear just how counterintuitive infinity can be. Given two infinite sets, one of which seems like it ought to be larger than the other, we might end up actually finding that the two sets have the same cardinality!

It turns out that this exact same idea can be used to show that the two line segments from earlier on have exactly the same number of points in them. Consider the ranges (0, 1) and (0, 2), which each contain infinitely many real numbers. We will show that $|(0, 1)| = |(0, 2)|$ by finding a way of pairing up all the elements of the two sets. Specifically, we can do this by pairing each element $x$ in the range $|(0, 1)|$ with the element $2x$ in $|(0, 2)|$. This pairs every element of (0, 1) with a unique element of (0, 2), and ensures that every element $z \in (0, 2)$ is paired with some real number in (0, 1) (namely, $z / 2$). So, informally, doubling an infinite set doesn't make the set any bigger. It still has the same (albeit infinite) cardinality.

Let's do another example, one which is attributed to Galileo Galilei. A natural number is called a *perfect square* if it is the square of some natural number. For example, $16 = 4^2$ and $25 = 5^2$ are perfect squares, as are 0, 1, 100, and 144. Now, consider the set of all perfect squares, which we'll call *Squares*:

$$\textit{Squares} = \{ n \mid n \in \mathbb{N} \text{ and } n \text{ is a perfect square} \}$$

These are the numbers 0, 1, 4, 9, 16, 25, 36, etc. An interesting property of the perfect squares is that as they grow larger and larger, the spacing between them grows larger and larger as well. The space between the first two perfect squares is 1, between the second two is 3, between the third two is 5, and more generally between the $n$th and $(n + 1)$st terms is $2n + 1$. In other words, the perfect squares become more and more sparse the further down the number line you go.

It was pretty surprising to see that there are the same number of even natural numbers as natural numbers, since intuitively it feels like there are twice as many natural numbers as even natural numbers. In the case of perfect squares, it seems like there should be substantially fewer perfect squares than natural numbers, because the perfect squares become increasing more rare as we go higher up the number line. But even so, we can find a way of pairing off the natural numbers with the perfect squares by just associating $n$ with $n^2$:

$$0 \leftrightarrow 0$$

$$1 \leftrightarrow 1$$

$$2 \leftrightarrow 4$$

$$3 \leftrightarrow 9$$

$$4 \leftrightarrow 16$$

$$\ldots$$

This associates each natural number $n$ with a unique perfect square, and ensures that each perfect square has some natural number associated with it. From this, we can conclude that

> ***Theorem:*** The cardinality of the set of perfect squares is $\aleph_0$.

This is not at all an obvious or intuitive result! In fact, when Galileo discovered that there must be the same number of perfect squares and natural numbers, his conclusion was that the entire idea of infinite quantities being "smaller" or "larger" than one another was nonsense, since infinite quantities are infinite quantities.

We have previously defined what it means for two sets to have the same size, but interestingly enough we haven't defined what it means for one set to be "bigger" or "smaller" than another. The basic idea behind these definitions is similar to the earlier definition based on pairing off the elements. We'll say that one set is no bigger than some other set if there's a way of pairing off the elements of the first set and the second set without running out of elements from the second set. For example, the set {1, 2} is no bigger than {a, b, c} because we can pair the elements as

$$1 \leftrightarrow a$$

$$2 \leftrightarrow b$$

Note that we're using the term "is no bigger than" rather than "is smaller than," because it's possible to perfectly pair up the elements of two sets of the same cardinality. All we know is that the first set can't be bigger than the second, since if it were we would run out of elements from the second set.

We can formalize this here:

> If $A$ and $B$ are sets, then $|A| \leq |B|$ precisely when each element of $A$ can be paired off with a unique element from $B$.
>
> If $|A| \leq |B|$ and $|A| \neq |B|$, then we say that $|A| < |B|$.

From this definition, we can see that $|\mathbb{N}| \leq |\mathbb{R}|$ (that is, there are no more natural numbers than there are real numbers) because we can pair off each natural number with itself. We can use similar logic to show that $|\mathbb{Z}| \leq |\mathbb{R}|$, since there are no more integers than real numbers.

# 1.8 Cantor's Theorem

In the previous section when we defined cardinality, we saw numerous examples of sets that have the same cardinality as one another. Given this, do all infinite sets have the same cardinality? It turns out that the answer is "no," and in fact there are infinite sets of differing cardinalities. A hugely important result in establishing this is Cantor's Theorem, which gives a way of finding infinitely large sets with different cardinalities.

As you will see, Cantor's theorem has profound implications beyond simple set theory. In fact, the key idea underlying the proof of Cantor's theorem can be used to show, among other things, that

1. There are problems that cannot be solved by computers, and
2. There are true statements that cannot be proven.

These are huge results with a real weight to them. Let's dive into Cantor's theorem to see what they're all about.

## 1.8.1 How Large is the Power Set?

If you'll recall, the power set of a set $S$ (denoted $\wp(S)$) is the set of all subsets of $S$. As you saw before, the power set of a set can be very, very large. For example, the power set of {1, 2, 3, 4} has sixteen elements.

The power set of {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} has over a thousand elements, and the power set of a set with one hundred elements is so huge that it could not be written out on all the sheets of paper ever printed.

For finite sets, we can show that $|\wp(S)| = 2^{|S|}$, meaning that the power set is exponentially larger than the original set. We'll formally prove this later on in this book, but for now we can argue based on the following intuition. In each subset of $S$, every element of $S$ is either present or it isn't. This gives two options for each element of the set. Given any combination of these yes/no answers, we can form some subset of $S$. So how many combinations are there? Let's line up all the elements in some order. There are two options for the first element, two options for the second, etc. all the way up to the very last element. Since each decision is independent of one another, the number of options ends up being $2 \times 2 \times \ldots \times 2 = 2^n$. Interestingly, we can visualize the subsets as being generated this way. For example, given the set {a, b, c}, the subsets are

| a | b | c | Result |
|---|---|---|---|
| Yes | Yes | Yes | {a, b, c} |
| Yes | Yes | No | {a, b} |
| Yes | No | Yes | {a, c} |
| Yes | No | No | {a} |
| No | Yes | Yes | {b, c} |
| No | Yes | No | {b} |
| No | No | Yes | {c} |
| No | No | No | Ø |

In summary, we can conclude the following:

> ***Thoerem:*** If $S$ is a finite set, $|S| < |\wp(S)|$, since $|\wp(S)| = 2^{|S|}$.

This is the first time we've found some operation on sets that produces a set that always has strictly greater cardinality than the original set.

Does this result extend to infinite sets? That is, is it *always* true that $|S| < |\wp(S)|$? We might be tempted to think so based on our analysis of the finite case, but as we've shown before our intuition about the sizes of infinite sets is often wrong. After all, there's the same number of even natural numbers as natural numbers, even though only half the natural numbers are even!

Let's take a minute to outline what we would need to do to prove whether or not this is true. Since this result will have to hold true for all infinite sets, we would need to show that *any* infinite set, whether it's a set of natural numbers, a set of strings, a set of real numbers, a set of other sets, etc., always has fewer elements than its power set. If this result is false, we just need to find a single counterexample. If there is any set $S$ with $|S| \geq |\wp(S)|$, then we can go home and say that the theorem is false. (Of course, being good mathematicians, we'd then probably go ask for which sets the theorem is true!) Amazingly, it turns out that $|S| < |\wp(S)|$, and the proof is a truly marvelous idea called *Cantor's diagonal argument*.

## 1.8.2 Cantor's Diagonal Argument

Cantor's diagonal argument is based on a beautiful and simple idea. We will prove that $|S| < |\wp(S)|$ by showing that no matter what way you try pairing up the elements of $S$ and $\wp(S)$, there is always some element of

$\wp(S)$ (that is, a subset of $S$) that wasn't paired up with anything. To see how the argument works, we'll see an example as applied to a simple finite set. We already know that the power set of this set must be larger than the set itself, but by seeing the diagonal argument in action in a concrete case it will make clearer just how powerful the argument is.

Let's take the simple set {a, b, c}, whose power set is { Ø, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c} }. Now, remember that two sets have the same cardinality if there's a way of pairing up all of the elements of the two sets. Since we know for a fact that the two sets don't have the same cardinality, there's no possible way that we can do this. However, we know this only because we happen to already know the sizes of the two sets. In other words, we know that there must be at least one subset of {a, b, c} that isn't paired up, but without looking at all of the elements in the pairing we can't necessarily find it. The diagonal argument gives an ingenious way of taking any alleged pairing of the elements of $S$ and its power set and producing some set that is not paired up. To see how it works, let's begin by considering some actual pairing of the elements of {a, b, c} and its power set; for example, this one:

$$a \leftrightarrow \{a, b\}$$

$$b \leftrightarrow \varnothing$$

$$c \leftrightarrow \{a, c\}$$

Now, since each subset corresponds to a set of yes/no decisions about whether each element of {a, b, c} is included in the subset, we can construct a two-dimensional grid like this one below:

| | a? | b? | c? |
|---|---|---|---|
| a paired with | **Y** | **Y** | **N** |
| b paired with | **N** | **N** | **N** |
| c paired with | **Y** | **N** | **Y** |

Here, each row represents the set that each element of {a, b, c} is paired with. The first row shows that a is paired with the set that contains a, contains b, but doesn't contain c, namely {a, b} (indeed, a is paired with this set). Similarly, b is paired with the set that doesn't contain a, b, or c, which is the empty set. Finally, c is paired with the set containing a and c but not b: {a, c}.

Notice that this grid has just as many rows as it has columns. This is no coincidence. Since we are pairing the elements of the set {a, b, c} with subsets of {a, b, c}, we will have one row for each of the elements of {a, b, c} (representing the pairing between each element and some subset) and one column for each of the elements of {a, b, c} (representing whether or not that element appears in the paired set). As a result, we can take a look at the main diagonal of this matrix, which runs from the upper-left corner to the lower-right corner. This is highlighted below:

| | **a?** | **b?** | **c?** |
|---|---|---|---|
| **a paired with** | Y | Y | N |
| **b paired with** | N | N | N |
| **c paired with** | Y | N | Y |

Notice that this diagonal has three elements, since there are three rows and three columns (representing the three elements of the set). This means that the diagonal, as a series of Y's and N's, can potentially be inter-

preted as a subset of {a, b, c}! In this case, since it includes a, excludes b, and includes c, then it would correspond to the set {a, c}. This set might already be paired with some element (in this case, it is – it's paired with c), though it doesn't have to be.

Cantor's brilliant trick is the following: suppose that we **complement** the diagonal of this matrix. That is, we'll take the diagonal and flip all the Y's to N's and N's to Y's. In the above case, this gives the following:

| | **a?** | **b?** | **c?** |
|---|---|---|---|
| **a paired with** | Y | Y | N |
| **b paired with** | N | N | N |
| **c paired with** | Y | N | Y |
| **Complemented Diagonal** | **N** | **Y** | **N** |

This complemented diagonal represents some subset of {a, b, c}. In this case, it's the set {b}. Now, does this set appear anywhere in the table? It turns out that we can *guarantee* that this set isn't paired with anything. Here's why. Let's look at the first row of the table. This row can't be the set {b}, because this row and the complemented diagonal disagree at their first position (the first row has a Y, the complemented diagonal has an N). So let's look at the second row. This row can't be the set {b} because it disagrees in the second position – there's an N in the second spot of the second row and a Y in the second spot of the complemented diagonal. Similarly, the third row disagrees in the third position, because there's a Y in the third spot of the third row and an N in the third spot of the complemented diagonal.

The deviousness of complementing the diagonal lies in the fact that we have specifically crafted a set that can't be paired with anything. The reason for this is as follows:

1. Consider any row *n* in the table.
2. That row can't be equal to the complemented diagonal, because it disagrees in the *n*th position.
3. Consequently, no row in the table is equal to the complemented diagonal.

Since the rows of the table represent all of the subsets paired with the elements of {a, b, c}, the fact that none of the rows are equal to the complemented diagonal guarantees us that the set represented by the complemented diagonal cannot possibly be paired up with any of the elements of the set! In other words, this diagonal argument gives us a way to take any pairing of the elements of {a, b, c} and its subsets and producing at least one subset that wasn't paired up. To see this argument in action again, here's another pairing:

$$a \leftrightarrow \{a\}$$

$$b \leftrightarrow \{b\}$$

$$c \leftrightarrow \{a, b\}$$

This gives the following table and complemented diagonal:

| | **a?** | **b?** | **c?** |
|---|---|---|---|
| **a paired with** | Y | N | N |
| **b paired with** | N | Y | N |
| **c paired with** | Y | Y | N |
| **Complemented Diagonal** | **N** | **N** | **Y** |

The complemented diagonal here is {c}, which is missing from the table.

If we didn't already know that the power set of {a, b, c} was bigger than the set {a, b, c}, this diagonal argument would have just proven it – it gives a way of taking any possible pairing of the elements of {a, b, c} with its subsets and shows that after pairing up all the elements of {a, b, c}, there is always some element left uncovered. This same technique can be applied to other sets as well. For example, suppose we have the set {a, b, c, d, e, f}, and this pairing:

$$a \leftrightarrow \{b, c, d\}$$

$$b \leftrightarrow \{e, f\}$$

$$c \leftrightarrow \{a, b, c, d, e, f\}$$

$$d \leftrightarrow \varnothing$$

$$e \leftrightarrow \{a, f\}$$

$$f \leftrightarrow \{b, c, d, e\}$$

We can then build the following table, which has its diagonal and complemented diagonal shown:

| | **a?** | **b?** | **c?** | **d?** | **e?** | **f?** |
|---|---|---|---|---|---|---|
| **a paired with** | **N** | Y | Y | Y | N | N |
| **b paired with** | N | **N** | N | N | Y | Y |
| **c paired with** | Y | Y | **Y** | Y | Y | Y |
| **d paired with** | N | N | N | **N** | N | N |
| **e paired with** | Y | N | N | N | **N** | Y |
| **f paired with** | N | Y | Y | Y | Y | **N** |
| **Complemented Diagonal** | **Y** | **Y** | **N** | **Y** | **Y** | **Y** |

From this, we get that the complemented diagonal is the set {a, b, d, e, f}, which indeed is not in the list of sets described in the pairing.

### 1.8.3 Formalizing the Diagonal Argument

We have just described the intuition behind Cantor's diagonal argument – we can show that in any pairing between a set $S$ and the set $\wp(S)$, there must be some element of $\wp(S)$ that isn't covered by the pairing. However, so far our proof requires us to construct a table representing the pairing whose size is determined by the number of elements in $S$. Given this, will this argument work when we are dealing with infinite sets? We've seen a lot of strange results that appear when working with the infinite, and so it doesn't seem particularly "safe" to assume that this approach, which works in the finite case, scales up to the infinite case.

It turns out, however, that this argument can indeed be applied to infinite sets! However, to do so will require us to be more precise and formal than our reasoning above, in which we just drew a picture. We need to find a way of nicely describing what set is constructed by the diagonal argument without having to draw out a potentially infinite table. Fortunately, there is a nicely straightforward way to do this. Let's consider the previous example:

$$a \leftrightarrow \{b, c, d\}$$

$$b \leftrightarrow \{e, f\}$$

$$c \leftrightarrow \{a, b, c, d, e, f\}$$

$$d \leftrightarrow \varnothing$$

$$e \leftrightarrow \{a, f\}$$

$$f \leftrightarrow \{b, c, d, e\}$$

If we draw this out as a table, we get the following:

| | **a?** | **b?** | **c?** | **d?** | **e?** | **f?** |
|---|---|---|---|---|---|---|
| **a paired with** | **N** | Y | Y | Y | N | N |
| **b paired with** | N | **N** | N | N | Y | Y |
| **c paired with** | Y | Y | **Y** | Y | Y | Y |
| **d paired with** | N | N | N | **N** | N | N |
| **e paired with** | Y | N | N | N | **N** | Y |
| **f paired with** | N | Y | Y | Y | Y | **N** |
| **Complemented Diagonal** | **Y** | **Y** | **N** | **Y** | **Y** | **Y** |

Now, let's think about the diagonal of this table. Notice that each diagonal entry represents whether some element $x \in S$ is paired with a set that contains itself. If the element $x$ is paired with a set that contains it, the entry on the diagonal is a *Y*, and if $x$ is paired with a set that doesn't contain it, then the entry on the diagonal is an *N*. In the complemented diagonal, this is reversed – the complemented diagonal entry is a *Y* if $x$ is paired with a set that doesn't contain $x$, and is an *N* if $x$ is paired with a set that does contain $x$. In other words, we can think about the set defined by the complemented diagonal (let's call it *D*) as follows:

$$D = \{ x \mid \text{there is an } N \text{ in the diagonal entry for } x \}$$

Or, more concretely:

$$D = \{ x \mid x \text{ is paired with a set that does not contain } x \}$$

Now this is interesting – we now have a definition of the diagonal set *D* that doesn't require us to even construct the table! The rule for finding *D* is straightforward: we simply find all the elements of the set *S* that are paired with subsets of *S* that don't contain themselves, then gather those up together into a set. Does this really work? Well, if experience is a guide, then yes! Here are a few pairings from before, along with the associated diagonal set:

| | | | |
|---|---|---|---|
| **Pairing** | a ↔ {a}<br>b ↔ {b}<br>c ↔ {a, b} | a ↔ {a, b}<br>b ↔ Ø<br>c ↔ {a, c} | a ↔ {b, c, d}<br>b ↔ {e, f}<br>c ↔ {a, b, c, d, e, f}<br>d ↔ Ø<br>e ↔ {a, f}<br>f ↔ {b, c, d, e} |
| **Complemented Diagonal Set** | {c} | {b} | {a, b, d, e, f} |

You can (and should!) check that in each case, the complemented diagonal set is indeed the set of elements that aren't paired with a set that contains them. For example, in the first example since a is paired with {a} and b is paired with {b}, neither is included in the complemented diagonal set, while c, which is paired with a set that doesn't contain c, is indeed in the complemented diagonal set.

## 1.8.4 Proving Cantor's Theorem

Given this definition of the diagonal set, we can formally prove Cantor's theorem.

> ***Theorem (Cantor's Theorem):*** For any set $S$, $|S| < |\wp(S)|$.

In order to prove Cantor's theorem, let's think about the definition of what it means for one set to have lesser cardinality than another. This would mean that

$$|S| \leq |\wp(S)| \text{ and } |S| \neq |\wp(S)|$$

We can prove each part of this independently. Cantor's diagonal argument will handle the second case (which we'll handle in a minute), but first let's show that $|S| \leq |\wp(S)|$. How can we show this? To do so, we need to show that we can pair off all of the elements of $S$ with some element in $\wp(S)$. This might seem hard, because we don't actually know what $S$ is; we need to show that for *any* set $S$, it's always possible to find such a pairing. This actually ends up being not too difficult. Note that for each element $x \in S$, the set $\{x\} \subseteq S$. Therefore, $\{x\} \in \wp(S)$. Consequently, one way of pairing up all the elements of $S$ with elements from $\wp(S)$ would be to associate each element $x \in S$ with the element $\{x\} \in \wp(S)$. This ensures that each element of $S$ is paired up with a unique element from $\wp(S)$.

Now, we need to formally prove that $|S| \neq |\wp(S)|$, even though we don't actually know what $S$ is (we're trying to prove the claim for any possible choice of $S$). So what does it mean for $|S| \neq |\wp(S)|$? Well, this would mean that there must be no possible way of pairing off all the elements from the two sets with one another. How can we show that this is impossible? Here, we will employ a technique called ***proof by contradiction***. In a proof by contradiction, we try to show that some statement $P$ is true by doing the following:

- Assume, hypothetically, that $P$ was false.
- Show that this assumption, coupled with sound reasoning, leads us to a conclusion that is obviously false.
- Conclude, therefore, that our assumption must have been wrong, so $P$ is true.

As an example of a proof of this style, suppose that you want to convince someone that it is not raining outside (if it's raining when you're reading this, then my apologies – please bear with me!) One way to convince someone that it's not raining is as follows:

1. Suppose, hypothetically, that it were raining.
2. Therefore, I should be soaking wet from my walk a few minutes ago.
3. But I am not soaking wet from my walk a few minutes ago.
4. Since steps (2) and (3) are logically sound, the only possible problem is step (1).
5. Conclude, therefore, that it's not raining outside.

Here, when proving that $|S| \neq |\wp(S)|$, we will use a proof by contradiction. Suppose, hypothetically, that $|S| = |\wp(S)|$. Then there is a way of pairing up all of the elements of $S$ and $\wp(S)$ together – we don't know what it is, but allegedly it exists. This should mean that any element of $\wp(S)$ that we choose should be paired up with some element of $S$. Since every element of $\wp(S)$ is a subset of $S$ (that's how we defined the power set!), this should mean that if we choose any subset of $S$, it should be paired up with some element from $S$. But now we can employ Cantor's diagonal argument. If it's really true that *any* subset of $S$ must be paired up with some element of $S$, then surely we should be able to find some element paired up with the set

$$D = \{ x \mid x \in S \text{ and } x \text{ is paired with a set that does not contain } x. \}$$

Well, since $D$ allegedly must be paired with something, let's call that element $d$. But if you'll recall from our discussion of the diagonal argument, we should now be able to show that $D$ actually isn't in this table, meaning that there really isn't an element $d$ that it could be paired with. But how do we show this? Here, we'll simply ask the following question: is $d$ contained in $D$? There are two possible options: either $d \in D$ or $d \notin D$. Let's consider these two cases individually.

First, suppose that $d \in D$. Recall that the definition of $D$ says that this means that $d$ would have to be paired with a set that doesn't contain $d$. Since $d$ is paired with $D$, this would have to mean that $d \notin D$, but this is clearly impossible, because we know that in this case $d \in D$. Since if $d \in D$ we conclude that $d \notin D$, we know that it must not be possible for $d \in D$ to be true.

So this means that $d \notin D$. Well, let's see what that means. Since $d \notin D$, then by looking at the definition of the set $D$, we can see that this means that the set that $d$ is paired with must contain $d$. Since $d$ is paired with the set $D$, this would mean that $d \in D$. But this isn't true!

We have just reached a logical contradiction. If $d \in D$, then we know that $d \notin D$, and similarly if $d \notin D$, then $d \in D$. In other words, $D$ contains $d$ if and only if $D$ does not contain $d$. We have reached a logical impossibility.

All of the reasoning we've had up to this point is sound, so we are forced to admit that the only possibility remaining is that our assumption that $|S| = |\wp(S)|$ is incorrect. Consequently, we have proven $|S| \neq |\wp(S)|$. Since earlier we proved $|S| \leq |\wp(S)|$, this collectively proves Cantor's theorem.

# 1.9 Why Cantor's Theorem Matters

We have just proven Cantor's theorem, that the number of subsets of a set $S$ is strictly greater than the number of elements of that set $S$. But why does this matter? It turns out that this is actually a hugely important result with a terrifying corollary. To begin with, note that Cantor's theorem says that there are more subsets of a set than elements of that set, even if the initial set is infinite. This suggests that there is no one concept of “infinity,” and that there are, in fact, different infinitely large quantities, each one infinitely larger than the previous! In fact this means that

- There are more sets of natural numbers than natural numbers ($|\mathbb{N}| < |\wp(\mathbb{N})|$)
- More sets of sets of natural numbers than sets of natural numbers ($|\wp(\mathbb{N})| < |\wp(\wp(\mathbb{N}))|$),
- etc.

The fact that there are different infinitely large numbers has enormous significance to the limits of computing. For example, there are infinitely many problems to solve, and there are infinitely many programs to solve them. But this doesn't mean that there are the same number of problems and solutions! In fact, it might be possible that there are more problems that we might want to solve than there are programs to solve them, even though both are infinite! In fact, this is the case. Let's see why.

# 1.10 The Limits of Computation

Let's begin with a pair of definitions:

> An ***alphabet*** is a set of symbols.

For example, we could talk about the English alphabet as the set $A$ = { A, B, C, D, …, Y, Z, a, b, …, z }. The binary alphabet is the alphabet {0, 1}, and the unary alphabet is the alphabet { 1 }. Given an alphabet, what words can we make from that alphabet? Typically, we will use the term "string" instead of "word:"

> A ***string*** is a finite sequence of symbols drawn from some alphabet.

For example, **hello** is a string drawn from the English alphabet, while 01100001 is a string drawn from the binary alphabet.

Every computer program can be expressed as a string drawn from the appropriate alphabet. The program's source code is a sequence of characters (probably Unicode characters) that are translated into a program using a compiler. In most programming languages, not all strings are legal programs, but many are. As a result, we can say that the number of programs is at most the number of strings, since we can pair up the programs and strings without exhausting all strings (just pair each program with its source code).

Now, let's think about how many problems there are out there that we might want to solve. This really depends on our notion of what a "problem" is, but we don't actually need to have a formal definition of "problem" quite yet. Let's just focus on one type of problem: deciding whether a string has some property. For example, some strings have even length, some are palindromes, some are legal Java programs, some are mathematical proofs, etc. We can think of a "property" of a string as just a set of strings that happen to share that property. For example, we could say that the property of being a palindrome (reading the same forwards and backwards) could be represented by the set

{ a, b, c, …, z, aa, bb, …, zz, aba, aca, ada, … }

While the property of having exactly four letters would be

{ aaaa, aaab, aaac, …, zzzz }

For each of these properties, we might think about writing a program that could determine whether a string has that given property. For example, with a few minutes' effort you could probably sit down and write a program that will check whether a string is a palindrome or contains just four characters, and with more ef-

fort could check if a string encoded a legal computer program, etc. In other words, each property of strings (that is, a set of strings) defines a unique problem – determine whether a given string has that property or not. As a result, the number of sets of strings is no bigger than the total number of problems we might want to solve, since there's at least one problem to solve per set of strings.

This leads to the following line of reasoning:

The number of programs is no bigger than the number of strings.

The number of strings is strictly less than the number of sets of strings (from Cantor's theorem).

The number of properties of strings is bigger than the number of strings.

Since each property of strings gives rise to a problem to solve:

The number of problems is at least the number of properties of strings.

Combining this all together gives the following:

The number of programs is strictly less than the number of problems.

In other words:

**There are more problems than there are programs to solve them.**

We have just proven, without even looking at how computers work or how clever programmers are, that there are problems that cannot possibly be solved by a computer. There are simply too many problems to go around, so even if we wrote all of the infinitely many possible programs, we would not have written a program to solve every problem.

### 1.10.1 What Does This Mean?

At this point, we could throw up our hands and despair. We have just shown the existence of unsolvable problems, problems that can be formulated but not possibly solved.

Unfortunately, it gets worse. Using more advanced set theory, we can show that the infinity of problems so vastly dwarfs the infinity of solutions that if you choose a totally random problem to solve, the chance that you can is 0. Moreover, since there are more problems to solve than possible strings, some of the problems we can't solve may be so complex that there is no way to describe them; after all, a description is a way of pairing a string with a problem!

But there's no way we can give up now. We have shown that there is an infinite abyss of unsolvable problems, but everywhere we look we can see examples of places where computers *have* solved problems.

Rather than viewing this result as a sign of defeat, treat it as a call to arms. Yes, there are infinitely many problems that we *can't* solve, but there are infinitely many problems that we *can* solve as well. What are they? What do they look like? Of the problems we can solve in theory, what can be solved in practice as well? How powerful of a computer would you need to solve them? These are questions of huge practical and theoretical importance, and it's these questions that we will focus on in the rest of this book. In doing so, you'll sharpen your mathematical acumen and will learn how to reason about problems abstractly. You'll learn new ways of thinking about computation and how it can impact your practical programming skills. And you'll see some of the most interesting and fascinating results in all of computer science.

Let's get started!

## 1.11 Chapter Summary

- A set is an unordered collection of distinct elements.
- Sets can be described by listing their elements in some order.
- Sets can also be described using set-builder notation.
- Set can be combined via union, intersection, difference, or symmetric difference.
- Two sets are equal precisely when they have the same elements.
- One set is a subset of another if every element of that set is in the other set.
- The power set of a set is the set of its subsets.
- A statement is vacuously true if its assertion doesn't apply to anything.
- The cardinality of a set is a measure of how many elements are in that set.
- Two sets have the same cardinality if all elements of both sets can be paired up with one another.
- Cantor's diagonal argument can be used to prove Cantor's theorem, that the cardinality of a set is always strictly less than the cardinality of its power set.

# Chapter 2 Mathematical Proof

Last chapter we concluded with Cantor's theorem, the fact that the cardinality of the power set of a set $S$ is always greater than the cardinality of the set $S$ itself. Although we worked through a strong argument that this should be true, did we really "prove" it? What does it mean to prove something, at least in a mathematical sense?

Proofs are at the core of the mathematical foundations of computing. Without proofs we couldn't be certain that any of our results were correct, and our definitions would be little better than an intuition to guide us. Accordingly, before we attempt to explore the limits of computation, we first need to build up the machinery necessary to reason about and firmly establish mathematical results.

Proofs are in many ways like programs – they have their own vocabulary, terminology, and structure, and you will need to train yourself to think differently in order to understand and synthesize them. In this chapter and the ones that follow, we will explore proofs and proof techniques, along with several other concepts that will serve as a "proving ground" for testing out these proof ideas.

One quick note before we continue – because this chapter focuses on how to structure mathematical proofs, some of the results that we'll be proving early on will be pretty trivial. I promise you that the material will get a lot more interesting toward the end of the chapter, and once we make it into Chapter Three the results we will be proving will be much more involved and a lot more interesting. Please don't get the impression that math is painfully boring and pedantic! It's a really fascinating subject, but we need to build up a few techniques before we can jump into the real meat of the material.

## 2.1 What is a Proof?

In order to write a proof, we need to start off by coming up with some sort of definition of the word "proof." Informally, a mathematical proof is a series of logical steps starting with one set of assumptions that ends up concluding that some statement must be true. For example, if we wanted to prove the statement

$$\text{If } x + y = 16, \text{ then either } x \geq 8 \text{ or } y \geq 8$$

then we would begin by assuming that $x + y = 16$, then apply sound logical reasoning until we had arrived at the conclusion that $x \geq 8$ or $y \geq 8$. Similarly, if we wanted to prove that

$$\text{For any set } S, |S| < |\wp(S)|$$

(as we started doing last chapter), we would take as our starting point all of the definitions from set theory – what the power set is, what it means for one set to have smaller cardinality than another, etc. - and would proceed through logical steps to conclude that $|S| < |\wp(S)|$.

Writing a proof is in many ways like writing a computer program. You begin with some base set of things that you know are true (for example, how primitive data types work, how to define classes, etc.), then proceed to use those primitive operations to build up something more complicated. Also like a program, proofs have their own vocabulary, language, structure, and expectations. Unfortunately, unlike programs, there is no "compiler" for proofs that can take in a proof and verify that it's a legal mathematical proof.* Consequently, learning how to write proofs takes time and effort.

---

* Technically speaking such programs exist, but they require the proof to be specified in a very rigid format that is almost never used in formal mathematical proofs.

In this chapter, we will introduce different types of proofs by analyzing real proofs and seeing exactly how they work. We'll also see what *doesn't* work and the sort of logical traps you can easily fall into when writing proofs.

## 2.1.1 Transitioning to Proof-Based Mathematics

The math that you're probably most familiar with is the style of math that you saw in high school. Typically, high school mathematics focuses on *calculation*. For example, you'll probably get problems like this one:

> Two trains are 50 miles apart from one another and on parallel tracks. One train takes off from the station heading at 24 miles per hour, while the other train takes off heading 12 miles per hour. The two trains are driving toward one another. How long will it take for the two trains to pass one another?

Or perhaps something like this:

> Evaluate the integral $\int_0^5 x\sqrt{1+x^2}\,dx$

Or perhaps something like this:

> A square is circumscribed inside a circle of radius 15. The square is then divided into four right triangles whose right angle is the center of the square. What is the sum of the perimeters of these triangles?

Problems like these have exact numeric values associated with them, and the goal of the problem is to work through a calculation to come up with some number. You can memorize different approaches for solving these sorts of problems and build up a repertoire of techniques for solving them. The focus on these problems is determining which calculations to perform.

The math that we will be doing in this course is of a totally different flavor. We will be interested in proving general properties of different types of objects (numbers, data structures, computer programs, etc.) In doing so, we may perform some small calculations from time to time, but our main objective will be establishing a clear line of reasoning that rigorously demonstrates a result. In fact, you'll find that most proofs that we'll be writing read like essays with some mathematical jargon, since the goal will be to describe a rigorous line of reasoning rather than to compute a value.

## 2.1.2 What Can We Assume?

One of the most difficult aspects of writing a proof is determining what you can assume going into the proof. In journals, proofs often assume that the reader is familiar with important results, and often cite them without reviewing why they're true. For our purposes, though, we will deliberately play dumb and start with a very weak set of assumptions. We will prove pretty much everything we need, even if it seems completely obvious, in order to see how to formalize intuitive concepts with a level of mathematical rigor.

In this book, we will assume that whoever is reading one of our proofs knows

1. All definitions introduced so far,
2. All theorems introduced so far, and
3. Basic algebra.

We will not assume anything more than this. For example, we're fine assuming that if $x < y$ and $y < z$, then $x < z$, but we will not assume that for any set $S$, $S \cap \varnothing = \varnothing$ even though this seems "obvious." As we build

up our mathematical repertoire, the set of assumptions we can make will grow, and it will become easier and easier to prove more elaborate results. This is similar to writing libraries in computer programs – although it's difficult and a bit tedious to implement standard data structures like **`ArrayList`** and **`HashMap`**, once you've put in the work to do so it becomes possible to build up off of them and start writing much more intricate and complex programs.

## 2.2 Direct Proofs

Just as it's often easiest to learn how to program by jumping into the middle of a "Hello, World!" program and seeing how it works, it's useful to jump right into some fully worked-out mathematical proofs to see how to structure general proofs.

To begin our descent into proofs, we'll introduce two simple definitions, then see how to prove results about those definitions.

> An integer $x$ is called ***even*** if there is some integer $k$ such that $x = 2k$.
>
> An integer $x$ is called ***odd*** if there is some integer $k$ such that $x = 2k + 1$.

For example, 4 is even since $4 = 2 \times 2$. 8 is even as well, since $8 = 4 \times 2$. 9 is odd, because $9 = 4 \times 2 + 1$. We consider 0 to be an even number, since $0 = 0 \times 2$.

Given this, let's prove the following result:

> ***Theorem:*** If $x$ is even, then $x^2$ is even.

> ***Proof***: Let $x$ be any even integer. Since $x$ is even, there is some integer $k$ such that $x = 2k$. This means that $x^2 = (2k)^2 = 4k^2 = 2(2k^2)$. Since $2k^2$ is an integer, this means that there is some integer $m$ (namely, $2k^2$) such that $x^2 = 2m$. Thus $x^2$ is even. ■

Let's look at how this proof works. The proof proceeds in several clean logical steps, each of which we can analyze independently.

First, note how the proof starts: "Let $x$ be any even integer." The goal of this proof is to show that if $x$ is even, then $x^2$ is even as well. This proof should work no matter what choice of $x$ we make – whether it's 0, 2, 4, 6, 8, etc. In order for our proof to work in this general setting, the proof will proceed by using $x$ as a placeholder for whatever even number we're interested in. If we wanted to convince ourselves that some particular even number has a square that's also even, we could just plug that even number into the proof wherever we were using $x$. For example, if we want to prove that $12^2$ is even, the proof would go like this:

> ***Proof***: Since ***12*** is even, there is some integer $k$ such that ***12*** $= 2k$. (This integer $k$ is the integer 6). This means that ***12***$^2 = (2 \times 6)^2 = 4 \times 6^2 = 2(2 \times 6^2) = 2 \times 72$. Since 72 is an integer, this means that there is some integer $m$ (namely, 72) such that ***12***$^2 = 2m$. Thus ***12***$^2$ is even. ■

All that we've done here is substitute in the number 12 for our choice of $x$. We could substitute in any other even number if we'd like and the proof would still hold. In fact, that's why the proof works – we've shown that no matter what choice of an even number you make for $x$, you can always prove that $x^2$ is even as well.

Let's continue dissecting this proof. After we've decided to let $x$ be a placeholder for whatever even number we'd like, we then say

> Since $x$ is even, there is some integer $k$ such that $x = 2k$

What does this statement mean? Well, we know that $x$ is an even number, which means that it must be twice some other number. We can't really say what that number is, since we don't know what our choice of $x$ is. However, we can say that there is *some* number such that $x$ is twice that number. In order to manipulate that number in this proof, we'll give this number a name (in this proof, we call it $k$). Interestingly, note that nowhere in this sentence do we actually say how to figure out what this value of $k$ is; we just say that it has to exist and move forward. From a programming perspective, this may seem strange – it seems like we'd have to show how to find this number $k$ in order to assert that it exists! But it turns out that it's perfectly fine to just say that it exists and leave it at that. Our definition of an even number is an integer that is equal to twice some other number, so we know for a fact that because $x$ is even, this number $k$ must exist.

At this point, we know that $x = 2k$, and our goal is to show that $x^2$ is even. Let's think about how to do this. To show that $x^2$ is even, we will need to find some integer $m$ such that $x^2 = 2m$. Right now, all that we know is that $x$ is even and, as a result, that $x = 2k$ for some choice of $k$. Since we don't have much else to go on right now, let's try seeing if we can describe $x^2$ in terms of $x$ and $k$. Perhaps doing this will lead us to finding some choice of $m$ that we can make such that $x^2 = 2m$. This leads to the next part of the proof:

> This means that $x^2 = (2k)^2 = 4k^2 = 2(2k^2)$. Since $2k^2$ is an integer, this means that there is some integer $m$ (namely, $2k^2$) such that $x^2 = 2m$

The first of these two sentences is a simple algebraic manipulation. We know that $x = 2k$, so $x^2 = (2k)^2$. If we simplify this, we get $x^2 = 4k^2$, which is in turn equal to $2(2k^2)$. This last step – factoring out the two from the expression – then makes it clearer that $x^2$ is twice some other integer (specifically, the integer $2k^2$). We can then conclude that there is some natural number $m$ such that $x^2 = 2m$, since we've found specifically what that value was. Because we've done this, we can conclude the proof by writing:

> Thus $x^2$ is even. ■

This holds because the definition of an even number is one that can be written as $2m$ for some integer $m$. Notice that we've marked the end of the proof with the special symbol ■, which serves as a marker that we're done. Sometimes you see proofs ended with other statements like “This completes the proof” or “QED” (from the Latin “quod erat demonstrandum,” which translates roughly to “which is what we wanted to show”). Feel free to end your own proofs with one of these three endings.

Let's take a look at an example of another proof:

***Theorem:*** If $m$ is even and $n$ is odd, then $mn$ is even.

***Proof***: Let $m$ be any even number and $n$ be any odd number. Then $m = 2r$ for some integer $r$, and $n = 2s + 1$ for some integer $s$. This means that $mn = (2r)(2s + 1) = 2(r(2s + 1))$. This means that $mn = 2k$ for some integer $k$ (namely, $r(2s + 1)$), so $mn$ is even. ■

The structure of this proof is similar to the previous proof. We want to show that the claim holds for any choice of even $m$ and odd $n$, so we begin by letting $m$ and $n$ be any even and odd number, respectively. From there, we use the definition of even and odd numbers to write $m = 2r$ and $n$ as $2s + 1$ for some integers $r$ and $s$. As with the previous proof, we don't actually know what these numbers are, but they're guaranteed to exist. After doing some simple arithmetic, we end up seeing that $mn = 2(r(2s + 1))$, and since $r(2s + 1)$ is an integer, we can conclude that $mn$ is twice some integer, and so it must be even.

The above proofs are both instances of ***direct proofs***, in which the proposition to be proven is directly shown to be true by beginning with the assumptions and ending at the conclusions.

### 2.2.1 Proof by Cases

Let's introduce one new definition, which you may be familiar with from your programming background:

The ***parity*** of an integer is whether it is odd or even. Two numbers have the same parity if they are both odd or both even.

For example, 1 and 5 have the same parity, because both of the numbers are odd, and 4 and 14 have the same parity because both 4 and 14 are even. However, 1 and 2 have opposite parity, because 1 is odd and 2 is even.

The following result involves the parity of integers:

***Theorem:*** If $m$ and $n$ have the same parity, then $m + n$ is even.

Before we try to prove this, let's check that it's actually correct by testing it on a few simple examples. We can see that $2 + 6 = 8$ is even, and $1 + 5 = 6$ is even as well. But how would we prove that this is true in the general case? In a sense, we need to prove two separate claims, since if $m$ and $n$ have the same parity, then either both $m$ and $n$ are even or both $m$ and $n$ are odd. The definitions of odd numbers and even numbers aren't the same, and so we have to consider the two options separately. We can do this cleanly in a proof as follows:

**Proof**: Let $m$ and $n$ be any two integers with the same parity. Then there are two cases to consider:

*Case 1:* $m$ and $n$ are even. Then $m = 2r$ for some integer $r$ and $n = 2s$ for some integer $s$. Therefore, $m + n = 2r + 2s = 2(r + s)$. Thus $m + n = 2k$ for some integer $k$ (namely, $r + s$), so $m + n$ is even.

*Case 2:* $m$ and $n$ are odd. Then $m = 2r + 1$ for some integer $r$ and $n = 2s + 1$ for some integer $s$. Therefore, $m + n = 2r + 1 + 2s + 1 = 2r + 2s + 2 = 2(r + s + 1)$. Thus $m + n = 2k$ for some integer $k$ (namely, $r + s + 1$), so $m + n$ is even. ■

Note how this proof is structured as two cases – first, when $m$ and $n$ are even, and second, when $m$ and $n$ are odd. This style of proof is sometimes called a **proof by cases** or a **proof by exhaustion** (because we've exhausted all possibilities and found that the claim is true, not because we're tired of writing proofs). Each of the branches of the proof reads just like a normal proof, but is individually insufficient to prove the general result. Only when we show that in both of the possible cases the result holds can we conclude that the claim is true in general.

When writing a proof by exhaustion, it's critically important to remember to check that you have covered all possible cases! If you have a proof where four options are possible and you only prove three cases, your proof is likely to be incorrect.

Let's see another example of a proof by cases:

***Theorem:*** If $n$ is even and $m$ is an integer, then $n + m$ has the same parity as $m$.

Before proving this, it's always good to check that it works for a few test cases. If we let $n = 4$, then we can see that

- $4 + 3 = 7$, and 7 has the same parity as 3.
- $4 + 6 = 10$, and 10 has the same parity as 6.

Let's see a proof of this result:

***Proof***: Consider any even integer $n$. Now, consider any integer $m$ and the sum $n + m$. We consider two possibilities for $m$:

*Case 1:* $m$ is even. Then $m$ and $n$ have the same parity, so by our previous result (if $m$ and $n$ have the same parity, then $m + n$ is even) we know that $m + n$ is even. Therefore $m$ and $m + n$ have the same parity.

*Case 2:* $m$ is odd. Since $n$ is even, $n = 2r$ for some integer $r$, and since $m$ is odd, $m = 2s + 1$ for some integer $s$. Then $n + m = 2r + 2s + 1 = 2(r + s) + 1$. This means that $n + m = 2k + 1$ for some $k$ (namely, $r + s$), so $n + m$ is odd. Therefore $m$ and $m + n$ have the same parity. ■

This proof is interesting for two reasons. First, notice that in proving that Case 1 is true, we used the result that we have proven previously: if $n$ and $m$ have the same parity, then $n + m$ is even. This means that we didn't have to try writing $n = 2r$ and $m = 2s$, and we ended up saving a lot of space in our proof. Whenever you're writing a proof, feel free to cite any result that you have previously proven. In CS103, it's perfectly

fine to cite proofs from lecture, this book, or the problem sessions, as long as you make it clear what result you're using.

Second, notice that in this proof the cases resulted from the parity of just one of the variables ($m$). We knew that the parity of $n$ must be even, and the only thing that was unclear was whether $m$ was odd or even.

#### 2.2.1.1 A Quick Aside: Choosing Letters

If you'll notice, the proofs that we've done so far use lots of letters to stand for numbers: $m$, $n$, $k$, $r$, $s$, etc. In general, when writing a proof, you should feel free to choose whatever letters you think will make the proof flow most cleanly. However, you should make an effort to pick a consistent naming convention, much in the same way that you should adopt a naming convention when writing computer programs.

In this set of course notes, I will typically use single capital letters ($S$, $T$, $U$) to represent sets. I tend to use the letters $m$, $n$, and $k$ to represent natural numbers and $x$, $y$, $z$ to represent integers. If I run out of letters, I might borrow others from other parts of the alphabet (for example, $r$, $s$, and $t$ for natural numbers) if we exhaust our normal supply.

When working with values in a sequence (more on that later), I'll tend to use subscripted symbols like $x_1$, $x_2$, …, $x_i$. In those cases, the letters $i$, $j$, and $k$ will typically refer to variable indices, and $n$ will represent quantities like the total number of elements in the sequence.

### 2.2.2 Proofs about Sets

In the last chapter, we explored sets and some of the operations on them. You have already seen one theorem about sets (specifically, Cantor's theorem). But what else can we prove about sets? And how do we prove them?

Let's begin with a very simple proof about sets:

> ***Theorem:*** For any sets $A$ and $B$, $A \cap B \subseteq A$.

This theorem intuitively makes sense. We can think of $A \cap B$ as the set of things that $A$ and $B$ have in common. In other words, we're filtering down the elements of $A$ by just considering those elements that also happen to be in $B$. As a result, we should end up with a set that's a subset of the set $A$. So how do we prove this? As you will see, the proof works similarly to our proof about odd and even numbers: it calls back to the definitions of intersection and subset, then proceeds from there.

> ***Proof***: Consider any sets $A$ and $B$. We want to show that $A \cap B \subseteq A$. By the definition of subset, this means that we need to show that for any $x \in A \cap B$, $x \in A$. So consider any $x \in A \cap B$. By the definition of intersection, $x \in A \cap B$ means that $x \in A$ and $x \in B$. Therefore, if $x \in A \cap B$, $x \in A$. Since our choice of $x$ was arbitrary, $A \cap B \subseteq A$. ■

Let's examine the structure of the proof. We initially wanted to prove that $A \cap B \subseteq A$. To do this, we said something to the effect of "okay, I need to prove that $A \cap B \subseteq A$. What does this mean?" By using the definition of subset, we were able to determine that we needed to prove that for any choice of $x \in A \cap B$, it's true that $x \in A$. Again we ask – so what does it mean for $x \in A \cap B$? Again we call back to the definition: $x \in A \cap B$ means that $x \in A$ and $x \in B$. But at this point we're done – we needed to show that any $x \in A \cap B$ also satisfies $x \in A$, but the very definition of $A \cap B$ guarantees this to us!

This proof illustrates a crucial step in many proofs – if you are unsure about how to proceed, try referring to the definitions of the terms involved. Often this simplification will help you make progress toward the ultimate proof by rewriting complex logic in terms of something similar.

Let's do another simple proof:

> ***Theorem:*** For any sets $A$ and $B$, $A \subseteq A \cup B$.

This result says that if we take any collection of things (the set $A$) and combine it together with any other set of things (forming the set $A \cup B$), then the original set is a subset of the resulting set. This seems obvious – after all, if we mix in one set of things with another, that initial set is still present! Of course, it's good to formally establish this, which we do here:

> **Proof:** Consider any sets $A$ and $B$. We want to show that $A \subseteq A \cup B$. To do this, we show that for any $x \in A$, that $x \in A \cup B$ as well. Note that by definition, $x \in A \cup B$ iff $x \in A$ or $x \in B$.
>
> Consider any $x \in A$. It is therefore true that $x \in A$ or $x \in B$, since we know $x \in A$. Consequently, $x \in A \cup B$. Since our choice of $x$ was arbitrary, this shows that any $x \in A$ also satisfies $x \in A \cup B$. Consequently, $A \subseteq A \cup B$, as required. ■

Again, notice the calling back to the definitions. To prove $A \subseteq A \cup B$, we argue that every $x \in A$ also satisfies $x \in A \cup B$. What does it mean for $x \in A \cup B$? Well, the definition of $A \cup B$ is the set of all $x$ such that either $x \in A$ or $x \in B$. From there we can see that we're done – if $x \in A$, then it's also true that $x \in A$ or $x \in B$, so it's true that $x \in A \cup B$.

Let's do another proof, this time proving a slightly more complex result:

> ***Theorem:*** For any sets $A$, $B$, and $C$, we have $C - (A \cap B) \subseteq (C - A) \cup (C - B)$

As an example, let's take $A = \{1, 2, 3\}$, $B = \{3, 4, 5\}$, and $C = \{1, 2, 3, 4, 5\}$. Then we have that

$$C - (A \cap B) = \{1, 2, 3, 4, 5\} - \{3\} = \{1, 2, 4, 5\}.$$

We also have that

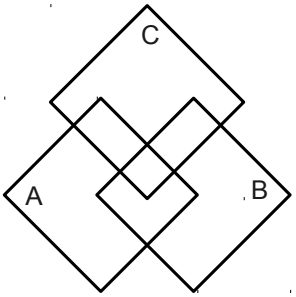
$$(C - A) \cup (C - B) = \{4, 5\} \cup \{1, 2\} = \{1, 2, 4, 5\}.$$

Thus in this single case, $C - (A \cap B) \subseteq (C - A) \cup (C - B)$, since the two sets are equal.

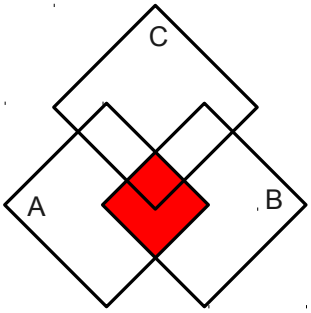
This theorem worked out in the above case, but it's not at all clear exactly why this would be true. How, then, could we prove this?

Whenever you need to write a proof, *always be sure that you understand why what you're proving is true before you try to prove it!* Otherwise, you're bound to get stuck. So before we start to work through the actual proof, let's try to build up an intuition for why this result is true. To do so, let's turn to Venn diagrams, which are surprisingly useful when proving results like these.

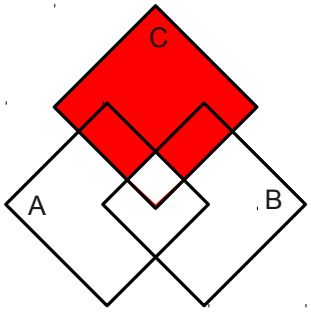
Let's start with this Venn diagram for three sets:*



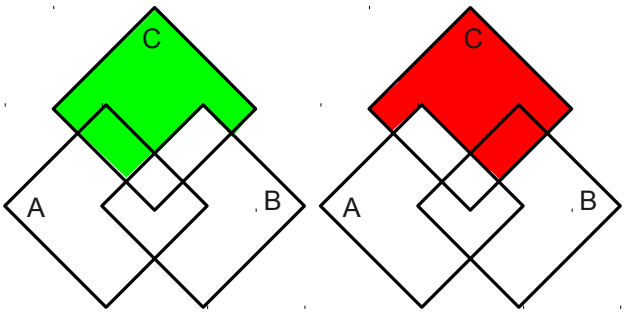

If we highlight the set $A \cap B$, we get this region of the diagram:




Given this, we can see that $C - (A \cap B)$ corresponds to this region:

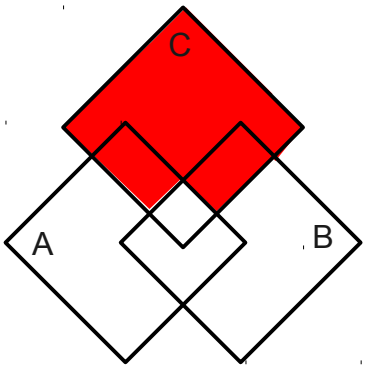



Now, let's take a look at $(C - A) \cup (C - B)$. If we highlight $C - A$ and $C - B$ separately, we get these regions:




* I'm using diamonds instead of circles here because my drawing program (LibreOffice) makes it tricky to fill in circular regions. If you have any suggestions on how to draw better Venn diagrams, please let me know!

Consequently, their union $(C - A) \cup (C - B)$ is this region here:




Now it's a bit clearer why this result should be true – the two sets are actually equal to one another! Moreover, it's easier to see why. To build up the sets $C - (A \cap B)$, we can construct the sets $C - A$ and $C - B$, then combine them together.

That said, the above picture isn't really a mathematical proof in the conventional sense. We still need to write out the proof longhand. To do this, we'll try to translate the above pictorial intuition into words. Specifically, we can work as follows. If we take any element of $C - (A \cap B)$, then (as you can see above) it belongs to at least one of $C - A$ or $C - B$. We can therefore write a proof by cases and show that regardless of which of these two sets our element belongs to, we know that the element must belong to $(C - A) \cup (C - B)$.

This is formalized below:

***Proof:*** Consider any sets $A$, $B$, and $C$. We will show $C - (A \cap B) \subseteq (C - A) \cup (C - B)$. By definition, this is true if for any $x \in C - (A \cap B)$, we also have $x \in (C - A) \cup (C - B)$. So consider any $x \in C - (A \cap B)$. By the definition of set difference, this means that $x \in C$ and $x \notin A \cap B$. Since $x \notin A \cap B$, we know that it is not the case that both $x \in A$ and $x \in B$. Consequently, it must be true that either $x \notin A$ or $x \notin B$. We consider these two cases individually:

*Case 1:* $x \notin A$. Since we know that $x \in C$ and $x \notin A$, we know that $x \in C - A$. By our earlier result, we therefore have that $x \in (C - A) \cup (C - B)$.

*Case 2:* $x \notin B$. Since we know that $x \in C$ and $x \notin B$, we know that $x \in C - B$. By our earlier result, we therefore have that $x \in (C - A) \cup (C - B)$.

In either case we have that $x \in (C - A) \cup (C - B)$. Since our choice of $x$ was arbitrary, we have that $C - (A \cap B) \subseteq (C - A) \cup (C - B)$ as required. ■

Notice that in the course of this proof, we ended up referring back to the proof we did above in which we claimed that for any sets $A$ and $B$, $A \subseteq A \cup B$. Using this theorem, we were able to conclude that if $x \in C - A$, then $x \in (C - A) \cup (C - B)$. This is extremely common in mathematics. We begin with a few simple terms and definitions, then build up progressively more elaborate results from simpler ones. Most major results do not work from first principles, but instead build off of earlier work by combining well-known results and clever insights.

### 2.2.3 Lemmas

Let's think about the simple result that $A \subseteq A \cup B$. In itself, this isn't very surprising. The proof is simple and straightforward, and in the end we don't end up with anything particularly complex. However, as you saw above, this simple result can be used as a building block for proving more elaborate results.

A result that is primarily used as a small piece in a larger proof is sometimes called a **lemma**. Lemmas are distinguished from theorems primarily by how they're used. Some lemmas, such as the pumping lemma (which you'll learn more about later) are actually quite impressive results on their own, but are mostly used as a step in more complex proofs. Other lemmas, like the one you just saw, are simple but necessary as a starting point for future work.

When proving results about sets, lemmas like $A \subseteq A \cup B$ are often useful in simplifying more complex proofs. In fact, many seemingly obvious results about sets are best proven as lemmas so that we can use them later on.

The first lemma that we'll actually treat as such is the following result, which helps us prove that two sets are equal to one another:

> ***Lemma:*** For any sets $A$ and $B$, $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$.

Note the use of the phrase ***if and only if*** in this lemma. The phrase "$P$ if and only if $Q$" means that whenever $P$ is true, $Q$ is true, and whenever $Q$ is true, $P$ is true. In other words, "$P$ if and only if $Q$" means that $P$ and $Q$ have the same truth value – either both $P$ and $Q$ are true, or both $P$ and $Q$ are false. The statement "if and only if" is a very strong assertion – it says that any time we'd like to speak about whether $P$ is true or false, we can instead speak of whether $Q$ is true or false.

As long as we're on the subject, you sometimes see the word ***iff*** used to mean "if and only if." This is a term that we'll use throughout this text, as it's widely used in the mathematical world. Consequently, we might rewrite the above lemma as

> ***Lemma:*** For any sets $A$ and $B$, $A = B$ iff $A \subseteq B$ and $B \subseteq A$.

Note that "iff" is read aloud as "if and only if." That way, we don't need to try to differentiate between "if" and "iff" by listening to how long the speaker draws out the final "f." This means that the above lemma would still be read aloud as "for any sets $A$ and $B$, A equals $B$ if and only if $A$ is a subset of $B$ and $B$ is a subset of $A$."

Now, let's get down to business – what does this lemma say? The above lemma tells us that two sets $A$ and $B$ are equal to one another if and only if (in other words, precisely when) $A$ is a subset of $B$ and vice-versa. Recall that two sets are equal when they have exactly the same elements; it doesn't matter how we describe or construct the sets, just that they have the same elements. The above lemma states that if we want to show that two sets are equal, all we need to do is show that all of the elements of one set are contained in the other and vice-versa.

So how exactly do we go about proving this lemma? So far, all of the proofs that we've seen have taken the form "if $P$, then $Q$." If we want to prove a statement of the form "$P$ iff $Q$," then we need to prove two things – first, if $P$ is true, then $Q$ is true; second, if $Q$ is true, then $P$ is true as well. In other words, both $P$ and $Q$ imply one another.

Given this setup, here is one proof of this result:

***Proof***: We prove both directions of implication.

( $\Rightarrow$ ) First, we show that, for any sets $A$ and $B$, if $A = B$, then $A \subseteq B$ and $B \subseteq A$. If $A = B$, consider any $x \in A$. Since $A = B$, this means that $x \in B$. Since our choice of $x$ was arbitrary, any $x \in A$ satisfies $x \in B$, so $A \subseteq B$. Similarly, consider any $x \in B$, then since $A = B$, $x \in A$ as well. Since our choice of $x$ was arbitrary, any $x \in B$ satisfies $x \in A$, so $B \subseteq A$.

( $\Leftarrow$) Next, we prove that if $A \subseteq B$ and $B \subseteq A$, then $A = B$. Consider any two sets $A$ and $B$ where $A \subseteq B$ and $B \subseteq A$. We need to prove that $A = B$. Since $A \subseteq B$, for any $x \in A$, $x \in B$ as well. Since $B \subseteq A$, for any $x \in B$, $x \in A$ as well. Thus every element of $A$ is in $B$ and vice-versa, so the two sets have the same elements. ■

This proof looks different from the ones we've seen so far both structurally and notationally. This proof is essentially two separate proofs that together prove a larger result; the first half proves that if two sets are equal each is a subset of the other, and the second half proves that if two sets are subsets of one another they are equal. This is because in order to prove the biconditional, we need to prove two independent results, which together combine to prove the biconditional. Within each piece of the proof, notice that the structure is similar to before. We call back to the definitions of subset and set equality in order to reason about how the elements of the sets are related to one another.

You probably noticed the use of ( $\Rightarrow$ ) and ( $\Leftarrow$ ) here. In proofs involving biconditionals, it's common to split the proof up into two logically separate sections, one for the forward direction and one for the reverse direction. To demarcate the sections, we often use the symbols ( $\Rightarrow$ ) and ( $\Leftarrow$ ). The ( $\Rightarrow$ ) part of the proof denotes the forward direction: when proving $A$ iff $B$, this is the "if $A$, then $B$" part. Similarly, the ( $\Leftarrow$ ) part of the proof denotes the "if $B$, then $A$" part. As a courtesy to the proof reader, it usually is a good idea to briefly restate what it is that you're going to be proving before you jump into the proof.

Now that we have this lemma, let's go and use it to prove some Fun and Exciting Facts about set equality! Let's begin with a simple result that teaches something about how symmetric difference works:

***Theorem:*** For any sets $A$ and $B$, $(A \cup B) - (A \cap B) = A \Delta B$.

Intuitively, this says that we can construct the symmetric difference of $A$ and $B$ (that is, the set of elements that are either in $A$ or $B$, but not both) as follows. First, combine the two sets $A$ and $B$ together into the larger set $A \cup B$. Next, take out from that set all of the elements that are in the intersection of $A$ and $B$. The remaining elements form the set $A \Delta B$.

To prove this result, we can use our lemma from above, which says that two sets are equal iff each is a subset of the other. The structure of our proof will thus be as follows – we'll show that each set is a subset of the other, then we'll use the previous lemma to conclude the proof.

Let's begin by showing that $(A \cup B) - (A \cap B) \subseteq A \Delta B$. Since this acts as a stepping stone toward the larger proof, we'll pose it as a lemma.

***Lemma 1***: $(A \cup B) - (A \cap B) \subseteq A \Delta B$.

How might we prove this lemma? To do so, we'll just call back to the definitions of union, intersection, difference, and symmetric difference:

***Proof of Lemma 1:*** We will show that for any $x \in (A \cup B) - (A \cap B)$, $x \in A \Delta B$. So consider any $x \in (A \cup B) - (A \cap B)$. This means that $x \in A \cup B$, but $x \notin A \cap B$. Since $x \in A \cup B$, we know that $x \in A$ or $x \in B$. Since $x \notin A \cap B$, we know that $x$ is not contained in both $A$ and $B$. We thus have that $x$ is in at least one of $A$ and $B$, but not both. Consequently, $x \in A \Delta B$. Since our choice of $x$ was arbitrary, we therefore have that $(A \cup B) - (A \cap B) \subseteq A \Delta B$. ■

The other direction also will be a lemma for the same reasons. Here's the lemma and the proof:

***Lemma 2***: $A \Delta B \subseteq (A \cup B) - (A \cap B)$

This proof is a little bit more involved because there are two completely separate cases to consider when dealing with elements of $A \Delta B$. The proof is below:

***Proof of Lemma 2:*** We will show that for any $x \in A \Delta B$, $x \in (A \cup B) - (A \cap B)$. Consider any $x \in A \Delta B$. Then either $x \in A$ and $x \notin B$, or $x \in B$ and $x \notin A$. We consider these cases separately:

*Case 1:* $x \in A$ and $x \notin B$. Since $x \in A$, $x \in A \cup B$. Since $x \notin B$, $x \notin A \cap B$. Consequently, $x \in (A \cup B) - (A \cap B)$.

*Case 2:* $x \in B$ and $x \notin A$. Since $x \in B$, $x \in A \cup B$. Since $x \notin A$, $x \notin A \cap B$. Consequently, $x \in (A \cup B) - (A \cap B)$.

In either case, $x \in (A \cup B) - (A \cap B)$. Since our choice of $x$ was arbitrary, we have that $A \Delta B \subseteq (A \cup B) - (A \cap B)$. ■

Now that we have these two lemmas, the proof of the general result is surprisingly straightforward:

***Proof of Theorem***: By Lemma 1, $(A \cup B) - (A \cap B) \subseteq A \Delta B$. By Lemma 2, $A \Delta B \subseteq (A \cup B) - (A \cap B)$. Since each set is a subset of the other, by our earlier lemma we have that $(A \cup B) - (A \cap B) = A \Delta B$. ■

That's all that we have to show!

Before we move on to show more applications of the lemma, let's take a minute to examine the proof of Lemma 2. I've reprinted it below:

***Proof of Lemma 2:*** We will show that for any $x \in A \Delta B$, $x \in (A \cup B) - (A \cap B)$. Consider any $x \in A \Delta B$. Then either $x \in A$ and $x \notin B$, or $x \in B$ and $x \notin A$. We consider these cases separately:

*Case 1:* $x \in A$ and $x \notin B$. Since $x \in A$, $x \in A \cup B$. Since $x \notin B$, $x \notin A \cap B$. Consequently, $x \in (A \cup B) - (A \cap B)$.

*Case 2:* $x \in B$ and $x \notin A$. Since $x \in B$, $x \in A \cup B$. Since $x \notin A$, $x \notin A \cap B$. Consequently, $x \in (A \cup B) - (A \cap B)$.

In either case, $x \in (A \cup B) - (A \cap B)$. Since our choice of $x$ was arbitrary, we have that $A \Delta B \subseteq (A \cup B) - (A \cap B)$. ■

Notice the similarity between Case 1 and Case 2. These two cases are virtually identical, except that we've interchanged the role of the sets $A$ and $B$. If you'll notice, there really isn't anything in the above proof to suggest that set $A$ is somehow "more important" than set $B$. If we interchange set $A$ and set $B$, we change the sets $(A \cup B) - (A \cap B)$ and $A \Delta B$ to the sets $(B \cup A) - (B \cap A)$ and $B \Delta A$. But these are exactly the sets we started with! In a sense, because there really isn't an appreciable difference between $A$ and $B$, it seems silly to have two completely difference cases dealing with which sets $x$ is contained in.

This situation – in which multiple parts of a proof end up being surprisingly similar to one another – is fairly common, and mathematicians have invented some shorthand to address it. Mathematicians often write proofs like this one:

***Proof of Lemma 2:*** We will show that for any $x \in A \Delta B$, $x \in (A \cup B) - (A \cap B)$. Consider any $x \in A \Delta B$. Then either $x \in A$ and $x \notin B$, or $x \in B$ and $x \notin A$. Assume without loss of generality that $x \in A$ and $x \notin B$. Since $x \in A$, $x \in A \cup B$. Since $x \notin B$, $x \notin A \cap B$, *so* $x \in (A \cup B) - (A \cap B)$. Since our choice of $x$ was arbitrary, we have that $A \Delta B \subseteq (A \cup B) - (A \cap B)$. ■

Notice the use of the phrase "without loss of generality." This phrase indicates in a proof that there are several different cases that need to be considered, but all of them are identical to one another once we change the names around appropriately. If you are writing a proof where you find multiple cases that seem identical to one another, feel free to use this phrase to write the proof just once. That said, be careful not to claim that you haven't lost generality if the cases are actually different from one another!

As another example of a proof using "without loss of generality," let's consider the following theorem, which has nothing to do with sets:

***Theorem:*** If $m$ and $n$ have opposite parity, $m + n$ is odd.

We can check this pretty easily – $3 + 4 = 7$, which is odd, $137 + 42 = 179$, which is odd, etc. How might we prove this? Well, there are two cases to consider – either $m$ is even and $n$ is odd, or $m$ is odd and $n$ is even. But these two cases are pretty much identical to one another, since $m + n = n + m$ and it doesn't really matter whether it's $m$ or $n$ that's odd. Using this, let's write a quick proof of the above result:

***Proof***: Without loss of generality, assume that $m$ is odd and $n$ is even. Since $m$ is odd, there exists an integer $r$ such that $m = 2r + 1$. Since $n$ is even, there exists an integer $s$ such that $n = 2s$. Then $m + n = 2r + 1 + 2s = 2(r + s) + 1$. Consequently, $m + n$ is odd. ■

This proof is about half as long as it would be otherwise.

## 2.2.4 Proofs with Vacuous Truths

To see if we can get some more mileage out of our lemma about set equality, let's try proving some more results about sets. Let's consider the following result:

***Theorem:*** For any sets $A$ and $B$, if $A \subseteq B$, then $A - B = \varnothing$.

Now, how might we prove this? Right now, the main tool at our disposal for proving two sets are equal is to show that those two sets are subsets of one another. In other words, to prove the above result, we might try proving two lemmas:

***Lemma 1:*** For any sets $A$ and $B$, if $A \subseteq B$, then $\varnothing \subseteq A - B$.
***Lemma 2:*** For any sets $A$ and $B$, if $A \subseteq B$, then $A - B \subseteq \varnothing$.

Okay, let's set out to prove them. Let's begin by trying to prove lemma 1. To do this, we need to show that every element of the empty set is also contained in $A - B$. But wait a minute – this doesn't make any sense, since there aren't any $x \in \varnothing$! But not to worry. If you'll recall from Chapter 1, we introduced the idea of a vacuous truth, a statement that is true because it doesn't apply to anything. Fortunately, that's exactly what we have right here – there aren't any elements of the empty set, so it's vacuously true that every element of the empty set is also contained in $A - B$, regardless of what $A$ and $B$ actually are. After all, it's also true that every element of the empty set is made of fire, that every element of the empty set is your best friend,* etc.

How do we formalize this in a proof? Well, we can just say that it's vacuously true! This is shown here:

***Proof of Lemma 1:*** We need to show that every element $x \in \varnothing$ also satisfies $x \in A - B$. But this is vacuously true, as there are no $x$ satisfying $x \in \varnothing$. ■

Well, that was surprisingly straightforward. On to the second lemma!

At first glance, this statement doesn't seem to make any sense. There are no elements of the empty set, so how could something be a subset of the empty set? This would only happen if there are no elements in the first set, since if there were some element $x \in A - B$, then it would have to be true that $x \in \varnothing$, which we know to be impossible. This actually gives us a hint about how to approach the problem. We know that we shouldn't be able to find any $x \in A - B$, so one route for proving that $A - B \subseteq \varnothing$ is to directly show that the statement "for any $x \in A - B$, $x \in \varnothing$" is vacuously true. This is shown below:

* Saying "every element of the empty set is your best friend" is not the same as saying "the set of your best friends is the empty set." The former is a vacuous truth. The latter is a mathematical insult.

***Proof of Lemma 2:*** We need to show that any $x \in A - B$ also satisfies $x \in \varnothing$. Consider any $x \in A - B$. This means that $x \in A$ and $x \notin B$. Since $A \subseteq B$ and since $x \in A$, we know that $x \in B$. But this means simultaneously that $x \in B$ and $x \notin B$. Consequently, there are no $x \in A - B$, so the claim that any $x \in A - B$ also satisfies $x \in \varnothing$ is vacuously true. ■

Notice the structure of the proof. We begin by using definitions to tease apart what it means for an element to be in $A - B$, then show that, in fact, no elements can be in this set. We conclude, therefore, that the entire lemma must be vacuously true.

We can use these two lemmas to complete the proof:

***Proof of Theorem:*** Consider any sets $A$ and $B$ such that $A \subseteq B$. By Lemma 1, we have that $\varnothing \subseteq A - B$. By Lemma 2, we have that $A - B \subseteq \varnothing$. Thus by our earlier lemma, $A - B = \varnothing$ as required. ■

# 2.3 Indirect Proofs

The proofs that we have done so far have directly shown that a particular statement must be true. We begin with a set of assumptions, then manipulate those assumptions to arrive at a desired conclusion. However, there is an entirely different family of proof techniques called ***indirect proofs*** that indirectly prove that some proposition must be true.

This may seem a bit strange at first, but there are many familiar analogs in real life. For example, suppose that you're biking to class and can't remember whether or not you brought your keys with you. You could directly prove whether you have your keys on you by stopping, getting off your bike, and checking your pockets or purse for your keys. But alternatively, you could use the following line of reasoning. Assuming that you lock your bike (which you should!), you couldn't have unlocked your bike in the first place if you didn't have your keys. Since you definitely unlocked your bike – after all, you're riding it! – you must have your keys with you. You didn't explicitly check to see that you have your keys, but you can be confident that you do indeed have them with you.

In this section, we'll build up two indirect proof techniques – *proof by contradiction*, which shows that a proposition has to be true because it can't be false, and *proof by contrapositive*, which proves that $P$ implies $Q$ by proving that an entirely different connection holds between $P$ and $Q$.

## 2.3.1 Logical Implication

Before we can move on to talk about proofs by contradiction and contrapositive, we need to discuss logical implication. Many of the proofs that we have done so far are proofs of the form

If $P$, then $Q$.

For example, we have proven the following:

If $x$ is even, then $x^2$ is even.

If $m$ is even and $n$ is odd, then $mn$ is even.

If $m$ and $n$ have the same parity, then $m + n$ is even.

If $n$ is even and $m$ is an integer, then $n + m$ has the same parity as $m$.

If $A \subseteq B$, then $A - B = \emptyset$.

In structuring each of these proofs, the general format has been as follows: first, we assume that $P$ is true, then we show that given this assumption $Q$ must be true as well. To understand why this style of proof works in the first place, we need to understand what the statement "If $P$, then $Q$" means. Specifically, the statement "If $P$, then $Q$" means that any time $P$ is true, $Q$ is true as well. For example, consider the statement

If $x \in A$, then $x \in A \cup B$.

This statement says that any time that we find that $x$ is contained in the set $A$, it will also be contained in the set $A \cup B$. If $x \notin A$, this statement doesn't tell us anything. It's still possible for $x \in A \cup B$ to be true, namely if $x \in B$, but we don't have any guarantees.

Let's try this statement:

If I pet the fuzzy kitty, I will be happy.

This tells us that in the scenario where I pet the fuzzy kitty, it's true that I will be happy. This doesn't say anything at all about what happens if I don't pet the kitty. I still might be happy (perhaps I petted a cute puppy, or perhaps Stanford just won another football game).

The general pattern here is that a statement of the form

If $P$, then $Q$.

only provides information if $P$ is true. If $P$ is true, we can immediately conclude that $Q$ must be true. If $P$ is false, $Q$ could be true and could be false. We don't have any extra information.

An important point to note here is that implication deals purely with how the truth or falsity of $P$ and $Q$ are connected, not whether or not there is a causal link between the two. For example, consider this (silly) statement:

If I will it to be true, then $1 + 1 = 2$.

Intuitively, this statement is false: $1 + 1 = 2$ because of the laws of mathematics, not because I consciously wish that it is! But mathematically, the statement is true. If I want $1 + 1 = 2$ to be true, you will indeed find that $1 + 1 = 2$. You'll find $1 + 1 = 2$ regardless of whether or not I want it to be. Consequently, the statement "If I will it to be true, $1 + 1 = 2$" is always true.

Why discuss these (seemingly pedantic) details at all? The reason for this is to make clear what exactly it means for an implication to be true so that we can discuss what it means for an implication to be false. The statement "If $P$, then $Q$" is true if whenever we find that $P$ is true, we also find that $Q$ is true. In order for the statement "If $P$, then $Q$" to be false, we have to find an example where $P$ is true (meaning that we expect $Q$ to be true as well), but to our surprise found that $Q$ actually is false. For example, if we wanted to disprove the claim

If $x + y$ is even, then $x$ is odd.

we would have to find an example where $x + y$ was even, but $x$ was not odd. For example, we can take $x = 2$ and $y = 2$ as a counterexample, since $x + y = 4$, but $x$ is not odd. However, if we were to take something like $x = 3$ and $y = 2$, it would not be a counterexample: $3 + 2$ is not even, so the above claim says nothing about what's supposed to happen.

It's important to make this distinction, because it's surprisingly easy to think that you have disproven an implication that's perfectly true. For example, consider the statement

If $A \subseteq B$, then $A - B = \emptyset$

What happens if we take the sets A = {1, 2} and B = {3}? Then the statement $A \subseteq B$ is false, as is the statement $A - B = \varnothing$. However, we have not contradicted the above statement! The above statement only tells us something about what happens when $A \subseteq B$, and since $A$ isn't a subset of $B$ here, the fact that $A - B \neq \varnothing$ doesn't matter.

## 2.3.2 Proof by Contradiction

One of the most powerful tools in any mathematician's toolbox is proof by contradiction. A proof by contradiction is based on the following logical idea: If a statement cannot possibly be false, then it has to be true.

In a proof by contradiction, we prove some proposition $P$ by doing the following:

1. Assume, hypothetically, that $P$ is **not** true. This is the opposite of what we want to prove, and so we want to show that this assumption couldn't possibly have been correct.
2. Using the assumption that $P$ is false, arrive at a *contradiction* – a statement that is logically impossible.
3. Conclude that, since our logic was good, the only possible mistake we could have made would be in assuming that $P$ is not true. Therefore, $P$ absolutely *must* be true.

Let's see an example of this in action. Earlier, we proved the result that if $n$ is even, then $n^2$ must be even as well. It turns out that the converse of this is true as well:

> ***Theorem:*** If $n^2$ is even, then $n$ is even.

Empirically, this seems to pan out. 36 is even, and $36 = 6^2$, with 6 even. 0 is even, and $0 = 0^2$, with 0 even as well. But how would we actually prove this? It turns out that this is an excellent use case for a proof by contradiction.

To prove this statement by contradiction, let's assume that it's false, which means that the statement "If $n^2$ is even, then $n$ is even" is incorrect. As we just saw, this would have to mean that $n^2$ is even, but $n$ itself is odd. Is this actually possible?

The answer is no – if $n$ were odd, then $n^2$ would have to be odd as well. However, one of the assumptions we made was that $n^2$ is even. This contradiction tells us that something is wrong here. The only thing questionable we did was making the assumption that $n$ is odd with $n^2$ even. Consequently, we know that this combination must be impossible. Therefore, if $n^2$ is even, we know that $n$ is even as well.

We can formalize this in a proof as follows:

> ***Proof:*** By contradiction; assume that $n^2$ is even but that $n$ is odd. Since $n$ is odd, $n = 2k + 1$ for some integer $k$. Therefore $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$. This means that $n^2$ is odd, contradicting the fact that we know that $n^2$ is even. We have reached a contradiction, so our assumption must have been wrong. Therefore, if $n^2$ is even, $n$ must be even. ■

Let's look at this proof in more depth. First, note how it starts off:

> By contradiction; assume that $n^2$ is even but that $n$ is odd.

This sets up how we are going to approach the proof. We state explicitly that we are going to attempt a proof by contradiction. We immediately then say what assumption we are going to make. Here, since we want to contradict the statement "If $n^2$ is even, $n$ is even," we say that the contradiction is that $n^2$ is even, but $n$ is odd.

Once we have set up the proof by contradiction, the remainder of our proof is a quest to show that this assumption has to have been wrong by deriving a contradiction. The middle section of the proof does just that – it arrives at the conclusion that $n^2$ has to be both odd and even at the same time.

Now that we have our contradiction, we can finish the proof by stating that this contradiction means that we're done:

> We have reached a contradiction, so our assumption must have been wrong. Therefore, if $n^2$ is even, $n$ must be even. ■

All proofs by contradiction should end this way. Now that you have the contradiction, explain how it means that the initial assumption was wrong, and from there how this proves the overall result.

Proof by contradiction is a powerful tool. We saw this used in Cantor's theorem in the last chapter (though, admittedly, we haven't seen the formal proof yet), and you will see it used later to prove that several specific important problems cannot be solved by a computer. For now, let's build up some other small examples of how this proof technique can be used.

One interesting application of proofs by contradiction is to show that some particular task cannot be accomplished. Consider the following problem:

> You have 2,718 balls and five bins. Prove that you cannot distribute all of the balls into the bins such that each bin contains an odd number of balls.

This problem seems hard – there are a *lot* of ways to distribute those balls into the bins, though as you'll see there's no way to do it such that every bin has an odd number of balls in it. How might we show that this task is impossible? Using the idea of a proof by contradiction, let's start off by hypothetically assuming that you *can* indeed solve this. Could we then show that this solution leads to some sort of contradiction? Indeed we can. Think of it this way – if we have an odd number of balls in the five bins, then the total number of balls placed into those bins would have to be equal to the sum of five odd numbers. What numbers can you make this way? Well, if we add up two odd numbers, we get an even number (because we know that the sum of two numbers with the same parity is even). If we add up two more of the odd numbers, we get another even number. The sum of those two even numbers is even. If we then add in the last odd number to this even number, we get an odd total number of balls. This is extremely suspicious. We know that the total number of balls has to be odd, because we just proved that it has to. At the same time, we know that there are 2,718 balls distributed total. But this would imply that 2,718 is odd, which it most certainly is not! This is a contradiction, so something we did must have been wrong. Specifically, it has to have been our assumption that we can distribute all of the balls such that each bin has an odd number of balls in it. Therefore, there can't be a solution.

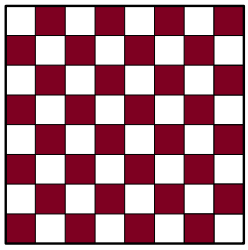
This argument is formalized below as a proof:

***Proof:*** By contradiction; assume that there is a way to distribute all 2,718 balls into five bins such that each bin has an odd number of balls in it. Consider any such way of distributing the balls, and let the number of balls in the five bins be $a$, $b$, $c$, $d$, and $e$. Write the sum $a + b + c + d + e$ as $((a + b) + (c + d)) + e$. Since all five numbers have the same parity, both $(a + b)$ and $(c + d)$ are even. Since $(a + b)$ and $(c + d)$ have the same parity, $((a + b) + (c + d))$ must be even. Then, since $((a + b) + (c + d))$ is even, the sum $((a + b) + (c + d)) + e$ must have the same parity as $e$. Since $e$ is odd, this means that sum of the number of balls in the five bins is odd, contradicting the fact that there are an even number of balls distributed across the bins (2,718). We have reached a contradiction, so our initial assumption must have been wrong and there is no way to distribute 2,718 balls into five bins such that each bin has an odd number of balls. ■

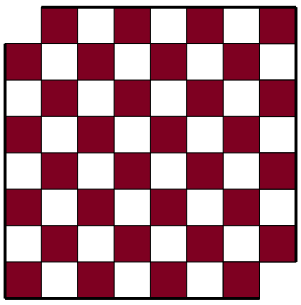
As an aside, I absolutely love this proof. It pulls together our discussion of direct proofs with parities along with proof by contradiction.

Before we move on, though, let's examine the structure of this proof one more time. Note that it has the same shape as the previous proof. We begin by stating that the proof is by contradiction and what that contradiction is. We then derive a contradiction, and conclude by saying that the contradiction proves the original theorem.

Here is yet another example of a classic proof by contradiction. Consider a standard 8 × 8 chessboard:



Now, suppose that we cut off two diagonally opposite corners, as shown here:



Suppose that we want to cover this chessboard with a set of 2 × 1 dominoes. These dominoes can be positioned horizontally or vertically, but never diagonally. Additionally, we cannot stack the dominoes on top of one another. The question is this – is it possible to cover every square on the modified chessboard with dominoes? Interestingly, the answer is no. It's impossible to do so.

So why is that? Well, let's approach this from the perspective of a proof by contradiction. Suppose, hypothetically, that we can cover the chessboard with dominoes. Since each domino covers two horizontally or vertically adjacent squares, we know for a fact that each domino covers exactly one white square and exactly one black square. Moreover, since no two dominoes can stack atop one another, if we add up the total number of white squares covered by each domino and the total number of black squares covered by each

domino, we should get the total number of white and black squares on the chessboard. But this is where we run into trouble. If each domino covers one white square and one black square, then the total number of white squares and black squares covered should have to be the same. Unfortunately, this isn't true. A standard chessboard has the same number of white and black squares. When we removed two opposite corners, we took away two white squares (check the picture above). This means that there are, in fact, two more black squares than white squares, contradicting the fact that we were supposed to have the same number of white squares and black squares. This means (again!) that our assumption was wrong, and that there must be no solution to this puzzle.

Formalized as a proof, the above argument looks like this:

***Theorem:*** There is no way to tile an $8 \times 8$ chessboard missing two opposite corners with dominoes such that each domino is aligned horizontally or vertically and no two dominoes overlap.

***Proof:*** By contradiction; assume that such a tiling exists. Since each domino is aligned horizontally or vertically across two tiles, each domino covers the same number of white and black squares. Since no two dominoes overlap, each square is covered by exactly one domino. Consequently, the number of white squares on the chessboard and the number of black squares on the chessboard should equal the number of dominoes. In turn, this means that the number of white squares and black squares on the chessboard must be equal. But this is impossible – there are 30 white squares and 32 black squares, and $30 \neq 32$. We have reached a contradiction, so our assumption must have been incorrect. Thus there is no solution to the puzzle. ■

## 2.3.3 Rational and Irrational Numbers

In computer science we commonly work with the natural numbers or integers because our computers are digital. However, the real numbers are quite important in mathematics, and it would be a disservice to them if we didn't spend at least a little time exploring their properties.

To begin with, we should make a distinction between two different types of real numbers – the *rational numbers* and the *irrational numbers*. Intuitively, rational numbers are real numbers that can be expressed as the ratio of two integers. For example, any integer is rational, because the integer $x$ is the ratio $x / 1$. Numbers like $^{7}/_{4}$ and $^{137}/_{42}$ are also rational. Formally, we define the rational numbers as follows:

A real number $r$ is called ***rational*** if there exist integers $p$ and $q$ such that $q \neq 0$ and $p / q = r$.

Let's take a minute to see what this says. We're going to say that a number $r$ is rational if there is some way that we can find two integers $p$ and $q$ where $q$ isn't zero and $p / q = r$. We're going to require that $q \neq 0$ so that we can safely use it as the denominator in the fraction. Note that $p$ can be zero, because we'd like $0 = {}^{0}/_{1}$ to count as a rational number.

An important property of this definition is that we say $r$ is rational if there is some way to write $r = p / q$ for integers $p$ and $q$. Note that there can actually be *many* different ways to do this. As an example, we can write $2 = {}^{2}/_{1}$, or $2 = {}^{-2}/_{-1}$, or $2 = {}^{6}/_{3}$, etc. For reasons that will become clearer in a moment, we often use the following property of rational numbers:

Any rational number $r$ can be written as $r = p / q$, where $p$ and $q$ are integers, $q \neq 0$, and $p$ and $q$ have no common factors other than ±1.

This statement essentially states that if $r$ is rational, we can write out $r$ in "simplest form" by writing out $r$ as a fraction that cannot be simplified. For example, when we write $1.5 = {}^{6}/_{4}$, we can simplify the fraction down to $1.5 = {}^{3}/_{2}$ because 6 and 4 have 2 as a common factor. However, we can't simplify ${}^{3}/_{2}$, because the only common factors of 3 and 2 are ±1. Note that we could also write $1.5 = {}^{-3}/_{-2}$ and say that it is in "simplest form" because the only common factors of -3 and -2 are ±1. This is certainly a less "pretty" fraction, though according to the above statement we'll consider it to be in simplest form.

One more definition is in order:

The set $\{ r \mid r \in \mathbb{R}$ and $r$ is rational $\}$, the ***set of all rational numbers***, is denoted $\mathbb{Q}$.

From the definition of $\mathbb{Q}$, it's clear that $\mathbb{Q} \subseteq \mathbb{R}$. However, is it true that $\mathbb{Q} = \mathbb{R}$? That is, is every real number rational? It turns out that the answer to this question is "no." There are many ways to show this using advanced mathematics, but one simple solution is to find an explicit example of an irrational number. It's not all that hard to find an example of an irrational number – numbers like $e$ and $\pi$ are irrational, for example – but to actually prove that these numbers are irrational is surprisingly difficult. Instead, we'll focus on a simple example of a number known to be irrational: $\sqrt{2}$.

Let's go prove the following theorem, which is a beautiful example of a proof by contradiction:

***Theorem:*** $\sqrt{2}$ is irrational.

How exactly can we show this? As you might have guessed from where we are right now, this is a good spot for a proof by contradiction. Let's suppose, for the sake of contradiction, that $\sqrt{2}$ actually is rational. This means that we can find integers $p$ and $q$ such that $q \neq 0$, $p / q = \sqrt{2}$, and $p$ and $q$ have no common factors other than 1 and -1 (that is, they're the "simplest" such $p$ and $q$ that we can use). What to do next? Well, ultimately we're going to try to derive some sort of contradiction. Right now, though, it's not clear what exactly that will be. That's fine, though. Let's just explore around a bit and see if we find anything interesting. If we do, great! We're done. If not, we can back up and try something else.

Let's start off with some simple algebraic manipulations. Since we have that

$$p / q = \sqrt{2}$$

We can square both sides to get

$$p^2 / q^2 = 2$$

If we then multiply both sides by $q^2$, we get

$$p^2 = 2q^2.$$

What does this tell us? For one thing, we know that $p^2$ has to be an even number, since $q^2$ is an integer and $p^2$ is twice $q^2$. But if you'll recall, one of the first proofs we did by contradiction was the proof that if $n^2$ is

even, then $n$ must be even as well. Since $p^2$ is even, this means that $p$ has to be even as well. This tells us that $p = 2k$ for some integer $k$.

We've just shown that if $p / q = \sqrt{2}$, then $p$ has to be even. What can we do with this? Looking above, we've shown that $p^2 = 2q^2$. What happens if we plug in $2k$ in place of $p$? This gives us

$$(2k)^2 = 2q^2$$
$$4k^2 = 2q^2$$
$$2k^2 = q^2$$

This last line tells us that $q^2$ has to be even as well, since it's twice $k^2$ and $k^2$ is an integer. It's at this point that we can see that something unusual is up. Using our previous result, since $q^2$ is even, $q$ has to be even as well. But then both $p$ and $q$ are even, which means that they have to be divisible by two – contradicting the fact that $p$ and $q$ can't have any divisors other than 1 and -1!

In short, our proof worked as follows. Starting with $p / q = \sqrt{2}$, we showed that $p$ had to be even. Since $p$ was even, $q$ had to be even as well, meaning that $p$ and $q$ weren't simplified as far as possible. In fact, there's no possible way for them to be simplified – we've shown that whatever choice of $p$ and $q$ you make, they can always be simplified further. This contradicts Rule 3 of rational numbers, and so $\sqrt{2}$ has to be irrational. This logic is formalized here in this proof:

> ***Proof:*** By contradiction; assume that $\sqrt{2}$ is rational. Then there exist integers $p$ and $q$ such that $q \neq 0$, $p / q = \sqrt{2}$, and $p$ and $q$ have no common divisors other than 1 and -1.
>
> Since $p / q = \sqrt{2}$, this means that $p^2 / q^2 = 2$, which means that $p^2 = 2q^2$. This means that $p^2$ is even, so by our earlier result $p$ must be even as well. Consequently, there exists some integer $k$ such that $p = 2k$.
>
> Since $p = 2k$, we have that $2q^2 = p^2 = (2k)^2 = 4k^2$, so $q^2 = 2k^2$. This means that $q^2$ is even, so by our earlier result $q$ must be even as well. But this is impossible, because it means that $p$ and $q$ have 2 as a common divisor, contradicting the fact that $p$ and $q$ have no common divisors other than 1 and -1.
>
> We have reached a contradiction, so our assumption must have been incorrect. Thus $\sqrt{2}$ is irrational. ■

We now have our first example of a number that we know is not rational. This alone is enough to prove that $\mathbb{Q} \neq \mathbb{R}$. However, is $\sqrt{2}$ the only irrational number? Or are there more irrational numbers like it? It turns out that a great many numbers are irrational; in fact, there are infinitely more irrational numbers than rational numbers! We'll prove this later on in Chapter 6 when we discuss the nature of infinity.

### 2.3.4 Proof by Contrapositive

There is one final indirect proof technique that we will address right now – proof by contrapositive.

To motivate a proof by contrapositive, let's return to our discussion of mathematical implication. Consider the following statement:

If I close the windows, the velociraptors can't get inside.

This statement says that whenever we know that the windows are closed, we know that the velociraptors won't be able to get inside. Now, let's suppose that we know that, unfortunately, the velociraptors did indeed get inside. What could we conclude from this? We know that I certainly didn't close the windows – if I had closed the window, then the raptors wouldn't be inside in the first place!

Let's try another example. Suppose that we know that

$$\text{If } A \subseteq B, \text{ then } A - B = \varnothing.$$

Suppose we find two sets $A$ and $B$ such that $A - B \neq \varnothing$. What can we conclude? Here, we can say that $A$ is not a subset of $B$, because if it were, then $A - B$ would have been equal to $\varnothing$.

There seems to be a pattern here. It seems like if we know that the statement "If $P$, then $Q$" is true and we know that $Q$ is false, then we know that $P$ must be false as well. In fact, that's exactly correct. Intuitively, the rationale is that if $P$ implies $Q$ and $Q$ is false, $P$ couldn't be true, because otherwise $Q$ would be true. Given any implication "If $P$, then $Q$," its **contrapositive** is the statement "If **not** $Q$, then **not** $P$." The contrapositive represents the above idea that if $Q$ is false, $P$ has to be false as well.

It's getting a bit tricky to use phrases like "If $P$, then $Q$" repeatedly through this text, so let's introduce a bit of notation. We will use the notation $P \to Q$ to mean that $P$ implies $Q$; that is, if $P$, then $Q$. Given an implication $P \to Q$, the contrapositive is **not** $Q \to$ **not** $P$.

The contrapositive is immensely useful because of the following result:

> ***Theorem:*** If **not** $Q \to$ **not** $P$, then $P \to Q$.

This theorem is very different from the sorts of proofs that we've done before in that we are proving a result about logic itself! That is, we're proving that if one implication holds, some other implication must hold as well! How might we go about proving this? Right now, we have two techniques at our disposal – we can proceed by a direct proof, or by contradiction. The logic we used above to justify the contrapositive in the first place was reminiscent of a proof by contradiction ("well, if $Q$ is false, then $P$ couldn't be true, since otherwise $Q$ would have been true."). Accordingly, let's try to prove this theorem about the contrapositive by contradiction.

How might we do this? First, let's think about the contradiction of the above statement. Since we are contradicting an implication, we would assume that **not** $Q \to$ **not** $P$, but that $P \to Q$ is false. In turn we would ask: what does it mean for $P \to Q$ to be false? This would only be possible if $P$ was true but $Q$ was not. So at this point, we know the following:

1. **not** $Q \to$ **not** $P$.
2. $P$ is true.
3. $Q$ is false.

And now all of the pieces fall into place. Since $Q$ is false, we know that **not** $Q$ is true. Since **not** $Q$ implies **not** $P$, this means that **not** $P$ is true, which in turn tells us that $P$ should be false. But this contradicts the fact that $P$ is true. We've hit our contradiction, and can conclude, therefore, that if **not** $Q \to$ **not** $P$, then $P \to Q$.

Here is a formal proof of the above:

***Proof:*** By contradiction; assume that **not** $Q \to$ **not** $P$, but that $P \to Q$ is false. Since $P \to Q$ is false, we know that $P$ is true but $Q$ is false. Since $Q$ is false and **not** $Q \to$ **not** $P$, we have that $P$ must be false. But this contradicts the fact that we know that $P$ is true. We have reached a contradiction, so our initial assumption must have been false. Thus if **not** $Q \to$ **not** $P$, then $P \to Q$. ■

This proof has enormous importance for how we can prove implications. If we want to prove that $P \to Q$, we can always instead prove that **not** $Q \to$ **not** $P$. This then implies $P \to Q$ is true.

Let's work through an example of this. Earlier we proved the following result:

***Theorem:*** If $n^2$ is even, then $n$ is even.

Our proof proceeded by contradiction. What if we wanted to prove this result by contrapositive? Well, we want to show that if $n^2$ is even, then $n$ is even. The contrapositive of this statement is that if $n$ is not even, then $n^2$ is not even. More clearly, if $n$ is odd, then $n^2$ is odd. If we can prove that this statement is true, then we will have successfully proven that if $n^2$ is even, then $n$ is even. Such a proof is shown here:

***Proof:*** By contrapositive; we prove that if $n$ is odd, then $n^2$ is odd. Let $n$ be any odd integer. Since $n$ is odd, $n = 2k + 1$ for some integer $k$. Therefore, $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$. Thus $n^2$ is odd. ■

Notice the structure of the proof. As with a proof by contradiction, we begin by announcing that we're going to use a proof by contrapositive. We then state the contrapositive of the statement that we want to prove, both so that readers know what to expect and so that we're clear on what we want to show. From there, we proceed just as we would in a normal proof – we need to show that if $n$ is odd, $n^2$ is odd, and so we assume that $n$ is odd and proceed from there. The result is a remarkably clean and elegant proof.

Here's another example of a proof by contrapositive: suppose that we have 16 objects that we want to distribute into two bins. There are many ways that we might do this – we might split them evenly as an 8/8 split, or might put all of them into one bin to give a 16/0 split, or might have something only a bit lopsided, like a 10/6 split. Interestingly, though, notice that in each case we have at least one bin with at least 8 objects in it. Is this guaranteed to happen? Or is it just a coincidence?

It turns out that this isn't a coincidence, and in fact we can prove the following:

***Theorem:*** If $m + n = 16$, then $m \geq 8$ or $n \geq 8$.

To prove this by contrapositive, we first need to figure out what the contrapositive of the above statement is. Right now, we have the following:

$$m + n = 16 \to m \geq 8 \text{ or } n \geq 8$$

The contrapositive of this statement is

$$\textbf{not } (m \geq 8 \text{ or } n \geq 8) \to \textbf{not } (m + n = 16)$$

Hmmm... that's not very easy to read. Perhaps we can simplify it. Let's start with the right-hand side. We can simplify **not** ($m + n = 16$) to the easier $m + n \neq 16$. This gives

$$\textbf{not } (m \geq 8 \text{ or } n \geq 8) \rightarrow m + n \neq 16$$

But what about the first part? This is a bit more subtle. What is the opposite of $m \geq 8$ or $n \geq 8$? Well, this statement is true if either $m \geq 8$ or $n \geq 8$, so for it to be false we need to ensure that both $m \geq 8$ and $n \geq 8$ are false. This would be true if $m < 8$ and $n < 8$. This gives us the final contrapositive of

$$m < 8 \text{ and } n < 8 \rightarrow m + n \neq 16$$

The important takeaway point from this process is as follows – when determining the contrapositive of a statement, be very careful to make sure that you understand how to negate things properly!

From here, the reason why the initial statement is true should be a bit clearer. Essentially, if both $m$ and $n$ are too small, then their sum can't be 16. This is formalized below:

***Proof:*** By contrapositive; we show that if $m < 8$ and $n < 8$, then $m + n \neq 16$. To see this, note that

$$\begin{aligned} m + n &< 8 + n \\ &< 8 + 8 \\ &= 16 \end{aligned}$$

So $m + n < 16$. Consequently, $m + n \neq 16$. ■

# 2.4 Writing Elegant Proofs

We've now seen three approaches to writing proofs: direct proofs, which follow a logical train of thought to show a conclusion; proof by contradiction, which assumes the opposite of what it wants to prove and shows that this leads to impossible conclusions; and proof by contrapositive, which proves that $P$ implies $Q$ by proving that not $Q$ implies not $P$. We will use these proof techniques throughout the rest of our exploration of the mathematical foundations of computing.

Although we've described the technical details of how to write proofs of this sort, we have not talked about how to write *elegant* proofs in these styles. Proofs are like essays – you can express the same ideas in many different ways, and depending on how you do it you can make it easier or harder for your reader to understand what you're doing. This final part of the chapter explores techniques for writing good proofs and how to determine that your proofs are correct.

## 2.4.1 Treat Proofs as Essays

Proofs are rigorous arguments intended to prove some point. The point of writing a proof is to convey a mathematically rigorous argument. Your goal when writing a proof is not just to write down the argument, but to help someone unfamiliar with the result see why it must be true. Well-written proofs make it easy for others to understand your argument, while poorly-written proofs – even ones that have sound reasoning – can actually *prevent* others from following your reasoning.

Approach writing proofs as you would writing essays. Describe what you're going to talk about before you jump right into it. Give guideposts about where the proof is going so that a reader can tell what you are planning on doing. If you have a long, complicated thought, break it down into smaller pieces (i.e. lemmas) and explain each one individually.

As an example, consider the following proof:

> ***Theorem:*** If $n$ is an even integer, then $n^2$ is an even integer.

> ***Proof:*** $(2k)^2 = 2(2k^2)$ for any integer $k$. If we have a particular $2k^2$, we can write $2k^2 = r$ for some integer $r$. If $n$ is even, then $n = 2k$ for some integer $k$. Therefore, $n^2 = 2r$ for some integer $r$. ■

Make sense? Didn't think so. All of the right pieces of the proof are here, but they're so jumbled and disorganized that it's almost impossible to determine what's being said here.

There are lots of things that are confusing about this proof: statements are introduced with no context or justification, the flow isn't clear at all, and the ultimate conclusion only makes sense if you skip around randomly. Does that mean the logic isn't a valid? In this case, no; the logic is perfectly fine. However, the *argument* is extremely weak because it requires an intense effort on the part of the reader to see why it works at all.

The standard proof of this result, which we saw earlier, more clearly lays out the steps in sequence. First, we start by writing $n = 2k$ for some integer $k$, because that's one of the few things we're assuming. Since we want to talk about $n^2$, it's reasonable to square both sides to get $n^2 = (2k)^2$. We want to write $n^2$ as $2r$ for some choice of $r$, so we can simplify the math by noting $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$. And then we're done, because $2k^2$ is an integer and $n^2$ is twice that integer. If we were to turn this line of reasoning into a proof, it would be significantly easier to follow because it follows a reasonable narrative – at each point, it makes sense to try the step we're about to try.

Logical flow is the main reason we start all our proofs by contradiction or contrapositive by saying something to the effect of "we proceed by contradiction" or "by contrapositive;" by doing so, we provide context for what will follow so that the reader (often, you!) can tell where we're going.

## 2.4.2 Avoid Shorthand or Unnecessary Symbols

A lot of math involves working with symbols; at this point, we've seen all of the following symbols (and a few more) used to denote specific concepts or quantities:

$$\cup \; - \; \cap \; \Delta \; \wp \; \varnothing \; \mathbb{N} \; \mathbb{R} \; \mathbb{Z} \; \mathbb{Q} \; \blacksquare \; \rightarrow \; \subseteq$$

Although mathematics relies on using precise symbols to denote precise concepts, that doesn't mean that a proof should consist purely of symbols. Rather, a proof should be a piece of writing that conveys how different concepts relate to one another and how the truth of the overall theorem follows from those relations. When you need to use symbols to convey an object or concept, such as the fact that $A$ is a subset of $B$, it's perfectly appropriate to write out $A \subseteq B$ in symbolic notation, since the notation is specifically designed to convey this. Similarly, if you want to talk about the set of all elements in at least one of $A$ and $B$, it's best to just write $A \cup B$. This is appropriate because ultimately you are trying to prove a mathematical result that itself is represented in mathematical notation.

On the other hand, you should try to avoid using mathematical notation to describe the *structure* of the argument. For example, you may have seen the symbols $\therefore$ and $\because$ to mean "therefore" and "because." These symbols are commonly-used shorthands that are great for taking notes or writing something up on a whiteboard. However, they can make proofs extremely dense. For example, consider the following proof, which is technically correct but almost impossible to read.

> ***Theorem:*** If $n$ is an even integer, then $n^2$ is even.

> ***Proof:*** $\because n$ even, $\exists k \in \mathbb{Z}$ s.t. $n = 2k$. $\therefore n^2 = (2k)^2 = 2(2k^2)$. $n^2 = 2(2k^2) \Rightarrow \exists r \in \mathbb{Z}$ s.t. $n^2 = 2r$ ($\because r = 2k^2$). $\therefore n^2$ even. ■

If you haven't seen some of the symbols or shorthands in this proof before, you can appreciate just how inscrutable this proof is. (In case it helps: $\exists$ means "there exists," the $\Rightarrow$ symbol means "implies," and s.t. is shorthand for "such that.") If you have seen these symbols, hopefully your initial response is "wow, that's really, really dense."

Perhaps the best way to summarize the problem with this proof is that it's not *inviting*. This proof isn't trying to help you understand what's going on; it's trying to save on space. You can think of it as "dehydrated proof:" all the essential parts are there, but you're going to have to add water and stir for a long time before you can rehydrate it back into an argument like this one:

> ***Theorem:*** If $n$ is an even integer, then $n^2$ is even.

> *Proof:* Because $n$ is even, there is some integer $k$ such that $n = 2k$. Therefore, $n^2 = (2k)^2 = 2(2k^2)$. Consequently, there is an integer $r$ (namely, $2k^2$) such that $n^2 = 2r$. Therefore, $n^2$ is even. ■

The argument has the same structure as the one given above, but it's much, much easier to read.

### 2.4.3 Write Multiple Drafts

Building off our proofs-are-essays metaphor, it's extremely valuable to sketch out a draft of a proof before writing up your final version. If you start a proof without a clear sense of where you're going with it, you will probably end up with a very convoluted line of reasoning that goes down a lot of unnecessary paths before arriving at the result. Consequently, we recommend writing out a draft of the proof, looking over it critically, then rewriting it to be as clean as possible (and, possibly, repeating this process.)

When you have a first draft of a proof, we suggest tracing through the line of reasoning and checking it for correctness (we'll discuss this in more detail a bit later on). If you're sure the proof is correct, we then recommend taking a look at what you've written and seeing how much of it is actually necessary to show the overall result. Did you prove something you thought would be useful but didn't actually need? Leave it out of the next draft! You can often shrink proofs significantly by using this approach, and the resulting proof will be a lot more focused, easier to read, and less likely to contain extraneous (and potentially incorrect!) details.

### 2.4.4 Avoid "Clearly" and "Obviously"

When you're writing a proof, you are trying to show a logical train of thought that establishes a result. Key to doing so is the fact that you can back up all of the assertions you're making along the way. Usually, this means manipulating definitions or combining together steps you took earlier in the proof.

When writing proofs, it is extremely common to arrive at a point where you need to show a result that just feels *obvious*, only to find that it's surprisingly hard to show. When this happens, it can be tempting to justify the result by writing "clearly, $X$ is true" or "$X$ is obviously true." *Resist the temptation to do this!* When someone is reading over your proof, words like "clearly" and "obviously" come across as belittling. The reader might look at it and say "wow, apparently it's *clearly* true, but I guess I'm not smart enough to see why..." or "something must be wrong with me, because *I* don't see this as obvious..." Given that the main reason why you might want to mark something as "clearly" or "obviously" true is that it just *seems* true even though you can't prove it, this sort of writing can often be perceived as insulting or condescending.

A good test for whether something is "clearly" or "obviously" true is the following: if you want to write something to the effect of "clearly, $X$ is true," grab a pen or pencil and try to write out a proof of $X$ on a sheet of paper. If you can immediately sketch out a proof, then *write that proof* instead of claiming that $X$ is obvious so that others can follow how you know $X$ is true. If you can't, then probably $X$ isn't as obvious as you might think it is, and you should take the time to work through the proof of $X$!

# 2.5 Chapter Summary

- A mathematical proof is a series of logical steps starting from a basic set of assumptions and arriving at a conclusion. Assuming the assumptions are valid and the logic is sound, the result is incontrovertibly true.
- Proofs often involve *lemmas*, smaller proofs of intermediate results which then build into the overall proof.
- Proofs often involve *cases*, branches in the proof that cover different possibilities.
- The *parity* of an integer is whether it is even or odd. Parity interacts in interesting ways with addition and multiplication.
- Two sets are equal if and only if each set is a subset of the other.
- Logical implications are statements of the form "If $P$, then $Q$." We denote this $P \to Q$. Such a statement means that whenever $P$ is true, $Q$ must be true as well, but say nothing about causality or correlation.
- To disprove an implication, one finds a way for $P$ to be true and $Q$ to be false.
- A *proof by contradiction* works by assuming the opposite of what is to be shown, then deriving a *contradiction*, a logically impossible statement.
- A number is called *rational* if it is the ratio of two integers, the second of which is not zero, which share no common factors other than $\pm 1$.
- The *contrapositive* of the implication "If $P$, then $Q$" is the statement "If not $Q$, then not $P$." A statement is logically equivalent to its contrapositive.
- A *proof by contrapositive* proves an implication by proving its contrapositive instead.

# 2.6 Chapter Exercises

1. Let's define the function $max(x, y)$ as follows: if $x < y$, then $max(x, y) = y$; else, $max(x, y) = x$. For example, $max(1, 3) = 3$, $max(2, 2) = 2$, and $max(-\pi, 137) = 137$. Prove that the following holds for any $x$, $y$, and $z$: $max(x, max(y, z)) = max(max(x, y), z)$.
2. Let's define the absolute value function $|x|$ as follows: if $x < 0$, then $|x| = -x$; otherwise, $|x| = x$. Prove that $|xy| = |x||y|$.
3. Prove that $mn$ is odd iff $m$ is odd and $n$ is odd.
4. Prove that if $n$ is an integer and $n$ is a multiple of three (i.e. $n = 3k$ for some integer $k$), then $n^2$ is a multiple of three.
5. A number $n$ called *congruent to one modulo three* iff $n = 3k + 1$ for some integer $k$ and is called *congruent to two modulo three* iff $n = 3k + 2$ for some integer $k$. Every integer is either a multiple of three, congruent to one modulo three, or congruent to two modulo three. Prove that if $n$ is an integer and $n^2$ is a multiple of three, then $n$ is a multiple of three.
6. Prove that $\sqrt{3}$ is irrational.
7. A triple of positive natural numbers $(a, b, c)$ is called a ***Pythagorean triple*** if there is a right triangle whose sides have length $a$, $b$, and $c$. Formally, $a^2 + b^2 = c^2$. Some examples of Pythagorean triples include (3, 4, 5), (5, 12, 13), and (7, 24, 25). Prove that if $(a, b, c)$ is a Pythagorean triple, then at least one of $a$, $b$, or $c$ must be even.
8. Prove that if $(a, b, c)$ is a Pythagorean triple, then $(a + 1, b + 1, c + 1)$ is *not* a Pythagorean triple.
9. Prove that $(a, a, b)$ is *never* a Pythagorean triple.
10. A natural number $n$ is called a multiple of four iff there is some $k \in \mathbb{N}$ such that $n = 4k$. For every natural number $n$, exactly one of $n$, $n + 1$, $n + 2$, or $n + 3$ is a multiple of four. Prove that for any natural number $n$, that either $n^2$ or $n^2 + 3$ is a multiple of four.
11. According to the World Bank, the population of Canada in 2011 was 34,482,779.* Prove that there are no natural numbers $m$ and $n$ such that $m^2 + n^2 = 34{,}482{,}779$.
12. Prove or disprove: if $r$ is rational and $s$ is irrational, then $r + s$ is irrational.
13. Prove or disprove: $r$ is rational iff $-r$ is rational.
14. Prove or disprove: if $r$ is irrational and $s$ is irrational, then $r + s$ is irrational. ★
15. Prove or disprove: if $r$ is irrational and $s$ is irrational, then $r^s$ is irrational. ★
16. Suppose you are having dinner with nine friends and want to split the bill, which is $44. Everyone pays in dollar bills. Prove that at least two people in your group paid the same amount of money.
17. Suppose that $A$, $B$, and $C$ are sets. Prove that $(C - B) - A = C - (B \cup A)$.
18. Prove that for any sets $A$, $B$, and $C$, that $(A \Delta B) \Delta C = A \Delta (B \Delta C)$. This shows that symmetric difference is ***associative***.
19. Prove that $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.
20. Prove that $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.
21. Prove or disprove: If $A = B$, then $\wp(A) = \wp(B)$.

* Source: http://www.google.com/publicdata/explore?ds=d5bncppjof8f9_&met_y=sp_pop_totl&idim=country:CAN&dl=en&hl=en&q=population+of+canada, as of September 30, 2012.

22. Prove or disprove: If $\wp(A) = \wp(B)$, then $A = B$.

23. Prove that if $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$. This shows that $\subseteq$ is *transitive*.

24. A *right triomino* is an L-shaped tile made of three squares. Prove that it is impossible to tile an $8 \times 8$ chessboard missing two opposite corners using right triominoes.

25. Is it *ever* possible to tile an $n \times n$ chessboard missing two opposite corners with right triominoes, assuming $n \geq 3$? If so, find an $n$ for which it's possible. If not, prove that it's always impossible for any $n \geq 3$.

26. Suppose that you have twenty-five balls to place into five different bins. Eleven of the balls are red, while the other fourteen are blue. Prove that no matter how the balls are placed into the bins, there must be at least one bin containing at least three red balls.

27. Suppose that you have a 3" × 3" × 3" cube of cheese (or 3cm × 3cm × 3cm if you live in a place that uses a sane, rational system of measurements) that's subdivided into 27 1" × 1" × 1" smaller cubes of cheese. A mouse wants to eat the cube of cheese and does so as follows: she first picks any cube to eat first, then moves to an adjacent cube of cheese (i.e. a cube that shared a face with the cube that was just eaten) to eat next. Is it possible for the mouse to eat the center cube of cheese last? If so, show how. If not, prove it's impossible.[*] ★

28. Consider the quadratic question $ax^2 + bx + c = 0$, where $a$, $b$, and $c$ are integers. Prove that if $a$, $b$, and $c$ are odd, then $ax^2 + bx + c = 0$ has no rational roots (that is, there are no rational values of $x$ for which $ax^2 + bx + c = 0$). As a hint, proceed by contradiction; assume that $x = p / q$ for some $p$ and $q$, then think about the parities of $p$ and $q$. ★

29. A *Latin square* is an $n \times n$ grid filled with the natural numbers 1 through $n$ such that every row and every column contains each number exactly once. A *symmetric Latin square* is one where the square is symmetric across the main diagonal; that is, the number at position $(i, j)$ is equal to the number at position $(j, i)$.

    Prove that in every symmetric Latin square, every natural number between 1 and $n$ appears exactly once on the main diagonal.

* Adapted from Problem 4E of *A Course in Combinatorics*, *Second Edition* by Lint and Wilson.

# Chapter 3 Mathematical Induction

In the previous chapter, we saw how to prove statements that are true for all objects of some type – all natural numbers, all real numbers, all chessboards, etc. So far, you have three techniques at your disposal: direct proof, proof by contradiction, and proof by contrapositive.

Suppose that we restrict ourselves to proving facts about the natural numbers. The natural numbers have many nice properties – no two adjacent natural numbers have any values between them, every natural number is even or odd, etc. – which makes it possible to prove things about the natural numbers using techniques that do not apply to other structures like the real numbers, pairs of natural numbers, etc.

This chapter explores proof by induction, a powerful proof technique that can be used to prove various results about natural numbers and discrete structures. We will use induction to prove certain properties about the natural numbers, to reason about the correctness of algorithms, to prove results about games, and (later on) to reason about formal models of computation.

## 3.1 The Principle of Mathematical Induction

The principle of mathematical induction is defined as follows:

> ***(The principle of mathematical induction)*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:
>
> $P(0)$ is true
> For any $n \in \mathbb{N}$, $P(n) \to P(n + 1)$
>
> Then for any $n \in \mathbb{N}$, $P(n)$ is true.

Let's take a minute to see exactly what this says. Suppose that we have some property $P(n)$, perhaps $P(n)$ is "$n$ is either even or odd", or $P(n)$ is "the sum of the first $n$ odd numbers is $n^2$." We know two things about $P(n)$. First, we know that $P(0)$ is true, meaning that the property is true when applied to zero. Second, we know that if we ever find that $P(n)$ is true, we will also find that $P(n + 1)$ is true. Well, what would that mean about $P(n)$? Since we know that $P(0)$ is true, we know that $P(1)$ must be true. Since $P(1)$ must be true, we know that $P(2)$ must be true as well. From $P(2)$ we get $P(3)$, and from $P(3)$ we get $P(4)$, etc. In fact, it seems like we should be able to prove that $P(n)$ is true for arbitrary $n$ by using the fact that $P(0)$ is true and then showing $P(0)$, $P(1)$, $P(2)$, …, $P(n – 1)$, $P(n)$.

The principle of mathematical induction says that indeed we can conclude this. If we find some property that starts true ($P(0)$ holds) and continues to be true when started ($P(n) \to P(n + 1)$), then we can conclude that indeed $P(n)$ will be true for all natural numbers $n$.

Induction is very different from the other proof techniques we have seen before. It gives us a way to show that some property is true for all natural numbers $n$ not by directly showing that it must be true, but instead by showing that we could incrementally build up the result one piece at a time.

We can find all sorts of examples of induction in the real world. Before we start working through formal proofs by induction, let's see if we can build up an intuition for how induction works.

As a simple example, consider climbing up a flight of stairs. How exactly do you get to the top? Well, we know that you can climb up zero steps, since you can just stand at the base of the stairs and not go anywhere. Moreover, we know that if you're able to climb zero steps, you should also be able to climb one step by climbing zero steps and then taking one step up. We also know that you can climb two steps, since you can get up to the first step and then take one step to the second step. If you can get to the second step, you can get to the third step by just taking one more step. Repeating this process, we can show that you can get to the top of any staircase.

We could think about this inductively as follows. Let $P(n)$ be "you can climb to the top of $n$ stairs." We know that $P(0)$ is true, because you can always climb to the top of zero stairs by just not moving. Furthermore, if you can climb to the top of $n$ steps, you can climb to the top of $n + 1$ steps by just taking one more step. In other words, for any $n \in \mathbb{N}$, $P(n)$ implies $P(n + 1)$. Using the principle of mathematical induction, you could conclude that you can climb a staircase of any height.

### 3.1.1 The Flipping Glasses Puzzle

Consider the following puzzle: you are given five wine glasses, as shown here:

You want to turn all of the wine glasses upside-down, but in doing so are subject to the restriction that you always flip two wine glasses at a time. For example, you could start off by flipping the first and last glasses, as shown here:

And could then flip the second and fourth glasses, like this:

From here, we might flip the first and third glasses to get this setup:

If you play around with this puzzle, though, you'll notice that it's tricky to get all of the wine glasses flipped over. In fact, try as you might, you'll never be able to turn all of the wine glasses over if you play by these rules. Why is that? Figuring the answer out requires a bit of creativity. Let's count how many wine glasses are facing up at each step. Initially, we have five wine glasses facing up. After our first step, we flip two of the wine glasses, so there are now three wine glasses facing up. At the second step, we have several options:

1. Flip over two glasses, both of which are facing up,
2. Flip over two glasses, both of which are facing down, or
3. Flip over two glasses, one of which is facing up and one of which is facing down.

How many wine glasses will be facing up after this step? In the first case, we decrease the number of wine glasses facing up by two, which takes us down to one glass facing up. In the second case, we increase the number of wine glasses facing up by two, which takes us to five glasses facing up. In the third case, the net change in the number of wine glasses facing up is zero, and we're left with three glasses facing up.

At this point we can make a general observation – at any point in time, each move can only change the number of up-facing wine glasses by +2, 0, or -2. Since we start off with five wine glasses facing up, this means that the number of wine glasses facing up will always be exactly 1, 3, or 5 – all the odd numbers between 0 and 5, inclusive. To solve the puzzle, we need to get all of the wine glasses facing down, which means we need zero wine glasses facing up. But this means that the puzzle has to be impossible, since at any point in time the number of upward-facing wine glasses is going to be odd.

The question now is how we can formalize this as a proof. Our argument is the following:

- The number of upward-facing wine glasses starts off odd.
- At any point, if the number of upward-facing wine glasses is odd, then after the next move the number of upward-facing wine glasses will be odd as well.

The argument here is inherently inductive. We want to prove that the number of glasses starts odd, and that if it starts odd initially it will stay odd forever. There are many ways to formalize the argument, but one idea would be to prove the following for all natural numbers $n$:

> ***Lemma:*** For any natural number $n$, after $n$ moves have been made, the number of upward-facing glasses is an odd number.

The phrasing here says that no matter how many moves we make (say, $n$ of them), the number of upward-facing glasses will be odd. Given this lemma, it's extremely easy to prove that the puzzle is unsolvable.

So how exactly do we prove the lemma? In a proof by induction, we need to do the following:

1. Define some property $P(n)$ that we want to show is true for all natural numbers $n$.
2. Show that $P(0)$ is true.
3. Show that for any natural number $n$, if $P(n)$ is true, then $P(n + 1)$ is true as well.

Let's walk through each of these steps in detail. First, we'll need to come up with our property $P(n)$. Here, we can choose something like this:

$$P(n) \equiv \text{“After } n \text{ steps, there are an odd number of upward-facing glasses.”}$$

Notice that our choice of $P(n)$ only asserts that there are an odd number of upward-facing glasses for some specific $n$. That is, $P(4)$ just says that after four steps, there are an odd number of upward-facing glasses, and $P(103)$ just says that after 103 steps, there are an odd number of upward-facing glasses. This is perfectly normal in an induction proof. Mathematical induction lets us define properties like this one, then show that the property is true for all choices of natural numbers $n$. In other words, even though we want to prove that the claim is true for all natural numbers $n$, our property only says that the claim must be true for some specific choice of $n$.

Now that we have our property, we need to prove that $P(0)$ is true. In this case, that means that we have to show that after 0 steps, there are an odd number of upward-facing glasses. This is true almost automatically – we know that there are five upward-facing glasses to begin with, and if we make 0 steps we can't possibly change anything. Thus there would have to be five upward-facing glasses at the end of this step, and five is odd.

It seems almost silly that we would have to make this argument at all, but it's crucial in an inductive proof. Remember that induction works by showing $P(0)$, then using $P(0)$ to get $P(1)$, then using $P(1)$ to get $P(2)$, etc. If we don't show that $P(0)$ is true, then this entire line of reasoning breaks down! Because the entire inductive proof hinges on $P(0)$, $P(0)$ is sometimes called the **inductive basis** or the **base case**.

When writing inductive proofs, you'll often find that $P(0)$ is so trivial that it's almost comical. This is perfectly normal, and is confirmation that your property is not obviously incorrect. Always make sure to prove $P(0)$ in an inductive proof.

The last step in an induction is to show that for any choice of $n \in \mathbb{N}$, that if $P(n)$ is true, $P(n + 1)$ must be true as well. Notice the structure of what we need to show – for any choice of $n$, we must show that $P(n)$ implies $P(n + 1)$. As you saw last chapter, to prove something like this, we'll choose some arbitrary natural number $n$, then prove that $P(n) \to P(n + 1)$. Since our choice of $n$ is arbitrary, this will let us conclude that $P(n) \to P(n + 1)$ for any choice of $n$. In turn, how do we then show that $P(n) \to P(n + 1)$? This statement is an implication, so as we saw last chapter, one option is to assume that $P(n)$ is true, then to prove $P(n + 1)$. This step of the proof is called the **inductive step**, and the assumption we're making, namely that $P(n)$ is true, is called the **inductive hypothesis**.

If you think about what we're saying here, it seems like we're assuming that for any $n$, $P(n)$ is true. This is not the case! Instead, what we are doing is supposing, hypothetically, that $P(n)$ is true for one specific natural number $n$. Using this fact, we'll then go to show that $P(n + 1)$ is true as well. Since the statements $P(n)$ and $P(n + 1)$ are not the same thing, this logic isn't circular.

So we now have the structure of what we want to do. Let's assume that for some arbitrary natural number $n \in \mathbb{N}$, that $P(n)$ is true. This means that after $n$ steps, the number of upward-facing glasses is odd. We want to show that $P(n + 1)$ is true, which means that after $n + 1$ steps, the number of upward-facing glasses is odd. How would we show this? Well, we're beginning with the assumption that after $n$ steps there are an odd number of upward-facing glasses. Let's call this number $2k + 1$. We want to assert something about what happens after $n + 1$ steps, so let's think about what that $(n + 1)^{\text{st}}$ step is. As mentioned above, there are three cases:

- We flip two upward-facing glasses down, so there are now $2k + 1 - 2 = 2(k - 1) + 1$ upward-facing glasses, which is an odd number.
- We flip two downward-facing glasses up, so there are now $2k + 1 + 2 = 2(k + 1) + 1$ upward-facing glasses, which is an odd number.
- We flip one upward-facing glass down and one downward-facing glass up, which leaves the total at $2k + 1$ upward-facing glasses, which is also odd.

So in every case, if after $n$ steps the number of upward-facing glasses is odd, then after $n + 1$ steps the number of upward-facing glasses is odd as well. This statement, combined with our proof of $P(0)$ from before, lets us conclude by mathematical induction that after *any* number of steps, the number of upward-facing glasses is odd.

This line of reasoning can be adapted into a short and elegant formal proof by induction, which is shown here:

***Lemma:*** For any natural number $n$, after $n$ moves have been made, the number of upward-facing glasses is an odd number.

***Proof:*** By induction. Let $P(n)$ be "after $n$ moves have been made, the number of upward-facing glasses is an odd number." We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction, from which the lemma follows.

As our base case, we will prove $P(0)$, that after 0 moves have been made, the number of upward-facing glasses is an odd number. After zero moves are made, the glasses are still in their initial configuration. Since we begin with five upward-facing glasses, this means that after 0 moves, the number of upward-facing glasses is five, which is odd.

For our inductive step, assume that for some $n \in \mathbb{N}$ that $P(n)$ is true and that after $n$ moves have been made, the number of upward-facing glasses is odd. We will prove $P(n + 1)$, that after $n + 1$ moves have been made, the number of upward-facing glasses is odd. Any sequence of $n + 1$ moves consists of a sequence of $n$ moves followed by any single move. So consider any sequence of $n$ moves. By our inductive hypothesis, after these $n$ moves are made, the number of upward-facing glasses is odd; let the number of upward-facing glasses be $2k + 1$ for some $k \in \mathbb{Z}$. Consider the $(n + 1)^{st}$ move. This flips two glasses, and there are three cases to consider:

*Case 1:* We flip two upward-facing glasses down. This means there are now $2k + 1 - 2 = 2(k - 1) + 1$ upward-facing glasses, which is an odd number.

*Case 2:* We flip two downward-facing glasses up. This means there are now $2k + 1 + 2 = 2(k + 1) + 1$ upward-facing glasses, which is an odd number.

*Case 3:* We flip one downward-facing glass up and one upward-facing glass down. This means there

are still $2k + 1$ upward-facing glasses, which is an odd number.

Thus in each case, the number of upward-facing glasses after $n + 1$ steps is an odd number, so $P(n + 1)$ holds. This completes the induction. ■

Take a minute to notice the structure of this proof. As with a proof by contradiction or contrapositive, we begin by announcing that the proof will be by induction. We then define our choice of property $P(n)$ that we will prove correct by induction. Next, we announce that we are going to prove $P(0)$, state what $P(0)$ is, then prove $P(0)$ is true. Having done this, we then announce that we're going to assume that $P(n)$ is true for some choice of natural number $n$, and mention what this assumption means. We then state that we're going to prove $P(n)$ and what specifically this means that we're going to show. We then use the assumption of $P(n)$ as a starting point to prove $P(n + 1)$, and proceed as in a normal proof. Finally, we conclude the proof by noting that we've done a legal induction. Although the types of proofs you can do by induction vary greatly (as you'll see in this chapter), the basic structure of an induction proof will almost always follow this general template.

Given this lemma, we can formally prove that the flipping glasses puzzle is unsolvable:

***Theorem:*** The flipping glasses puzzle has no solution.

***Proof:*** By contradiction; suppose there is a solution. If this solution has $k$ steps, then after the $k$th step, all the glasses must be facing down. By our previous lemma, we know that an odd number of glasses must be facing up. But this is impossible, since if all five glasses are facing down, then zero are facing up, and zero is even. We have reached a contradiction, so our assumption must have been wrong. Thus there is no solution to the puzzle. ■

# 3.2 Summations

One of the most common applications of induction is in simplifying summations. Summations arise frequently in computer science when analyzing the growth rates of certain algorithms, in combinatorics when determining how many objects there are of certain sizes, etc.

As an example, consider the **selection sort** algorithm, an algorithm for sorting a list of values into ascending order. The algorithm works based on the following observation. If we remove the smallest element from the list, we know that in the sorted ordering of the list it would appear at the front. Consequently, we can just move that element to the front of the list, then sort what remains. We then move the smallest of the remaining elements to the second-smallest position, the smallest of what remains after that to the third-smallest position, etc. As an example, suppose that we want to sort this list of values:

4 1 0 3 2

We begin by removing the zero and putting it in the front of the list:

0 4 1 3 2

Now, we sort what remains. To do this, we find the smallest element of what's left (the 1), remove it from the list, and place it after the 0:

0 1 4 3 2

Repeating this moves the smallest element (2) to the result:

0 1 2 4 3

We then move the smallest value of what remains (3) to get

0 1 2 3 4

and finally, we move the last element (4) to get the overall sorted sequence

0 1 2 3 4

How efficient of a sorting algorithm is this? In order to answer this question, we need to find some way to quantify how much work is being done. Once we've done that, we can analyze what that quantity is to determine just how efficient the overall algorithm is.

Intuitively, the selection sort algorithm works as follows;

- While there are still elements left to be sorted:
  - Scan across all of them to find the smallest of what remains.
  - Append that to the output.

It seems that appending the smallest remaining value to the output is unlikely to take a lot of time (we'll talk about this a bit more later). Scanning over each element to determine which element is smallest, on the other hand, might take quite a bit of time.

As an example, suppose we want to sort one million integers. Using selection sort, we'd begin by scanning over the entire list of one million elements to determine which was the smallest. Once we've done that, we then scan over the 999,999 remaining elements to determine which of them is the smallest. After that, we scan over the 999,998 remaining elements to determine which of them is the smallest, etc. This seems like it's going to take a while, but just how much time is it going to take?

In order to sort a list of $n$ values using selection sort, we need to scan $n$ elements on the first round, then $n - 1$ on the second, $n - 2$ on the third, etc. This means that the total number of elements that we're going to scan will be given by

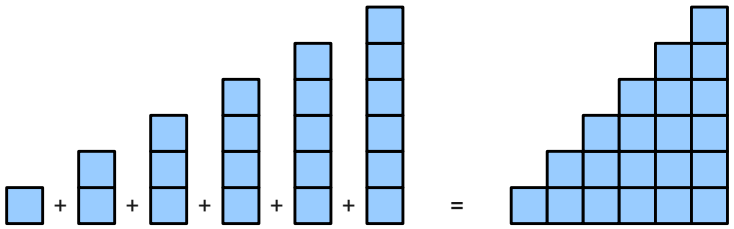
$$n + (n - 1) + (n - 2) + \ldots + 3 + 2 + 1$$

What is this value equal to? Well, that depends on our choice of $n$. If we try this for small values of $n$, we get the following:

- When $n = 1$, the sum is equal to 1.
- When $n = 2$, the sum is $1 + 2 = 3$.
- When $n = 3$, the sum is $1 + 2 + 3 = 6$.
- When $n = 4$, the sum is $1 + 2 + 3 + 4 = 10$.
- When $n = 5$, the sum is $1 + 2 + 3 + 4 + 5 = 15$.

Is there some sort of trend here? As with most interesting parts of mathematics, the answer is definitely "yes," but what exactly is this trend? When confronted with a sequence like this one (1, 3, 6, 10, 15, …) where we can't spot an immediate pattern, there are many techniques we can use to try to figure out what the sum is equal to. In fact, there are entire textbooks written on the subject.

In this case, one option might be to try drawing a picture to see if we can spot anything interesting. Suppose that we visualize each of the numbers as some quantity of blocks. We might draw 1 as one block, 2 as two blocks, 3 as three blocks, etc. Suppose that we place all of these blocks next to one another, like this:



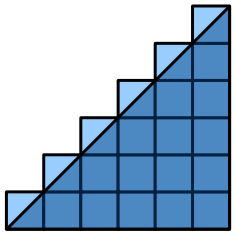
We now have a nice triangle shape to work with. If this were an actual triangle, we could try using the formula $A = \frac{1}{2}bh$ in order to compute the total area here. If we are summing $n + (n - 1) + \ldots + 2 + 1$, then the base of the triangle has width $n$ and the height has width $n$ as well. According to the formula for the area of a triangle, we'd therefore expect the number of blocks to be $\frac{1}{2}n^2$. Does this work? Unfortunately, no. The first few values of $\frac{1}{2}n^2$ are

$$0, 0.5, 2, 4.5, 8, 12.5, \ldots$$

whereas the first few values of the sum of the first $n$ positive natural numbers is

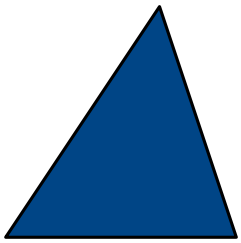
$$0, 1, 3, 6, 10, 15, \ldots$$

Why doesn't this reasoning exactly work? Well, if we superimpose a real triangle of width $n$ and height $n$ on top of our boxy triangle, we get this:

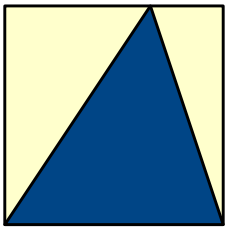


As you can see, our boxy triangle extends past the bounds of the real triangle by a small amount. This accounts for why the sum of the first $n$ positive natural numbers is a little bit bigger than $n^2$.

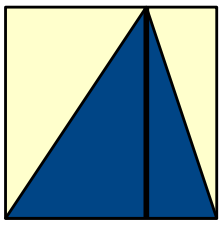
Although this doesn't exactly work out correctly, this geometric line of reasoning is actually quite interesting. Why exactly is the area of a triangle equal to $\frac{1}{2}bh$? One way to derive this is to start with any triangle we like, perhaps this one:



then to draw a box around it, like this:

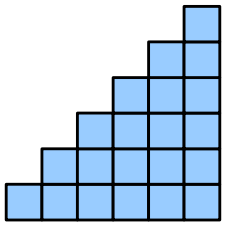


If we then draw a vertical line downward through the apex of the triangle, we get the following picture:

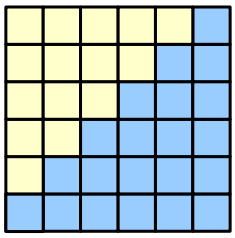


Notice that in each of the two pieces of the box, half of the area is filled up! This means that if we take the total area of the box ($bh$) and cut it in half, we should have the area of the triangle. Hence the area of the triangle is $\frac{1}{2}bh$.

Could we use this sort of reasoning to figure out what our sum is equal to? Well, we already have this triangle lying around:

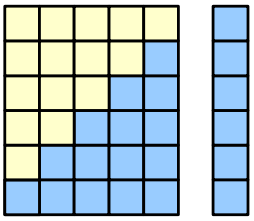


So perhaps we could do something similar to our triangle example by putting this collection of boxes into a larger box. Initially, we might try something like this:



In this picture, we can see that roughly half of the boxes in the large rectangle belong to our sum, while the other half do not. We can therefore get a rough estimate for $1 + 2 + \ldots + n$ as being about $n^2$. However, this is not an exact figure, because it's not an even split. For example, in the above figure, there are 21 boxes from our original sum, and 15 boxes that we added.

Although the above drawing doesn't exactly work, it's very close to what we want. There are several techniques we can use to fix it. One clever observation we can have is that the boxes we have added form a triangle of width $n - 1$ and height $n - 1$, compared to our original triangle, which has width $n$ and height $n$. Given this, suppose that we pull off the last column of our triangle. This gives us the following picture:



This picture gives us a very nice intuition for the sum. If we look at the rectangle on the left, we now have that exactly half of the boxes are from our original sum and exactly half of the boxes are from the completion. This box has width $n - 1$ and height $n$, so of the $n(n - 1)$ total boxes, one-half of them are from the original sum. We also have one final column from our original sum, which has $n$ boxes in it. This means that we might expect $1 + 2 + \ldots + n$ to be equal to

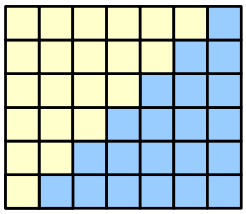
$$\frac{n(n-1)}{2} + n = \frac{n(n-1)}{2} + \frac{2n}{2} = \frac{n(n-1)+2n}{2} = \frac{n(n-1+2)}{2} = \frac{n(n+1)}{2}$$

Indeed, if we check the first few terms of $n(n + 1) / 2$, we end up getting the sequence

$$0, 1, 3, 6, 10, 15, 21, ...$$

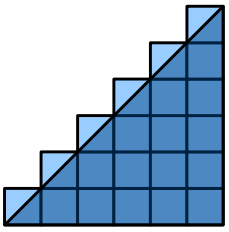
which matches our values for 0, 1, 1 + 2, 1 + 2 + 3, etc.

A different way of manipulating our diagram would be to change how we add in the extra boxes. If instead of creating a square, we create this rectangle:



then we can see that exactly half of the squares in this rectangle are used by our triangle. Since this rectangle has area $n(n + 1)$, this means that the sum of the first $n$ positive natural numbers should probably be $n(n + 1) / 2$, which agrees with our previous answer.

One final way we can think about the geometric intuition is to abandon the idea of completing the rectangle and to instead return to this earlier drawing, in which we superimposed the real triangle width and height $n$ on top of our boxy triangle:



As before, one idea might be to treat the total area of the boxes as the sum of two different areas – the area covered by the triangle, and the area filled by the pieces of the boxes extending above the triangle. If our triangle has width $n$ and height $n$, then there will be $n$ smaller triangles extending beyond the $n$ by $n$ triangle. Each of these triangles has width 1 and height 1, and therefore has area ½. Consequently, the total area taken up by our boxes would be given by the total area of the large triangle, plus $n$ copies of the smaller triangle. This is

$$\frac{n^2}{2} + \frac{n}{2} = \frac{n^2 + n}{2} = \frac{n(n+1)}{2}$$

And again we've reached the same result as before!

The takeaway point from this is that there are *always* different ways of thinking about problems in mathematics. You might end up at the same result through several different paths, each of which casts light on a slightly different angle of the problem.

So the big question is how this has anything to do with induction at all. Well, at this point we have a pretty good idea that the sum of the first $n$ positive natural numbers is going to be $n(n + 1) / 2$. But how would we rigorously establish this? Here, induction can be invaluable. We can prove that the above sum is correct by showing that it's true when $n = 0$, and then showing that if the sum is true for some choice of $n$, it must be true for $n + 1$ as well. By the principle of mathematical induction, we can then conclude that it must be true for any choice of $n$.

Here is one possible proof:

***Theorem:*** The sum of the first $n$ positive natural numbers is $n(n + 1) / 2$.

***Proof:*** By induction. Let $P(n)$ be "the sum of the first $n$ positive natural numbers is $n(n + 1) / 2$." We prove that $P(n)$ is true for all $n \in \mathbb{N}$, from which the result immediately follows.

For our base case, we prove $P(0)$, that the sum of the first 0 positive natural numbers is $0(0 + 1) / 2$. The sum of zero numbers is 0, and $0 = 0(0 + 1)/2$. Consequently, $P(0)$ holds.
For the inductive step, assume that for some $n \in \mathbb{N}$ that $P(n)$ is true and the sum of the first $n$ positive natural numbers is $n(n + 1) / 2$. We will prove that $P(n + 1)$ is true; that is, the sum of the first $n + 1$ positive natural numbers is $(n + 1)(n + 2) / 2$. Consider the sum of the first $n + 1$ positive natural numbers. This is the sum of the first $n$ positive natural numbers, plus $n + 1$. By our inductive hypothesis, the sum of the first $n$ positive natural numbers is $n(n + 1) / 2$. Thus the sum of the first $n + 1$ positive natural numbers is

$$\frac{n(n+1)}{2} + n + 1 = \frac{n(n+1)}{2} + \frac{2(n+1)}{2} = \frac{n(n+1) + 2(n+1)}{2} = \frac{(n+2)(n+1)}{2}$$

Thus $P(n + 1)$ is true, completing the induction. ■

What a trip this has been! We began by asking how efficient the selection sort algorithm was. In doing so, we made a detour into geometry to build up an intuition for the answer, and then used induction to formalize the result.

So back to our original question – how efficient is selection sort? Answer: not very. Selection sorting $n$ elements requires us to scan a total of $n(n + 1) / 2$ elements in the course of completing the algorithm. Plugging in $n = 1{,}000{,}000$ gives us that we will make 500,000,500,000 scans. That's 500 billion element lookups! Even a processor operating in the gigahertz range will take a while to finish sorting that way.

The beauty of the result that we have just proven, though, is that from this point forward if we ever see an algorithm that has this sort of behavior we immediately know how much work it will have to do.

## 3.2.1 Summation Notation

In the previous section, we considered the sum

$$1 + 2 + \ldots + (n - 1) + n$$

In the course of the proof, we kept referring to this sum as "the sum of the first $n$ positive natural numbers." This is a fairly long-winded way of explaining what sum we're computing, and it would be nice if there were a simpler way to do this.

When working with summations, mathematicians typically use Σ notation to describe the sum more compactly. Rather than writing out a sequence with an ellipsis in the middle, we instead describe a general formula for each individual term being summed together, then specify how many terms we want to sum up.

In general, we can describe the sum of $a_1 + a_2 + \ldots + a_n$ as follows:

$$\sum_{i=1}^{n} a_i$$

Let's piece this apart to see what it says. The large Σ indicates that we are looking at the sum of some number of terms. The values below and above the Σ tell us over what values the summation ranges. Here, the sum ranges from $i = 1$ to $n$, so we will sum up over the terms inside the sum when $i = 1$, $i = 2$, $i = 3$, etc. up to and including when $i = n$. Finally, we have the values actually being summed together. Here these are the values $a_i$. As $i$ ranges from 1 to $n$, we will aggregate and sum up all of these terms.

More formally:

> $\sum_{i=m}^{n} a_i$ is the sum of all $a_i$ where $i \in \mathbb{N}$ and $m \leq i \leq n$.

For example, let's consider our original sum $1 + 2 + \ldots + n$. We can write this sum as

$$\sum_{i=1}^{n} i$$

This says that we should sum up $i$ as $i$ ranges from 1 up to and including $n$. For example, if we pick a specific $n$ (say, $n = 5$), then we have that

$$\sum_{i=1}^{5} i = 1 + 2 + 3 + 4 + 5$$

If you are coming from a programming background, you can think of the summation as a sort of mathematical "for loop" that ranges over choices of $i$ and sums up the values that are listed.

Summations can have more complex bounds. For example, we could write the sum (-3) + (-2) + (-1) + 0 + 1 as

$$\sum_{i=-3}^{1} i = (-3) + (-2) + (-1) + 0 + 1$$

or could sum from 0 to 4 as

$$\sum_{i=0}^{4} i = 0 + 1 + 2 + 3 + 4$$

In addition to changing the loop bounds, we can also change what's inside the summation actually getting added up. For example, suppose that we wanted to sum up the first $n$ perfect cubes (that is, $0^3 + 1^3 + \ldots + (n - 1)^3$). We could write this as follows:

$$\sum_{i=0}^{n-1} i^3 = 0^3 + 1^3 + 2^3 + \ldots + (n-1)^3$$

There are two important details to note here. First, note that the upper bound on the sum is $n - 1$, not $n$, even though we're summing up the first $n$ perfect cubes. The reason for this is that the lower bound of the summation is 0, not 1. This means that there are a total of $n$ elements being summed up, not $n - 1$. Second, notice that the value being summed this time is not $i$, but $i^3$. In general, we can perform any arbitrary manipulations of the index of summation inside the sum. We could, for example, sum up powers of two this way:

$$\sum_{i=0}^{n-1} 2^i = 2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^{n-1}$$

When working with induction, one special kind of sum arises with surprising frequency. Consider what happens when we have a sum like this one:

$$\sum_{i=1}^{0} a_i$$

Notice that this sum is the sum from $i = 1$ to 0. What is the value of this sum? In this case, the sum doesn't include any numbers. We call sums like this one – where no values are being added up – **empty sums**. It is specified here:

> A sum of no numbers is called the ***empty sum*** and has value 0.

Thus all of the following sums are empty sums and therefore equal to zero:

$$\sum_{i=1}^{0} 2^i = 0 \qquad \sum_{i=137}^{42} (i + 1) = 0 \qquad \sum_{i=-1}^{-2} i = 0 \qquad \sum_{i=5}^{0} i^i = 0$$

However, note that the following sum is **not** empty:

$$\sum_{i=0}^{0} 2^i$$

Since the indices of summation are inclusive, this means that this sum includes the term where $i = 0$. Consequently, this sum is equal to $2^0 = 1$.

Empty sums may seem like little more than a curiosity right now, but they appear frequently in the base cases of inductive proofs.

Now that we have a formal notation we can use to manipulate sums, let's return back to our previous inductive proof. We proved that $1 + 2 + \ldots + n = n(n + 1) / 2$. What might this look like using summations? Well, we can rewrite the sum $1 + 2 + \ldots + n$ as

$$\sum_{i=1}^{n} i$$

Consequently, we can restate the theorem we have just proven as follows:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Let's repeat our previous proof, this time using summation notation. From this point forward, we will almost exclusively use summation notations in formal proofs involving sums.

***Theorem:*** For any $n \in \mathbb{N}$, $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

***Proof:*** By induction. Let $P(n)$ be defined as

$$P(n) \equiv \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction on $n$. As our base case, we prove $P(0)$, that is, that

$$\sum_{i=1}^{0} i = \frac{0(0+1)}{2}$$

The left-hand side of this equality is the empty sum, which is 0. The right-hand side of the equality is also 0, so $P(0)$ holds.

For the inductive step, assume that for some natural number $n$, $P(n)$ holds, meaning that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

We will prove $P(n + 1)$, meaning that

$$\sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}$$

To see this, note that

$$\sum_{i=1}^{n+1} i = \left( \sum_{i=1}^{n} i \right) + n + 1 = \frac{n(n+1)}{2} + n + 1 = \frac{n(n+1) + 2(n+1)}{2} = \frac{(n+2)(n+1)}{2}$$

Thus $P(n + 1)$ holds, completing the induction. ■

One of the key steps in this proof was recognizing that

$$\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + n + 1$$

Why does this step work? Well, the left-hand side is the sum of the first $n + 1$ positive natural numbers. The right-hand side is the sum of the first $n$ positive natural numbers, plus $n + 1$, the $(n + 1)^{st}$ positive natural number. All that we've done is "peel off" the last term of the sum to make it a bit easier to work with. This technique arises in many inductive proofs, since in order to reason about the sum of the first $n + 1$

terms of a series it may help to consider the sum of the first $n$ terms of the series, plus the $(n + 1)^{st}$ term by itself.

## 3.2.2 Summing Odd Numbers

Now that we have a framework for manipulating sums of numbers, let's do some exploration and see if we can find some other interesting sums to explore.

What happens if we start adding together the first $n$ odd numbers? If we do this, we'll find the following:

- The sum of the first 0 odd numbers is 0. It's an empty sum.
- The sum of the first 1 odd numbers is 1.
- The sum of the first 2 odd numbers is 1 + 3 = 4.
- The sum of the first 3 odd numbers is 1 + 3 + 5 = 9.
- The sum of the first 4 odd numbers is 1 + 3 + 5 + 7 = 16.

Now *that's* surprising... the first five terms of this sequence are 0, 1, 4, 9, 16 = $0^2$, $1^2$, $2^2$, $3^2$, $4^2$. Does this trend continue? If so, could we prove it? One of the beautiful consequences of mathematical induction is that once you have spotted a trend, you can sit down and attempt to prove that it is not a coincidence and in fact continues for all natural numbers. Even if we don't have an intuition for why the sums of odd numbers might work out this way, we can still prove that they must.

Let's see how a proof of this fact might work. First, we have to figure out what we want to show. Our goal is to show that the sum of the first $n$ odd numbers is equal to $n^2$. How would we phrase this as a summation? Well, we know that the odd numbers are numbers of the form $2k + 1$ for integers $k$, so one way of phrasing this sum would be

$$\sum_{i=0}^{n-1} (2i + 1)$$

Notice that the summation ranges from $i = 0$ to $n - 1$, so the sum has $n$ terms in it. It seems like this might cause a problem when $n = 0$, since then the sum ranges from 0 to -1. However, this is nothing to worry about. When $n = 0$, we don't want to sum anything up (we're talking about the sum of no numbers), and if we try evaluating a sum ranging from 0 to -1 we are evaluating an empty sum, which is defined to be 0.

Given this setup, let's try to prove that the sum of the first $n$ odd numbers is $n^2$.

***Theorem:*** For any natural number $n$, $\sum_{i=0}^{n-1}(2i+1)=n^2$.

***Proof***: By induction. Let $P(n)$ be defined as

$$P(n) \equiv \sum_{i=0}^{n-1}(2i+1)=n^2$$

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction on $n$. As our base case, we prove $P(0)$, that

$$\sum_{i=0}^{-1}(2i+1)=0^2$$

The left-hand side of this equality is the empty sum, which is 0. The right-hand side of the equality is also 0, so $P(0)$ holds.

For the inductive step, assume that for some natural number $n$, $P(n)$ holds, meaning that

$$\sum_{i=0}^{n-1}(2i+1)=n^2$$

We will prove P($n$ + 1), meaning that

$$\sum_{i=0}^{n}(2i+1)=(n+1)^2$$

To see this, note that

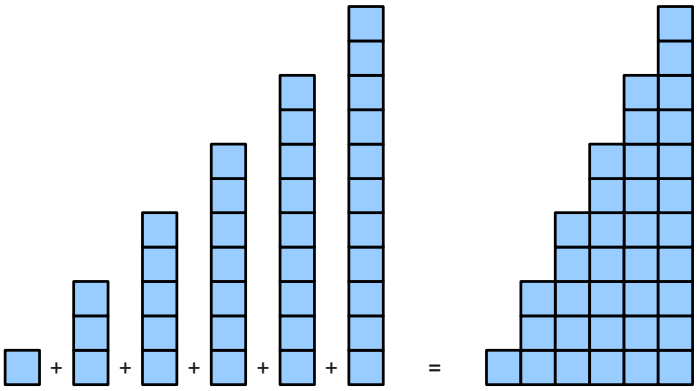
$$\sum_{i=0}^{n}(2i+1)=(n+1)^2=\sum_{i=0}^{n-1}(2i+1)+2n+1=n^2+2n+1=(n+1)^2$$

Thus P($n$ + 1) holds, completing the induction. ■

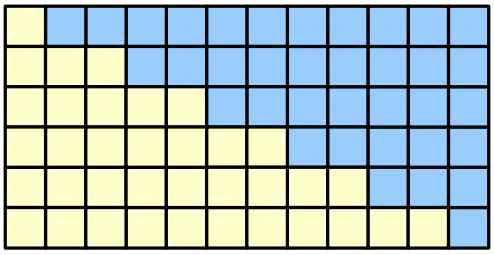
Here, the last step is a consequence of the fact that $(n + 1)^2$ expands out to $n^2 + 2n + 1$.

So we now have a mathematical proof of the fact that the sum of the first $n$ odd natural numbers is equal to $n^2$. But why is this? It's here that we see one shortcoming of induction as a technique. The above proof gives us almost no intuition as to why the result is correct.

We might then ask – so why is that true? As with our proof about the sum of the first $n$ positive natural numbers, it might help to draw a picture here. Suppose we start adding up odd numbers, like this:

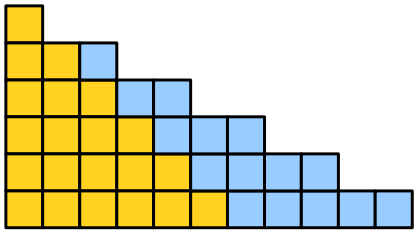


There isn't an immediate pattern here, but using the same intuition we had for the sum of the first $n$ natural numbers we might try completing this second rectangle to form some box:

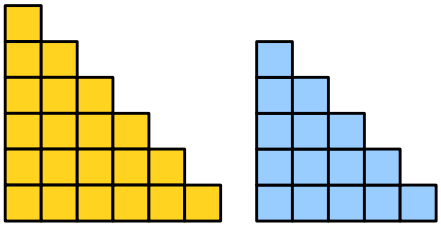


What are the dimensions of this larger box? Since there are $n$ odd numbers being summed up, the height of the box is $n$. The width of the box is one plus the largest odd number being added up. If we look at our summation, we note that we're adding up terms of the form $2i + 1$ and stop when $i = n - 1$. Plugging in $i = n - 1$ to get the largest odd number added up, we get $2(n - 1) + 1 = 2n - 2 + 1 = 2n - 1$. Since the width of the box is one more than this, we end up seeing that the width of the box is $2n$. Thus the box has dimensions $n$ by $2n$, so its area is $2n^2$. Since half of that area is used up by the boxes for our sum, the sum should be equal to $n^2$, as we saw before.

But this is just one intuition for the result. Let's look at our triangle one more time. We already know from before what the area of this highlighted triangle is:



This is the triangle we drew when we were considering the sum of the first $n$ positive natural numbers. So what remains in this picture? Well, notice that in the first row there are 0 blue squares, in the second row there is one blue square, in the third there are two blue squares, etc. In fact, the number of total blue squares is $1 + 2 + \ldots + n - 1$. We can see this by rearranging the above digram like this:

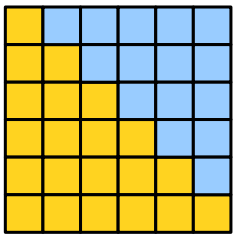


This means that the sum of the first $n$ odd numbers is the sum of the first $n$ positive natural numbers, plus the sum of the first $n - 1$ positive natural numbers. We happen to have formulas for these sums. If we add them together, we get the following:

$$\frac{n(n+1)}{2}+\frac{(n-1)n}{2}=\frac{n(n+1)+n(n-1)}{2}=\frac{n(n+1+n-1)}{2}=\frac{n(2n)}{2}=\frac{2n^2}{2}=n^2$$

*Et voilà*! We have our result.

But it turns out that with the above picture there's a much easier way of arriving at the result. What happens if we rotate the blue triangle 180 degrees? If we do this, we'll end up getting this picture:



The total area of this square is equal to the sum of the first $n$ odd numbers. Since the square has size $n \times n$, the sum of the first $n$ odd numbers should be $n^2$, as it indeed is.

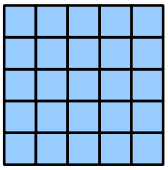
Of course, none of these intuitions match the intuition that we actually used in our proof. Let's revisit the proof for a short while to see if we can come up with a different explanation for why the result is true.

We can get a bit of an intuition from looking at the last step of the proof. If you'll notice, the proof works because given $n^2$, adding in $2n + 1$ (the $(n + 1)$st odd number) takes us up to $(n + 1)^2$. In other words, we have that

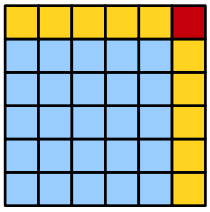
$$(n + 1)^2 - n^2 = n^2 + 2n + 1 - n^2 = 2n + 1$$

So one reason to think that this result would be true is that the spacing between consecutive powers of two is always an odd number. If we keep adding up odd numbers together over and over again, we thus keep advancing from one term in the sequence to the next.

But why exactly is that? It turns out that, again, there is a beautiful geometric intuition. Suppose that we have the sum of the first $n$ odd numbers, which we know is equal to $n^2$. We can draw this as follows:



Now, suppose that we add in the $(n + 1)$st odd number, which is $2n + 1$. One way to visualize what happens when we do this is to break the $2n + 1$ apart into three pieces – one piece of size $n$, one piece of size $n$, and one piece of size 1. Doing so gives us the following:



As you can see, these three pieces can be added to the square in a way that extends it into a square of the next size. This in a sense justifies our induction. The reason that the sum of the first $n$ odd numbers is equal to $n^2$ is because each odd number contributes enough to the previous perfect square to get us up to the next perfect square.

This intuition is actually extremely powerful, and we can use it as a stepping stone toward a larger result. The key idea is to think about what we just did backwards. Here, we started off with the sum of the first $n$ odd numbers and ended up with the perfect squares. But in reality, our proof works the other way. We showed that you can progress from one perfect square to the next by adding in the $(n + 1)^{st}$ odd number. This was almost a pure coincidence – it wasn't the fact that it was the $(n + 1)^{st}$ odd number so much as it was

the value $2n + 1$, which is the difference between two adjacent terms in the sequence. The fact that we call numbers of the form $2n + 1$ the odd numbers was entirely accidental.

### 3.2.3 Manipulating Summations

Any good mathematical proof can be lifted into a more general setting, as you will see repeatedly throughout your exploration of the mathematical foundations of computing. The proof we just completed about sums of odd numbers can indeed be generalized to a more elaborate and more powerful result that we can use to derive all sorts of results without having to directly resort to induction.

Before jumping off, let's review something that we already happened to know. We currently know a formula for the sum of the first $n$ positive integers; specifically, we have that

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Let's update this so that we have this formula phrased in terms of the sum of the first $n$ natural numbers, rather than positive integers. If we consider this value, we get the sum

$$\sum_{i=0}^{n-1} i$$

What is the value of this sum? Well, one thing we can note is the following:

$$\sum_{i=0}^{n-1} i = \sum_{i=0}^{n-1} i + n - n = \sum_{i=0}^{n} i - n$$

Now, we can exploit the fact that this first sum is equal to $0 + 1 + 2 + \ldots + n$. This has exactly the same value as $1 + 2 + \ldots + n$, because the zero doesn't contribute anything. In other words, we can just restart this summation at 0 rather than at 1 without changing the value of the expression. This gives us

$$\sum_{i=0}^{n-1} i = \sum_{i=0}^{n} i - n = \sum_{i=1}^{n} i - n = \frac{n(n+1)}{2} - n = \frac{n(n+1) - 2n}{2} = \frac{n(n+1-2)}{2} = \frac{n(n-1)}{2}$$

In other words, we have just shown that

$$\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$$

This formula is extremely important – you should definitely commit it to memory!

What we have just done right here was derive a new result about a summation based off an old result about summations. Most of the time that you need to evaluate summations, you can use standard techniques like these to get a nice value for the summation without having to draw pictures or use induction. This section focuses on some standard identities you can use to simplify summations, along with the proofs of why they work. In a sense, the proofs that we will do here will serve as lemmas that we can use later on when simplifying sums we have never seen before. Although most of the proofs here may seem obvious, it's good to justify that they always work correctly.

Right now, we have a nice closed-form solution for the sum of the first $n$ natural numbers. More explicitly, we have this:

$$\sum_{i=0}^{n-1} i^1 = \frac{n(n-1)}{2}$$

I've explicitly highlighted the fact that we're computing $0^1 + 1^1 + 2^1 + \ldots + (n - 1)^1$. What if we were to change the exponent? What effect would this have on the sum?

One simple change would be to set the exponent to 0, meaning we'd be computing $0^0 + 1^0 + 2^0 + \ldots (n - 1)^0 = 1 + 1 + \ldots + 1$.* This should come out to $n$, since we're adding 1 up $n$ times. This is indeed the case, as seen in this proof:

***Theorem:*** For any natural number $n$, $\sum_{i=0}^{n-1} 1 = n$.

***Proof***: By induction. Let P($n$) be defined as

$$P(n) \equiv \sum_{i=0}^{n-1} 1 = n$$

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction on $n$. As our base case, we prove $P(0)$, that is, that

$$\sum_{i=0}^{-1} 1 = 0$$

The left-hand side of this equality is the empty sum, which is 0. The right-hand side of the equality is also 0, so P(0) holds.

For the inductive step, assume that for some natural number $n$, $P(n)$ holds, meaning that

$$\sum_{i=0}^{n-1} 1 = n$$

We will prove $P(n + 1)$, meaning that

$$\sum_{i=0}^{n} 1 = n + 1$$

To see this, note that

$$\sum_{i=0}^{n} 1 = \sum_{i=0}^{n-1} 1 + 1 = n + 1$$

Thus $P(n + 1)$ holds, completing the induction. ■

This proof might seem silly, but it's good to be able to confirm results that we intuitively know to be true. This gives us a nice starting point for future work.

So we now know how to sum up $n^0$ and $n^1$ from zero forward. What other sums might we be interested in simplifying? One thing we might do at this point would be to revisit our earlier proof about sums of odd numbers. We proved explicitly by induction that

$$\sum_{i=0}^{n-1} (2i + 1) = n^2$$

* This assumes that $0^0 = 1$. In most of discrete mathematics, this is a perfectly reasonable assumption to make, and we will use this convention throughout this course.

Could we somehow prove this result without using induction? Here is one possible approach that we might be able to use. Right now we know how to sum up $n^0$ (1) and $n^1$ ($n$). Could we perhaps decompose this sum into these two pieces?

$$\sum_{i=0}^{n-1}(2i+1)=\sum_{i=0}^{n-1}2i+\sum_{i=0}^{n-1}1$$

We already know a value for the second term, since we explicitly proved that this sum is equal to $n$. This means that we have

$$\sum_{i=0}^{n-1}(2i+1)=\sum_{i=0}^{n-1}2i+n$$

It seems like we also should be able to simplify the first sum like this:

$$\sum_{i=0}^{n-1}(2i+1)=2\sum_{i=0}^{n-1}i+n$$

From there, we can use our formula for the second sum to get

$$\sum_{i=0}^{n-1}(2i+1)=2\frac{n(n-1)}{2}+n=n(n-1)+n=n^2-n+n=n^2$$

And we have an entirely new proof of the fact that the sum of the first $n$ odd numbers is equal to $n^2$. But unfortunately, this proof makes two intuitive leaps that we haven't yet justified. First, why can we split the initial sum up into two separate sums? Second, why can we factor a constant out of the sum? Both of these steps are reasonable because of properties of addition, but are we sure they work for the general case? The answer turns out to be “yes,” so the above proof is valid, but let's take a minute to prove each of these.

Our first proof will be something to justify why we can split a summation of the sum of two terms into two separate summations. Rather than just prove it for the case above, we'll prove a more general result that will let us split apart arbitrary summations containing the sum of two terms.

***Theorem:*** For any natural number $n$, $\sum_{i=0}^{n-1}(a_i + b_i) = \sum_{i=0}^{n-1} a_i + \sum_{i=0}^{n-1} b_i$.

***Proof:*** By induction. Let $P(n)$ be defined as

$$P(n) \equiv \sum_{i=0}^{n-1}(a_i + b_i) = \sum_{i=0}^{n-1} a_i + \sum_{i=0}^{n-1} b_i$$

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction on $n$. As our base case, we prove $P(0)$, that is:

$$\sum_{i=0}^{-1}(a_i + b_i) = \sum_{i=0}^{-1} a_i + \sum_{i=0}^{-1} b_i$$

All three of these sums are the empty sum. Since $0 = 0 + 0$, $P(0)$ holds.

For the inductive step, assume that for some $n \in \mathbb{N}$, $P(n)$ holds, so

$$\sum_{i=0}^{n-1}(a_i + b_i) = \sum_{i=0}^{n-1} a_i + \sum_{i=0}^{n-1} b_i$$

We will prove $P(n + 1)$, meaning that

$$\sum_{i=0}^{n}(a_i + b_i) = \sum_{i=0}^{n} a_i + \sum_{i=0}^{n} b_i$$

To see this, note that

$$\sum_{i=0}^{n}(a_i + b_i) = \sum_{i=0}^{n-1}(a_i + b_i) + a_n + b_n = \sum_{i=0}^{n-1} a_i + \sum_{i=0}^{n-1} b_i + a_n + b_n = \sum_{i=0}^{n} a_i + \sum_{i=0}^{n} b_i$$

Thus $P(n + 1)$ holds, completing the induction. ■

Great! We've established that we can split apart a summation of sums into two independent summations. If we can prove that we can always factor a constant term out of a summation, then we will be able to rigorously justify every step of our alternate proof about the sum of the first $n$ odd numbers. More formally, we want to prove the following:

***Theorem:*** For any natural number $n$ and any $r \in \mathbb{R}$, $\sum_{i=0}^{n-1} r \cdot a_i = r \cdot \sum_{i=0}^{n-1} a_i$.

This theorem is different from the other theorems we have proved by induction so far. Previously, our theorems have had the form "for any natural number $n$, $P(n)$ is true." In this case, we have two separate variables that we have to consider – the natural number $n$, and the real number $r$. How might we go about proving this?

Remember that we write a proof by induction by choosing some property $P(n)$, then proving that $P(n)$ is true for all $n \in \mathbb{N}$. What if we chose our property $P(n)$ such that if $P(n)$ holds true for all $n \in \mathbb{N}$, the overall theorem is true as well? Specifically, since we want our result to work for all $n \in \mathbb{N}$ and for all $r \in \mathbb{R}$, what if we chose our property $P(n)$ as follows:

$$P(n) \equiv \text{“For all } r \in \mathbb{R},\ \sum_{i=0}^{n-1} r \cdot a_i = r \cdot \sum_{i=0}^{n-1} a_i \text{”}$$

Now, if $P(n)$ is true for all natural numbers $n$, then we must have that for any real number $r$, we can factor $r$ out of a summation.

Given this, let's see what a proof of the theorem might look like:

***Proof:*** By induction. Let $P(n)$ be defined as

$$P(n) \equiv \text{for any } r \in R, \sum_{i=0}^{n-1} r \cdot a_i = r \cdot \sum_{i=0}^{n-1} a_i$$

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction on $n$. As our base case, we prove P(0), that is, that for any $r \in \mathbb{R}$,

$$\sum_{i=0}^{-1} r \cdot a_i = r \cdot \sum_{i=0}^{-1} a_i$$

Both of these sums are the empty sum, and we have that $0 = r \cdot 0$ for all $r \in \mathbb{R}$. Thus $P(0)$ holds.

For the inductive step, assume that for some n ∈ ℕ, P(n) holds, so for any $r \in \mathbb{R}$:

$$\sum_{i=0}^{n-1} r \cdot a_i = r \cdot \sum_{i=0}^{n-1} a_i$$

We will prove $P(n + 1)$, meaning that for any $r \in \mathbb{R}$, we have

$$\sum_{i=0}^{n} r \cdot a_i = r \cdot \sum_{i=0}^{n} a_i$$

To see this, consider any $r \in \mathbb{R}$. Then

$$\sum_{i=0}^{n} r \cdot a_i = \sum_{i=0}^{n-1} r \cdot a_i + r \cdot a_n = r \cdot \sum_{i=0}^{n-1} a_i + r \cdot a_n = r\left(\sum_{i=0}^{n-1} a_i + a_n\right) = r \sum_{i=0}^{n} a_n$$

Thus $P(n + 1)$ holds, completing the induction. ■

We now have four results we can use when trying to prove results about sums:

1. $\sum_{i=0}^{n-1} 1 = n$
2. $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$
3. $\sum_{i=0}^{n-1} (a_i + b_i) = \sum_{i=0}^{n-1} a_i + \sum_{i=0}^{n-1} b_i$
4. $\sum_{i=0}^{n-1} r \cdot a_i = r \cdot \sum_{i=0}^{n-1} a_i$

These four results make it possible to quickly evaluate many useful sums without having to resort to induction. For example, suppose that we want to evaluate the following sum, which represents the sum of the first $n$ even natural numbers:

$$\sum_{i=0}^{n-1} 2i$$

We can just compute this directly:

$$\sum_{i=0}^{n-1} 2i = 2\sum_{i=0}^{n-1} i = 2\frac{n(n-1)}{2} = n(n-1) = n^2 - n$$

This saves us the effort of having to even evaluate a few terms of the series to see if we can spot any trends! We immediately know the answer. To verify that it's correct, let's plug in a few terms to see if it matches our expectation:

- The sum of the first 0 even numbers is 0, and $0^2 - 0 = 0$.
- The sum of the first 1 even number is 0, and $1^2 - 1 = 1$..
- The sum of the first 2 even numbers is $0 + 2 = 2$, and $2^2 - 2 = 2$.
- The sum of the first 3 even numbers is $0 + 2 + 4 = 6$, and $3^2 - 3 = 6$.
- The sum of the first 4 even numbers is $0 + 2 + 4 + 6 = 12$, and $4^2 - 4 = 12$.

It's amazing how much simpler it is to analyze summations this way!

## 3.2.4 Telescoping Series

We now have some general techniques that we can use to manipulate summations. We are about to see a simple but powerful technique that makes it possible to evaluate more complex summations than before – **telescoping series**. Earlier in this chapter, we saw that the sum of the first $n$ odd numbers was $n^2$. We saw several ways to prove this, but the particular inductive approach we used was based on the fact that the difference of two consecutive perfect squares is an odd number. In fact, the odd numbers can be thought of as the differences between consecutive perfect squares. The insight we needed was that $(n + 1)^2 - n^2 = 2n + 1$. Given this, let's revisit our formula for the sum of the first $n$ perfect squares. The initial summation is

$$\sum_{i=0}^{n-1} (2i + 1)$$

Now, from above, we have that $2i + 1 = (i + 1)^2 - i^2$. As a result, we can replace the term $2i + 1$ in the summation with the expression $(i + 1)^2 - i^2$. What happens if we do this? In that case, we get this:

$$\sum_{i=0}^{n-1} ((i+1)^2 - i^2)$$

If we expand this sum out, something amazing starts to happen. Here's the sum when $n = 0$, 1, 2, 3, and 4:

$$\sum_{i=0}^{-1} ((i+1)^2 - i^2) = 0$$

$$\sum_{i=0}^{0} ((i+1)^2 - i^2) = (1^2 - 0^2) = 1$$

$$\sum_{i=0}^{1} ((i+1)^2 - i^2) = (2^2 - 1^2) + (1^2 - 0^2) = 2^2 - 0^2 = 4$$

$$\sum_{i=0}^{2} ((i+1)^2 - i^2) = (3^2 - 2^2) + (2^2 - 1^2) + (1^2 - 0^2) = 3^2 - 0^2 = 9$$

$$\sum_{i=0}^{3} ((i+1)^2 - i^2) = (4^2 - 3^2) + (3^2 - 2^2) + (2^2 - 1^2) + (1^2 - 0^2) = 4^2 - 0^2 = 16$$

Notice what starts to happen as we expand out these summations. Each term in the sum is a difference of two terms, where the second term of difference is the first term of the next difference. As a result, all of the inner terms completely disappear, and we're left with the difference of the first term and the last term.

One quick formalism:

The sum $\sum_{i=0}^{n-1}(x_{i+1}-x_i)$ is called a ***telescoping series***.

If we evaluate this sum, then the adjacent pairs will continuously collapse and eventually all that will remain are the first and last elements. We can formally prove this here:

***Theorem:*** For all natural numbers $n$, $\sum_{i=0}^{n-1}(x_{i+1}-x_i)=x_n-x_0$

***Proof***: By induction. Let $P(n)$ be defined as

$$P(n)\equiv\sum_{i=0}^{n-1}(x_{i+1}-x_i)=x_n-x_0$$

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction on $n$. As our base case, we prove $P(0)$, that is:

$$\sum_{i=0}^{-1}(x_{i+1}-x_i)=x_0-x_0$$

In this case, the left-hand side is the empty sum and the right-hand side is zero, so $P(0)$ holds.

For the inductive step, assume that for some $n \in \mathbb{N}$, P($n$) holds, so

$$\sum_{i=0}^{n-1}(x_{i+1}-x_i)=x_n-x_0$$

We will prove $P(n + 1)$, meaning

$$\sum_{i=0}^{n}(x_{i+1}-x_i)=x_{n+1}-x_0$$

To see this, notice that

$$\sum_{i=0}^{n}(x_{i+1}-x_i)=\sum_{i=0}^{n-1}(x_{i+1}-x_i)+x_{n+1}-x_n=(x_{n+1}-x_n)+(x_n-x_0)=x_{n+1}-x_0$$

Thus $P(n + 1)$ holds, completing the induction. ■

This result might not initially seem very important, but combined with our previous results we can now solve various summations that we previously would not be able to. For example, we happen to know the value of the following sums:

$$\sum_{i=0}^{n-1}1=n \qquad \sum_{i=0}^{n-1}i=\frac{n(n-1)}{2}$$

What is the value of the following sum?

$$\sum_{i=0}^{n-1} i^2$$

If we start expanding out some terms, we get the following sequence:

0, 1, 5, 14, 30, 55, …

It's hard to spot a pattern here. If we turn to geometry, we'll find that it's surprisingly tricky to get a good solid geometric intuition for this sum. What other tricks can we try?

Previously, we considered the difference of $(n + 1)^2$ and $n^2$ to learn something about sums of odd numbers. Now, let's suppose that we didn't already know the fact that the sum of the first $n$ natural numbers is $n(n - 1) / 2$. We could have figured this out as follows. Using properties of telescoping sums, we know that

$$\sum_{i=0}^{n-1} \left((i+1)^2 - i^2\right) = n^2$$

If we simplify the inside of this sum, we get that

$$\sum_{i=0}^{n-1} \left((i+1)^2 - i^2\right) = \sum_{i=0}^{n-1} \left(i^2 + 2i + 1 - i^2\right) = \sum_{i=0}^{n-1} (2\mathrm{i} + 1) = n^2$$

Now, using properties of summations, we can simplify this as follows:

$$n^2 = \sum_{i=0}^{n-1} (2\mathrm{i} + 1) = 2\sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 1$$

We can replace this final sum with $n$, which we already know, to get

$$n^2 = 2\sum_{i=0}^{n-1} i + n$$

If we subtract $n$ from both sides and divide by two, we get

$$\frac{n^2 - n}{2} = \sum_{i=0}^{n-1} i$$

Since we know that $n^2 - n = n(n - 1)$, we have just derived the formula for the sum of the first $n$ natural numbers in a completely different way.

The reason that this derivation is important is that we can use an almost identical trick to determine the sum of the first $n$ perfect squares. The idea is as follows. When working with the difference $(n + 1)^2 - n^2$, we were able to derive the formula for the sum of the natural numbers raised to the first power. What happens if we try considering the difference $(n + 1)^3 - n^3$? Well, using what we know about telescoping series, we can start off like this:

$$\sum_{i=0}^{n-1} \left((i+1)^3 - i^3\right) = n^3$$

Since $(i + 1)^3 - i^3 = i^3 + 3i^2 + 3i + 1 - i^3 = 3i^2 + 3i + 1$, this means that

$$\sum_{i=0}^{n-1} (3\mathrm{i}^2 + 3\mathrm{i} + 1) = n^3$$

Using the properties that we just developed, we can split our sum into three sums:

$$3\sum_{i=0}^{n-1} i^2 + 3\sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 1 = n^3$$

We already know values for these last two sums, so let's go simplify them:

$$3\sum_{i=0}^{n-1} i^2 + \frac{3n(n-1)}{2} + n = n^3$$

If we now try to isolate the mystery sum, we get the following:

$$3\sum_{i=0}^{n-1} i^2 = n^3 - \frac{3n(n-1)}{2} - n$$

$$\sum_{i=0}^{n-1} i^2 = \frac{n^3}{3} - \frac{n(n-1)}{2} - \frac{n}{3}$$

All that's left to do now is to simplify the right-hand side to make it easy to read:

$$\sum_{i=0}^{n-1} i^2 = \frac{n^3}{3} - \frac{n(n-1)}{2} - \frac{n}{3} = \frac{2n^3}{6} - \frac{3n(n-1)}{6} - \frac{2n}{6} = \frac{2n^3 - 3n(n-1) - 2n}{6}$$

$$= \frac{n(2n^2 - 3(n-1) - 2)}{6} = \frac{n(2n^2 - 3n + 3 - 2)}{6} = \frac{n(2n^2 - 3n + 1)}{6}$$

$$= \frac{n(n-1)(2n-1)}{6}$$

And so we can conclude (correctly) that

$$\sum_{i=0}^{n-1} i^2 = \frac{n(n-1)(2n-1)}{6}$$

The beauty of building up all of this machinery is that we don't need to do any induction at all to prove that this result is correct. We have arrived at this result purely by applying theorems that we have developed earlier. The key insight – treating odd numbers as the difference of adjacent perfect squares – is a very powerful one, and using techniques similar to what we've developed above it's possible to find formulas for the sum of the first *n* *k*th powers of natural numbers for any arbitrary natural number *k*.

So far, we have restricted ourself to using telescoping sums to reason about summations of terms of the form $x^n$ for some $n$. That is, we have sums of $x^0$, $x^1$, and $x^2$. But we can come up with summations for other sequences as well. For example, consider the series of powers of two: $2^0$, $2^1$, $2^2$, $2^3$, … = 1, 2, 4, 8, … . We might consider what happens when we start summing these numbers together. For example, we have the following:

$$\sum_{i=0}^{-1} ((i+1)^2 - i^2) = 0$$

$$\sum_{i=0}^{0} 2^i = 1$$

$$\sum_{i=0}^{1} 2^i = 1 + 2 = 3$$

$$\sum_{i=0}^{2} 2^i = 1 + 2 + 4 = 7$$

Can we spot a pattern here? Well, the sequence 0, 1, 3, 7, … is one less than the sequence 1, 2, 4, 8, …; that is, the actual sequence of powers of two. That's interesting... does the trend continue?

It turns out that the answer is yes. We could prove this by using a brand-new induction, but there's a much simpler and more direct way to accomplish this. Using our idea of manipulating telescoping series, we have the following:

$$\sum_{i=0}^{n-1} (2^{i+1} - 2^i) = 2^n - 2^0 = 2^n - 1$$

So what is $2^{i+1} - 2^i$? Well, doing some simple algebra tells us that

$$2^{i+1} - 2^i = 2(2^i) - 2^i = 2^i$$

Using this, we can simplify the above summation to get

$$\sum_{i=0}^{n-1} (2^{i+1} - 2^i) = \sum_{i=0}^{n-1} 2^i = 2^n - 1$$

So the sum of the first $n$ powers of 2 is $2^n - 1$. Amazing!

But why stop here? What happens if we sum up $3^0 + 3^1 + 3^2 + \ldots + 3^{n-1}$? Do we get $3^n - 1$, as we did with powers of two? Well, let's start summing up some terms in the series to see what we get:

$$\sum_{i=0}^{-1} 3^i = 0$$

$$\sum_{i=0}^{0} 3^i = 1$$

$$\sum_{i=0}^{1} 3^i = 1 + 3 = 4$$

$$\sum_{i=0}^{2} 3^i = 1 + 3 + 9 = 13$$

$$\sum_{i=0}^{3} 3^i = 1 + 3 + 9 + 27 = 40$$

This sequence (0, 1, 4, 13, 40, …) doesn't seem connected to the sequence of powers of three (1, 3, 9, 27, 81, …) in an immediately obvious way. If we were to just use induction here, we would fail before we started because we don't even have an idea of what we'd be trying to prove.

However, using our technique of telescoping sums, we can make the following observations. What happens if we consider the sum of differences of powers of three? That worked well for the case when we had powers of two, so perhaps it will work here as well. If we try this, we get the following starting point:

$$\sum_{i=0}^{n-1}\left(3^{i+1}-3^{i}\right)=3^{n}-1$$

So what is $3^{i+1} - 3^{i}$? In this case, it's *not* equal to $3^{i}$. However, a little arithmetic tells us that

$$3^{i+1}-3^{i}=3\left(3^{i}\right)-3^{i}=\left(3-1\right)3^{i}=2\cdot 3^{i}$$

So we can return to our original sum and simplify it as follows:

$$\sum_{i=0}^{n-1}\left(3^{i+1}-3^{i}\right)=\sum_{i=0}^{n-1}\left(2\cdot 3^{i}\right)=3^{n}-1$$

Finalizing the math with some arithmetic:

$$\sum_{i=0}^{n-1}\left(2\cdot 3^{i}\right)=3^{n}-1$$

$$2\sum_{i=0}^{n-1}3^{i}=3^{n}-1$$

$$\sum_{i=0}^{n-1}3^{i}=\frac{3^{n}-1}{2}$$

It turns out that this works out just beautifully. If we start with the sequence of powers of three:

1, 3, 9, 27, 81, 243, …

then subtract one, we get

0, 2, 8, 26, 80, 242, …

Dividing by two gives

0, 1, 4, 13, 40, 121, …

which indeed agrees with the sequence we had before.

To end on a high note, let's see if we can generalize this even further. Suppose we have the sum $k^0 + k^1 + k^2 + \ldots + k^{n-1}$ for some real number $k$. What is this sum equal to? Using the same trick we used for the case where $k = 3$, we start off by writing out the telescoping series:

$$\sum_{i=0}^{n-1}\left(k^{i+1}-k^{i}\right)=k^{n}-1$$

We can simplify the term inside the summation by rewriting it as

$$\sum_{i=0}^{n-1}\left(k^{i}\left(k-1\right)\right)=k^{n}-1$$

Now, let's assume that $k$ is not equal to one, meaning that $k - 1 \neq 0$. As a good self-check, think about what happens if we let $k = 1$; why is it reasonable to assume that $k \neq 1$ here? Given this assumption, we can then do the following:

$$\left(k-1\right)\sum_{i=0}^{n-1}k^{i}=k^{n}-1$$

$$\sum_{i=0}^{n-1} k^i = \frac{k^n - 1}{k - 1}$$

And we now have a way of simplifying sums of powers!

The techniques we have developed in this section extend to a much more elaborate system called the *finite calculus*, in which the notion of differences of adjacent terms take on a role analogous to integrals and derivatives in standard calculus. There are many good books and tutorials on the subject, and you're strongly encouraged to explore the finite calculus if you have the time to do so!

### 3.2.5 Products

As a closing remark for our discussion on summations, we should note that just as there is Σ notation for summations, there is a corresponding **Π notation** for products. The definition is analogous:

> $\prod_{i=m}^{n} a_i$ is the product of all $a_i$ where $i \in \mathbb{N}$ and $m \leq i \leq n$.

For example:

$$\prod_{i=1}^{5} i = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

Just as the empty sum is defined to be 0, the empty *product* is defined to be one.

A product of no numbers is called the **empty product** and is equal to 1.

There are many functions that are defined in terms of products. One special function worth noting is the *factorial* function, which is defined as follows:

> For any $n \in \mathbb{N}$, ***n factorial***, denoted $n!$, is defined as $n! = \prod_{i=1}^{n} i$

For example, 0! = 1 (the empty product), 1! = 1, 2! = 2, 3! = 6, 4! = 24, etc. We'll return to factorials in a later chapter as we explore combinatorics.

## 3.3 Induction and Recursion

There is a close connection between mathematical induction and recursion. In mathematical induction, we prove that something is true by proving that it holds for some simple case, then proving that each case implies the next case. In recursion, we solve a problem by identifying how to solve some simple case of that problem, and solve larger instances of the problem by breaking those instances down into smaller instances. This similarity makes it possible to use induction to reason about recursive programs and to prove their correctness.

As an example, consider the following recursive C function, which computes $n!$:

```
int factorial(int n) {
    if (n == 0) return 1;
    return n * factorial(n - 1);
}
```

How can we be sure that this actually computes *n* factorial? Looking over this code, in a sense it's "obvious" that this should work correctly. But how would we actually prove this?

This is where induction comes in. To prove that this function works correctly, we will show that for any natural number *n*, that the factorial function, as applied to *n*, indeed produces *n*!.* This in a sense plays out the recursion backwards. The recursive function works by calling itself over and over again with smaller inputs until it reaches the base case. Our proof will work by growing our knowledge of what this function does from the bottom-up until we have arrived at a proof that the factorial function works for our given choice of *n*.

***Theorem:*** For any $n \in \mathbb{N}$, `factorial(n)` = *n*!.

***Proof:*** By induction on *n*. Let P(*n*) be "`factorial(n)` = *n*!." We will prove that P(*n*) is true for all $n \in \mathbb{N}$.

As our base case, we prove P(0), that `factorial(0)` = 0!. By inspection, we have that `factorial(0)` = 1, and since 0! = 1.

For our inductive step, assume for some $n \in \mathbb{N}$ that P(*n*) holds and `factorial(n)` = n!. We prove P(*n* + 1), that `factorial(n + 1)` = (*n* + 1)!. To see this, note that since $n + 1 \neq 0$, `factorial(n + 1)` will return $(n + 1) \times$ `factorial`$((n + 1) - 1) = (n + 1) \times$ `factorial(n)`. By our inductive hypothesis, `factorial(n)` = n!, so `factorial(n+1)` = $(n + 1) \times n! = (n + 1)!$. Thus P(*n* + 1) holds, completing the induction. ■

This proof is mostly a proof-of-concept (no pun intended) that we can use induction to prove properties of recursive functions. Now that we know we can use induction this way, let's use it to explore some slightly more involved recursive functions.

The next example we will work with will involve recursive functions applied over lists of elements. Many programming languages, such as LISP and Haskell, use recursion and lists as their primary means of computation, while other languages like JavaScript and Python support this style of programming quite naturally. In the interests of clarity, rather than writing programs out using any concrete programming language, we'll use a pseudocode language that should be relatively easy to read. The main pieces of notation we will need are the following:

- If *L* is a list of elements, then *L*[n] refers to the *n*th element of that list, zero-indexed. For example, if *L* = (E, B, A, D, C), then *L*[0] = E, *L*[1] = B, etc.
- If *L* is a list of elements, then *L*[*m*:] refers to the sublist of *L* starting at position *m*. For example, with *L* defined as above, *L*[1:] = (B, A, D, C) and *L*[2:] = (A, D, C)
- If *L* is a list of elements, then |*L*| refers to the number of elements in *L*.

* Okay... technically speaking, this isn't 100% true because `int`s can't hold arbitrarily large values. We'll gloss over this detail here, though when formally verifying arbitrary programs you should be very careful to watch out for this case!

Now, let's suppose that we have a list of real numbers and want to compute their sum. Thinking recursively, we might break this problem down as follows:

- The sum of a list with no numbers in it is the empty sum, which is 0.
- The sum of a list of $(n + 1)$ numbers is the sum of the first number, plus the sum of the remaining $n$ numbers.

Written out in our pseudocode language, we might write this function as follows:

```
function sum(list L) {
    if |L| = 0:
        return 0.
    else:
        return L[0] + sum(L[1:]).
}
```

To see how this works, let's trace the execution of the function on the list (4, 2, 1):

```
  sum(4, 2, 1)
= 4 + sum(2, 1)
= 4 + (2 + sum(1))
= 4 + (2 + (1 + sum()))
= 4 + (2 + (1 + 0))
= 4 + (2 + 1)
= 4 + 3
= 7
```

The reason that the above recursive function works is that every time we call the **`sum`** function, the size of the list shrinks by one element. We can't shrink the list forever, so eventually we hit the base case.

So how might we prove that this function works correctly? With our previous recursive function, it made sense to prove the function was correct using recursion, because the argument to the function was itself a natural number. Now, the argument to our function is a list of values. Fortunately, though, this does not end up causing any problems. Although the actual list itself is not a natural number, the *length* of that list is a natural number. We can therefore prove the correctness of our algorithm by showing that it works correctly for lists of any length. This trick – using induction on the size or shape of some object – enables us to use recursion to prove results that don't directly apply to the natural numbers.

***Theorem:*** For any list $L$, **sum**($L$) = $\sum_{i=0}^{|L|-1} L[i]$

***Proof***: By induction. Let $P(n)$ be defined as follows:

$P(n)$ = "For any list $L$ of length $n$, **sum**($L$) = $\sum_{i=0}^{|L|-1} L[i]$.

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction. As our base case, we prove $P(0)$, that for any list $L$ of length 0, that **sum**($L$) = $\sum_{i=0}^{|L|-1} L[i]$. By definition, **sum**($L$) = 0 for any empty list, and $\sum_{i=0}^{|L|-1} L[i] = \sum_{i=0}^{-1} L[i] = 0$. Thus **sum**($L$) = $\sum_{i=0}^{|L|-1} L[i]$ as required.

For the inductive step, assume that $P(n)$ holds for some $n \in \mathbb{N}$ and that for all lists $L$ of length $n$, that **sum**($L$) = $\sum_{i=0}^{|L|-1} L[i]$. We prove $P(n + 1)$, that for all lists $L$ of length $n + 1$, that **sum**($L$) = $\sum_{i=0}^{|L|-1} L[i]$. Consider any arbitrary list $L$ of length $n + 1$. By definition, **sum**($L$) in this case is $L$[0] + **sum**($L$[1:]). For notational simplicity, let's let $L' = L$[1:]. The list $L'$ has length $n$, since it consists of all elements of $L$ except for the element at position 0. Thus by our inductive hypothesis, we have that **sum**($L'$) = $\sum_{i=0}^{|L'|-1} L'[i]$. Now, we know that $L'$ consists of all of the elements of $L$ except for the first, so $L'$[i] = $L$[i + 1] for all indices $i$. Therefore, we have **sum**($L'$) = $\sum_{i=0}^{|L'|-1} L[i+1]$. We can then adjust the indices of summation to rewrite $\sum_{i=0}^{|L'|-1} L[i+1] = \sum_{i=1}^{|L'|} L[i]$. Since |$L'$| = |$L$| - 1, we can further simplify this to $\sum_{i=1}^{|L'|} L[i] = \sum_{i=1}^{|L|-1} L[i]$, so **sum**($L'$) = $\sum_{i=1}^{|L|-1} L[i]$. This means that

$$\textbf{sum}(L) = L[0] + \sum_{i=1}^{|L|-1} L[i] = \sum_{i=0}^{|L|-1} L[i].$$

Since our choice of $L$ was arbitrary, this proves that for any list $L$ of length $n + 1$, we have that **sum**($L$) = $\sum_{i=0}^{|L|-1} L[i]$, completing the induction. ■

This proof is at times a bit tricky. The main complexity comes from showing that the sum of the elements of $L$[1:] is the same as the sum of the last $n$ elements of $L$, which we handle by changing our notation in a few places.

We now have a way of proving the correctness of functions that operate over lists! What other functions can we analyze this way? Well, we were able to write one function that operates over sums; could we write one that operates over products? Of course we can! Here's what this function looks like:

```
function product(list L) {
    if |L| = 0:
        return 1.
    else:
        return L[0] × product(L[1:]).
}
```

As with before, we can trace out the execution of this function on a small list; say, one containing the values (2, 3, 5, 7):

```
  product(2, 3, 5, 7)
= 2 × product(3, 5, 7)
= 2 × (3 × product(5, 7))
= 2 × (3 × (5 × product(7)))
= 2 × (3 × (5 × (7 × product())))
= 2 × (3 × (5 × (7 × 1)))
= 2 × (3 × (5 × 7))
= 2 × (3 × 35)
= 2 × 105
= 210
```

Proving this function correct is similar to proving our **sum** function correct, since both of these functions have pretty much the same recursive structure. In the interest of highlighting this similarity, here is the proof that this function does what it's supposed to:

***Theorem:*** For any list $L$, **product**($L$) = $\prod_{i=0}^{|L|-1} L[i]$

**Proof**: By induction. Let $P(n)$ be defined as follows:

$P(n)$ = "For any list $L$ of length $n$, **product**($L$) = $\prod_{i=0}^{|L|-1} L[i]$.

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction. As our base case, we prove $P(0)$, that for any list $L$ of length 0, that **product**($L$) = $\prod_{i=0}^{|L|-1} L[i]$. By definition, **product**($L$) = 1 for any empty list, and $\prod_{i=0}^{|L|-1} L[i] = \prod_{i=0}^{-1} L[i] = 1$. Thus **product**($L$) = $\prod_{i=0}^{|L|-1} L[i]$ as required.

For the inductive step, assume that $P(n)$ holds for some $n \in \mathbb{N}$ and that for all lists $L$ of length $n$, that **product**($L$) = $\prod_{i=0}^{|L|-1} L[i]$. We prove $P(n + 1)$, that for all lists $L$ of length $n + 1$, that **product**($L$) = $\prod_{i=0}^{|L|-1} L[i]$. Consider any arbitrary list $L$ of length $n + 1$. By definition, **product**($L$) in this case is $L[0]$ × **product**($L[1:]$). For notational simplicity, let's let $L' = L[1:]$. The list $L'$ has length $n$, since it consists of all elements of $L$ except for the element at position 0. Thus by our inductive hypothesis, we have that **product**($L'$) = $\prod_{i=0}^{|L'|-1} L'[i]$. Now, we know that $L'$ consists of all of the elements of $L$ except for the first, so $L'[i] = L[i + 1]$ for all indices $i$. Therefore, we have **product**(L') = $\prod_{i=0}^{|L'|-1} L[i+1]$. We can then adjust the indices of the product to rewrite $\prod_{i=0}^{|L'|-1} L[i+1] = \prod_{i=1}^{|L'|} L[i]$. Since $|L'| = |L| - 1$, we can further simplify this to $\prod_{i=1}^{|L'|} L[i] = \prod_{i=1}^{|L|-1} L[i]$, so **product**($L'$) = $\prod_{i=1}^{|L|-1} L[i]$. This means that

$$\textbf{product}(L) = L[0] \times \prod_{i=1}^{|L|-1} L[i] = \prod_{i=0}^{|L|-1} L[i]$$

Since our choice of $L$ was arbitrary, this proves that for any list $L$ of length $n + 1$, we have that **product**($L$) = $\prod_{i=0}^{|L|-1} L[i]$, completing the induction. ■

The fact that these proofs look very similar is not a coincidence; we'll soon investigate exactly why this is.

Let's do one more example. Suppose that we have a list of real numbers and want to return the maximum value contained in the list. For example, max(1, 2, 3) = 3 and max(π, e) = π. For consistency, we'll define that the maximum value of an empty list is -∞. We can write a function that computes the maximum value of a list as follows:

```
function listMax(list L) {
    if |L| = 0:
        return -∞.
    else:
        return max(L[0], listMax(L[1:])).
}
```

This looks surprisingly similar to the two functions we've just written. If we trace out its execution, it ends up behaving almost identically to what we had before:

```
  listMax(2, 3, 1, 4)
= max(2, listMax(3, 1, 4))
= max(2, max(3, listMax(1, 4)))
= max(2, max(3, max(1, listMax(4))))
= max(2, max(3, max(1, max(4, listMax()))))
= max(2, max(3, max(1, max(4, -∞))))
= max(2, max(3, max(1, 4)))
= max(2, max(3, 4))
= max(2, 4)
= 4
```

Proving that this function correct is quite easy, given that we've essentially written this same proof twice in the previous section!

***Theorem:*** For any list $L$, **listMax**($L$) = max{$L$[0], $L$[1], …, $L$[ |$L$| - 1]}

***Proof:*** By induction. Let $P(n)$ be defined as follows:

$P(n)$ = "For any list $L$ of length $n$, **listMax**($L$) = max{$L$[0], …, $L$[ |$L$| - 1]}.

We prove $P(n)$ is true for all $n \in \mathbb{N}$. As our base case, we prove $P(0)$, that for any list $L$ of length 0, that **listMax**($L$) = max{$L$[0], …, $L$[ |$L$| - 1]}. By definition, **listMax**($L$) = -∞ for any empty list, and max{$L$[0], …, $L$[ |$L$| - 1] } = max{ } = -∞. Thus **listMax**($L$) = max{$L$[0], …, $L$[ |$L$| - 1]} as required.

For the inductive step, assume that $P(n)$ holds for some $n \in \mathbb{N}$ and that for all lists $L$ of length $n$, that **listMax**($L$) = max{$L$[0], …, $L$[ |$L$| - 1]}. We prove $P(n + 1)$, that for any list $L$ of length $n + 1$, that **listMax**($L$) = max{$L$[0], …, $L$[ |$L$| - 1]}. Consider any arbitrary list $L$ of length $n + 1$. **listMax**($L$) in this case is max($L$[0], **listMax**($L$[1:]). For notational simplicity, let's let $L'$ = $L$[1:]. The list $L'$ has length $n$, since it consists of all elements of $L$ except for the element at position 0. Thus by our inductive hypothesis, we have that **listMax**($L'$) = max{$L'$[0], …, $L'$[ |$L'$| - 1] }. Now, we know that $L'$ consists of all of the elements of $L$ except for the first, so $L'[i] = L[i + 1]$ for all indices $i$. Therefore, we have **listMax**($L'$) = max{$L$[1], …, $L$[ |$L$| - 1] }. This means

$$\begin{aligned}\textbf{listMax}(L) &= \max(L[0], \max\{L[1], \ldots, L[\,|L| - 1]\,\}) \\ &= \max\{L[0], \ldots, L[|L| - 1]\}\end{aligned}$$

Since our choice of $L$ was arbitrary, this proves that for any list $L$ of length $n + 1$, we have that **listMax**($L$) = max{$L$[0], …, $L$[|$L$| - 1]}, completing the induction. ■

### 3.3.1 Monoids and Folds ★

We've just written three different functions, each of which computes a different property of a list, but which each have almost the exact same structure and same proof of correctness. At this point you should start to be curious if there is something more general at play here.

It turns out that the reason that these three functions look so similar is that they are all special cases of a more general function. To motivate what this function is, we will need to take a closer look at exactly what these functions are doing. Let's look at the base cases of these recursive functions. In the case of `sum`, the base case is 0. In the case of `product`, the base case is 1. For `listMax`, the base case is -∞. These values are not chosen arbitrarily; they each have very special properties. Notice for any $a$, we have

$$0 + a = a + 0 = a$$

$$1 \times a = a \times 1 = a$$

$$\max(-\infty, a) = \max(a, -\infty) = a$$

In other words, the number 0 is the ***identity element*** for addition, the number 1 is the identity element for multiplication, and the value -∞ is the identity element for max. More formally, given some binary operation ★, an identity element for ★ is some value $e$ such that for any $a$, we have that $a \bigstar e = e \bigstar a = a$. Not all binary operations necessarily have an identity element, though many do.

There is one other important property of addition, multiplication, and max that makes the above proof work – namely, all three operations are ***associative***. That is, for any $a$, $b$, and $c$, we have that

$$a + (b + c) = (a + b) + c$$

$$a \times (b \times c) = (a \times b) \times c$$

$$\max(a, \max(b, c)) = \max(\max(a, b), c)$$

More formally, a binary operation ★ is associative if for any $a$, $b$, and $c$, $(a \bigstar b) \bigstar c = a \bigstar (b \bigstar c)$. As a result, when writing out an associative operation as applied to a list of values, it doesn't matter how we parenthesize it; $(a \bigstar b) \bigstar (c \bigstar (d \bigstar e)) = a \bigstar (b \bigstar (c \bigstar (d \bigstar e)))$. In fact, we can leave out the parentheses and just write $a \bigstar b \bigstar c \bigstar d \bigstar e$, since any parenthesization of this expression yields the same value.

Binary operations that are associative and which have an identity element are extremely important in computer science. Specifically, we call them *monoids*:

> A ***monoid*** is a binary operation ★ that is associative and that has an identity element.

Addition, multiplication, and maximum are all examples of monoids, though many operations are not monoids. For example, subtraction is not a monoid because it is not associative; specifically, $1 - (2 - 3) = 2$, while $(1 - 2) - 3 = -4$. However, the set union operation is a monoid, since it is associative ($(A \cup B) \cup C = A \cup (B \cup C)$) and has an identity element ($A \cup \varnothing = \varnothing \cup A = A$).

Let's introduce one new piece of notation. When dealing with sums of multiple values, we introduced Σ notation to condense the sums into more workable forms. Similarly, when dealing with products, we introduced Π notation. Let's generalize this notation a bit more. Suppose that we have a sequence $x_0$, $x_1$, …, $x_{n-1}$ of values and want to compute $x_0 \bigstar x_1 \bigstar \ldots \bigstar x_{n-1}$. We can write this out as

$$\bigstar_{i=0}^{n-1} x_i$$

We can formally define what this means inductively:

- $\bigstar_{i=m}^{n} x_i = e$ if $n < m$. Here, $e$ is the identity element of the monoid. In other words, if we apply the operation zero times, then we just end up with the identity element. We define the empty sum as 0 and the empty product as 1, and this definition is just an extension of what we have before.
- If $m \le n$, then $\bigstar_{i=m}^{n} x_i = x_m \bigstar \left( \bigstar_{i=m+1}^{n} x_i \right)$. That is, if we have a whole bunch of terms that we need to apply the operator to, we can just "peel off" the first term, apply the operation to the rest of the terms, and then combine that result with the first value.

Given this new terminology, let's review the three functions that we wrote previously:

```
function sum(list L) {
    if |L| = 0:
        return 0.
    else:
        return L[0] + sum(L[1:]).
}
function product(list L) {
    if |L| = 0:
        return 1.
    else:
        return L[0] × product(L[1:]).
}
function listMax(list L) {
    if |L| = 0:
        return -∞.
    else:
        return max(L[0], listMax(L[1:])).
}
```

We can write this more generically in terms of monoids as follows:

```
function fold(list L) {
    if |L| = 0:
        return e.
    else:
        return L[0] ⋆ fold(L[1:]).
}
```

Here, as before, $e$ is the identity element of the monoid. This function is sometimes called a **fold**, **reduce**, or **accumulate** function and is a staple in most functional programming languages. Assuming that ★ is a monoid and that $e$ is its identity element, we can formally prove that this function is correct by generalizing our previous proofs:

***Theorem:*** For any monoid ★ and list $L$, **fold**($L$) = $\bigstar_{i=0}^{|L|-1} L[i]$

***Proof:*** By induction. Let ★ be any monoid with identity element $e$ and define $P(n)$ as follows:

$P(n)$ = "For any list $L$ of length n, **fold**(L) = $\bigstar_{i=0}^{|L|-1} L[i]$.

We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction. As our base case, we prove $P(0)$, that for any list $L$ of length 0, that **fold**($L$) = $\bigstar_{i=0}^{|L|-1} L[i]$. Note that **fold**($L$) = $e$ for any empty list, and we have that $\bigstar_{i=0}^{|L|-1} L[i] = \bigstar_{i=0}^{-1} L[i] = e$. Thus **fold**($L$) = $\bigstar_{i=0}^{|L|-1} L[i]$ as required.

For the inductive step, assume that $P(n)$ holds for some $n \in$ □ and that for all lists $L$ of length $n$, that **fold**($L$) = $\bigstar_{i=0}^{|L|-1} L[i]$. We prove $P(n + 1)$, that for all lists $L$ of length $n + 1$, that **fold**($L$) = $\bigstar_{i=0}^{|L|-1} L[i]$. Consider any arbitrary list $L$ of length $n + 1$. By definition, **fold**($L$) in this case is $L$[0] ★ **fold**($L$[1:]). For notational simplicity, let's let $L'$ = $L$[1:]. The list $L'$ has length $n$, since it consists of all elements of $L$ except for the element at position 0. By our inductive hypothesis, we have that **fold**($L'$) = $\bigstar_{i=0}^{|L'|-1} L'[i]$. Now, we know that $L'$ consists of all of the elements of $L$ except for the first, so $L'$[i] = $L$[i + 1] for all indices $i$. Therefore, we have **fold**($L'$) = $\bigstar_{i=0}^{|L'|-1} L[i+1]$. We can then adjust the indices to rewrite $\bigstar_{i=0}^{|L'|-1} L[i+1] = \bigstar_{i=1}^{|L'|} L[i]$. Since |$L'$| = |$L$| - 1, we can simplify this to $\bigstar_{i=1}^{|L'|} L[i] = \bigstar_{i=1}^{|L|-1} L[i]$, so **fold**($L'$) = $\bigstar_{i=1}^{|L|-1} L[i]$. This means that

$$\textbf{fold}(L) = L[0] \bigstar \bigstar_{i=1}^{|L|-1} L[i] = \bigstar_{i=0}^{|L|-1} L[i].$$

Since our choice of $L$ was arbitrary, this proves that for any list $L$ of length $n + 1$, we have that **fold**($L$) = $\bigstar_{i=0}^{|L|-1} L[i]$, completing the induction. ■

What we have just done is an excellent example of mathematics in action. We started with a collection of objects that shared some similar properties (in our case, recursive functions over list), then noticed that there was something similar connecting all of them. We then defined a new framework that captured all of our existing objects as special cases, then proved the result for the general result as a whole.

The beauty of what we've just done is that we immediately know that both of the following functions will work correctly:

```
function union(list L) {
    if |L| = 0:
       return ∅.
    else:
       return L[0] ∪ union(L[1:]).
}
```

```
function concatenateStrings(list L) {
    if |L| = 0:
       return "".
    else:
       return concat(L[0], concatenateStrings(L[1:])).
}
```

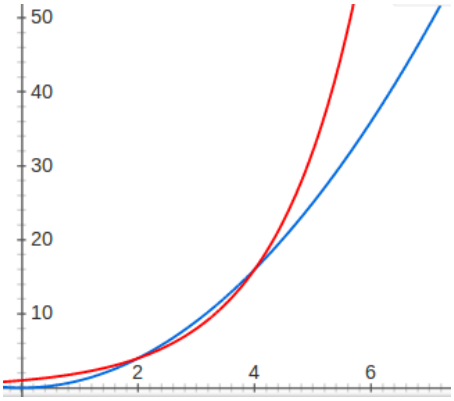
If we want to prove correctness of these functions, we don't have to do a full inductive proof. Instead, we can just prove that set union and string concatenation are monoids. I'll leave those proofs as exercises to the interested reader.

# 3.4 Variants on Induction

We have seen induction can be useful when proving properties that hold for all natural numbers. However, in many cases we are interested in proving properties of only a subset of the natural numbers – say, even numbers, powers of two, odd numbers, etc. In that case, it may be useful to use a variant of mathematical induction that captures the essential ideas of induction, but in a slightly different setting.

## 3.4.1 Starting Induction Later

Let's consider a simple question: what's bigger, $n^2$ or $2^n$? If you have played around with these functions before, then your intuition probably tells you that $2^n$ grows much, much more quickly than $n^2$. For example, when $n = 15$, $n^2 = 225$, while $2^n = 32{,}768$. However, if we plot the graphs of the two functions for small values, we see some interesting trends:




| n | $n^2$ | $2^n$ |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 2 |
| 2 | 4 | 4 |
| 3 | 9 | 8 |
| 4 | 16 | 16 |
| 5 | 25 | 32 |
| 6 | 36 | 64 |

*Image from Google Graphs*

As you can see, the two functions jockey for position. Initially, $2^n$ is larger, but then $n^2$ overtakes it. Once $n = 5$, $2^n$ becomes larger. But given that these functions have fluctuated before, is it possible that $n^2$ eventually catches up to and overtakes $2^n$?

It turns out that the answer is no. We can prove this in many ways, particularly differential calculus. However, we can also prove this result by induction. Namely, we will want to prove the following:

> ***Theorem***: For all $n \in$ □ where $n \geq 5$, $n^2 < 2^n$.

We won't concern ourselves with real numbers for now. Instead, we'll focus on natural numbers.

How exactly would we prove this? If we wanted to proceed by induction using the techniques we've seen so far, we would have to prove some claim about every natural number, not just the natural numbers greater than or equal to five. That said, it turns out that we can use our normal version of induction to prove this result. We will do this by being very clever with how we choose our property $P(n)$. Specifically, what if we choose this property:

$$P(n) \equiv (n + 5)^2 < 2^{(n + 5)}$$

If we can prove that this claim is true for all $n \in \mathbb{N}$, then we will have proven the theorem. The reason for this is that if $n \geq 5$, then we know that $n - 5$ must be a natural number. Consequently, the fact that $P(n - 5)$ is true implies that $((n - 5) + 5)^2 < 2^{((n - 5) + 5)}$, which simplifies down to $n^2 < 2^n$, as required. Using this approach, our proof proceeds as follows:

***Proof:*** By induction. Let $P(n) \equiv (n + 5)^2 < 2^{(n + 5)}$. We will prove that $P(n)$ holds for all $n \in \mathbb{N}$. From this, we can conclude that for any $n \in \mathbb{N}$ with $n \geq 5$ that $(n + 5)^2 < 2^{(n + 5)}$. This holds because for any natural number $n \geq 5$, we have that $n - 5 \in \mathbb{N}$. $P(n - 5)$ then implies that $n^2 < 2^n$.

For our base case, we prove $P(0)$, that $5^2 < 2^5$. Since $5^2 = 25$ and $2^5 = 32$, this claim is true. For the inductive step, assume for some $n \in \mathbb{N}$ that $P(n)$ holds and $(n + 5)^2 < 2^{(n + 5)}$. We will prove $P(n + 1)$, that $((n + 5) + 1)^2 < 2^{((n + 5) + 1)}$. To see this, note that

$$\begin{aligned}
&((n + 5) + 1)^2 \\
&= (n + 5)^2 + 2(n + 5) + 1 && \text{(distributing)} \\
&< 2^{n+5} + 2(n + 5) + 1 && \text{(by the inductive hypothesis)} \\
&< 2^{n+5} + 2(n + 5) + 5 && \text{(since } 1 < 5\text{)} \\
&\leq 2^{n+5} + 2(n + 5) + n + 5 && \text{(since } 0 \leq n\text{)} \\
&= 2^{n+5} + 3(n + 5) && \text{(collecting terms)} \\
&< 2^{n+5} + 5(n + 5) && \text{(since } 3 < 5\text{)} \\
&\leq 2^{n+5} + (n + 5)(n + 5) && \text{(since } 0 \leq n\text{)} \\
&= 2^{n+5} + (n + 5)^2 && \text{(simplifying)} \\
&< 2^{n+5} + 2^{n+5} && \text{(by the inductive hypothesis)} \\
&= 2(2^{n+5}) && \text{(simplifying)} \\
&= 2^{((n + 5) + 1)} && \text{(by powers of exponents)}
\end{aligned}$$

Thus $((n + 5) + 1)^2 < 2^{((n + 5) + 1)}$, so $P(n + 1)$ holds, completing the induction. ■

This proof is interesting for a few reasons. First, it shows that we can use induction to reason about properties of numbers larger than a certain size, though we have to be careful with how we phrase it. Second, it shows a style of proof that we have not seen before. To prove an inequality holds between two quantities, we can often expand out the inequality across multiple steps, at each point showing one smaller piece of the inequality. Since inequalities are transitive, the net result of these inequalities gives us the result that we want.

Let's try another example of a problem like this, this time using two other functions that grow quite rapidly: $2^n$ and $n!$. $n!$ is an *extremely* fast-growing function that dwarfs the comparatively well-behaved $2^n$. For example, for $n = 10$, $2^{10} = 1{,}024$, but $n! = 3{,}628{,}800$. However, for small values, we see the two functions vying for greatness:

| $n$ | $2^n$ | $n!$ |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 3 | 8 | 6 |
| 4 | 16 | 24 |
| 5 | 32 | 120 |

From the looks of this table, it seems that as soon as $n \geq 4$, $n!$ ends up overtaking $2^n$. Does $n!$ continue to dominate from this point forward? Or does $2^n$ ever catch up?

Well, considering that we're about to prove the following theorem:

***Theorem***: For any $n \in \mathbb{N}$ with $n \geq 4$, $2^n < n!$.

it looks like $n!$ is going to dominate from this point forward. The proof of this theorem is structurally quite similar to what we had before; the main changes are the fact that we're now starting from four, not five, and that our functions are different.

***Proof:*** By induction. Let $P(n) \equiv 2^{(n+4)} < (n+4)!$. We will prove that $P(n)$ holds for all $n \in \square$. From this, we can conclude that for any $n \in \square$ with $n \geq 4$ that $2^n < n!$. This holds because for any natural number $n \geq 4$, we have that $n - 4 \in \square$. $P(n-4)$ then implies that $n^2 < 2^n$.

For our base case, we prove P(0), that $2^4 < 4!$. To see this, note that $2^4 = 16$, while $4! = 4 \times 3 \times 2 \times 1 = 24$. For the inductive step, assume that for some $n \in \square$ that $P(n)$ holds and $2^{(n+4)} < (n+4)!$. We will prove that $P(n+1)$ holds, that $2^{((n+4)+1)} < ((n+4)+1)!$. To see this, note that

$$
\begin{aligned}
& 2^{((n+4)+1)} \\
&= 2(2^{(n+4)}) && \text{(using powers of exponents)} \\
&< 2(n+4)! && \text{(by the inductive hypothesis)} \\
&< 5(n+4)! && \text{(since } 2 < 5\text{)} \\
&\leq (n+5)(n+4)! && \text{(since } 0 \leq n\text{)} \\
&= (n+5)! && \text{(by definition of factorial)} \\
&= ((n+4)+1)!
\end{aligned}
$$

Thus $2^{((n+4)+1)} < ((n+4)+1)!$, so $P(n+1)$ holds, completing the induction. ■

The two proofs that we have just completed use induction, but are tricky to work with. In particular, we want to prove a result about numbers greater than some specific threshold, but our proof instead works by showing a result for all numbers, using addition to shift everything over the appropriate number of steps.

Let's consider an alternative style of proof. Suppose that we want to repeat our previous proof (the one about $n!$ and $2^n$). Can we just do the following:

- As our base case, prove that $2^4 < 4!$.
- As our inductive step, assume that for some $n \geq 4$, $2^n < n!$ and prove that $2^{n+1} < (n + 1)!$.

In other words, this would be a normal inductive proof, except that we have shifted the base case up from 0 to 4, and now make an extra assumption during our inductive hypothesis that $n \geq 4$. Otherwise, the proof proceeds as usual.

Of course, we're not even sure that it's mathematically legal to do this. Ignoring that (critical!) detail for now, though, let's see what the proof would look like were we allowed to write it. It turns out that the proof is much shorter than before and a lot easier to read:

***Proof:*** By induction. Let $P(n) \equiv 2^n < n!$. We will prove that $P(n)$ holds for all $n \in \mathbb{N}$ with $n \geq 4$ by induction.

For our base case, we prove P(4), that $2^4 < 4!$. To see this, note that $2^4 = 16$, while $4! = 4 \times 3 \times 2 \times 1 = 24$.

For the inductive step, assume that for some $n \in \mathbb{N}$ with $n \geq 4$ that $P(n)$ holds and $2^n < n!$. We will prove that $P(n + 1)$ holds, meaning that $2^{n+1} < (n + 1)!$. To see this, note that

$$
\begin{aligned}
& 2^{n+1} \\
&= 2(2^n) && \text{(by properties of exponents)} \\
&< 2(n!) && \text{(by our inductive hypothesis)} \\
&< (n + 1)(n!) && \text{(since } 2 < 4 \leq n < n + 1\text{)} \\
&= (n + 1)!
\end{aligned}
$$

Thus $2^{n+1} < (n + 1)!$, so $P(n + 1)$ holds, which completes the induction. ■

Wow! That's much cleaner, more succinct, and more clearly explains what's going on. The key step in this proof is the fact that we know that $2 < n + 1$. This gives a good justification as to why $2^n < n!$ once $n$ gets to four – from that point forward, going from $2^n$ to $2^{n+1}$ doubles the previous value, but going from $n!$ to $(n + 1)!$ increases the value by a factor of $n + 1$.

#### 3.4.1.1 Why We Can Start Later ★

How do we know that the above line of reasoning – starting induction later on – is even a valid mathematical proof? The only reason that we know induction works is because it was specifically sanctioned as a mathematically valid form of reasoning. When we start making changes to induction, we can't necessarily guarantee that the resulting form of reasoning is sound. We will need to justify why the previous proof technique is legitimate before we start using it any further.

If you'll recall from Chapter Two, we proved that proof by contrapositive was legal by using proof by contradiction as a starting point. That is, we used one type of proof to show that some other type of proof was possible. We will now use normal mathematical induction to justify the above variant of induction, where we start off our induction from a value other than zero.

We will specifically prove the following theorem:

***Theorem:*** Let $P(n)$ be a property that applies to natural numbers and let $k$ be a natural number. If the following are true:

$P(k)$ is true
For any $n \in \mathbb{N}$ with $n \geq k$, $P(n) \rightarrow P(n + 1)$

Then for any $n \in \mathbb{N}$ with $n \geq k$, $P(n)$ is true.

Compare this to the definition of the principle of mathematical induction from earlier in the chapter:

Let $P(n)$ be a property that applies to natural numbers. If the following are true:

$P(0)$ is true
For any $n \in \mathbb{N}$, $P(n) \rightarrow P(n + 1)$

Then for any $n \in \mathbb{N}$, $P(n)$ is true.

Our goal will be to show that our initial definition of mathematical induction will allow us to prove that the above theorem is true.

In doing so it is critical to understand exactly what it is that we are trying to prove. Specifically, we want to show the following:

(Principle of Mathematical Induction) → (Induction Starting at $k$)

In order to do this, we will do the following. We want to show that if the following hold:

- $P(k)$ is true
- For any $n \in \mathbb{N}$ with $n \geq k$, $P(n) \rightarrow P(n + 1)$

Then we can conclude that

- For any $n \in \mathbb{N}$ with $n \geq k$, $P(n)$ holds.

We will do this by means of a very clever trick. Suppose that we could define some *new* property $Q(n)$ using $P(n)$ as a building block. We will choose $Q(n)$ such that it has two special properties:

1. We can prove that $Q(n)$ is true by induction on $n$.
2. If $Q(n)$ is true for all natural numbers, then $P(n)$ is true for all natural numbers greater than or equal to $k$.

These may seem totally arbitrary, but there is a good reason for them. For property (1), it's important to realize that the property $P(n)$ we're trying to reason about looks similar to the sort of property we would try to prove by induction. Unfortunately, since the base case doesn't start at 0, and since the inductive step makes some assumptions about the value of $n$, we can't immediately use induction. If we could somehow adjust $P(n)$ in a way so that it didn't have these two modifications, we could easily prove it by induction. Our choice of $Q(n)$ will be a modified version of $P(n)$ that does just that.

We want property (2) to hold so that we can find a connection between $Q$ and $P$. If we don't have any restrictions on $Q$, then it's irrelevant whether or not we can prove it by induction. The trick will be to make

it so that proving $Q$ is true for all natural numbers shows that $P$ is true for all natural numbers that are greater than or equal to $k$, which is precisely what we want to show.

The good news is that we already have seen two examples of how to build $Q$ from $P$. The idea is actually quite simple – if we want to reason about numbers greater than or equal to $k$, we can just prove properties about natural numbers of the form $n + k$. Since every natural number greater than or equal to $k$ must be $k$ plus some smaller natural number, this means that we will have all of our cases covered. Specifically, we'll define

$$Q(n) \equiv P(n + k)$$

This is fairly abstract, so let's give some examples. Suppose we want to prove, as before, that $n^2 < 2^n$ for $n \geq 5$. In that case, we'd say that $P(n) \equiv n^2 < 2^n$. We then define $Q(n) \equiv P(n + 5)$, which, if we expand it out, says that $Q(n) \equiv (n + 5)^2 < 2^{n+5}$. If you look back at our first proof, this is exactly how we arrived at this result.

Given this setup, we can formally prove the theorem below.

***Theorem:*** Let $P(n)$ be a property that applies to natural numbers and let $k$ be a natural number. If the following are true:

$P(k)$ is true
For any $n \in \mathbb{N}$ with $n \geq k$, $P(n) \rightarrow P(n + 1)$

Then for any $n \in \mathbb{N}$ with $n \geq k$, $P(n)$ is true.

***Proof:*** Consider any property $P(n)$ of the natural numbers for which the above is true. Now, define $Q(n) \equiv P(n + k)$. This proof will work in two parts: first, we will prove, by induction, that $Q(n)$ is true for all $n \in \mathbb{N}$. Next, we will show that this implies that $P(n)$ is true for all $n \in \mathbb{N}$ where $n \geq k$.

First, we prove that $Q(n)$ is true for all $n \in \mathbb{N}$ by induction. As our base case, we prove $Q(0)$, meaning that $P(k)$ holds. By our choice of $P$, we know that this is true. Thus $Q(0)$ holds.

For the inductive step, assume that for some $n \in \mathbb{N}$ that $Q(n)$ is true, meaning that $P(n + k)$ holds. We will prove that $Q(n + 1)$ is true, meaning that $P(n + k + 1)$ holds. Now, since $n \in \mathbb{N}$, we know that $n \geq 0$. Consequently, we know that $n + k \geq k$. Thus by the properties of $P$, we know that $P(n + k)$ implies $P(n + k + 1)$. Since $Q(n + 1) \equiv P(n + k + 1)$, this proves that $Q(n + 1)$ holds, completing the induction. Thus $Q(n)$ is true for all natural numbers $n$.

Now, we use this result to prove that $P(n)$ is true for all natural numbers $n \geq k$. Consider any arbitrary natural number $n \geq k$. Thus $n - k \geq 0$, so $n - k$ is a natural number. Therefore, $Q(n - k)$ holds. Since $(n - k) + k = n$, this means that $P(n)$ holds. Since our choice of $n$ was arbitrary, this shows that $P(n)$ is true for all natural numbers $n \geq k$. ■

This proof is beautiful for several reasons. First, it combines proof by induction, which we've explored extensively in this chapter, with our previous style of proving that general claims are true – choose some arbitrary object, then show that the claim holds for that object. Second, it generalizes the two proofs that we did earlier in this section in a way that allows us to use this new, much more powerful form of induction. From this point forward, we can start off our inductions anywhere in the natural numbers, using the

proof we have just done to conclude that we have proven a result about all natural numbers greater than or equal to some starting value.

### 3.4.2 Fibonacci Induction ★

The *Fibonacci sequence* is a famous mathematical sequence that appears in a surprising number of places. The sequence is defined as follows: the first two terms are 0 and 1, and each successive term is the sum of the two previous terms. For example, the first several terms of the Fibonacci sequence are

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

Formally speaking, we can define the Fibonacci sequence using this beautiful inductive definition:

- $F_0 = 0$
- $F_1 = 1$
- $F_{n+2} = F_n + F_{n+1}$

From our perspective, the Fibonacci sequence is interesting in that it is clearly defined inductively (every term is defined by the two terms that precede it), but we cannot immediately use our principle of induction to prove its properties. Using the forms of induction we've seen so far, we can use knowledge about $n$ to prove properties about $n + 1$. For the Fibonacci sequence, we need to use information about $n$ and $n + 1$ to prove properties about $n + 2$. This slight difference – the fact that we rely on the last *two* terms to reason about the next term – complicates proofs about Fibonacci numbers.

In order to prove properties about the Fibonacci sequence or numbers related to them, we will need a new type of induction specifically suited to working with the Fibonacci sequence. Specifically, we'll try to construct our induction so that it mirrors the shape of the Fibonacci numbers. If we want to prove that some property $P(n)$ holds for the $n$th Fibonacci number, it would make sense to try to prove the following:

- $P(n)$ holds for the zeroth and first Fibonacci numbers.
- If $P(n)$ holds for the $n$th and $(n+1)$st Fibonacci numbers, then it holds for the $(n+2)$nd Fibonacci number.

From this, it should seem reasonably intuitive that we could claim that $P(n)$ holds for all Fibonacci numbers. We could see this because

- $P(0)$ and $P(1)$ hold.
- Because $P(0)$ and $P(1)$ hold, $P(2)$ holds.
- Because $P(1)$ and $P(2)$ hold, $P(3)$ holds.
- Because $P(2)$ and $P(3)$ holds, $P(4)$ holds.
- …

As with our previous result that we can fire off induction from any starting point, we will need to formally prove that this form of induction (which we'll dub *Fibonacci induction*) actually works correctly. Our goal will be to prove this theorem:

***Theorem:*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:

$P(0)$ is true.
$P(1)$ is true.
For any $n \in \mathbb{N}$, $P(n)$ and $P(n + 1) \to P(n + 2)$

Then for any $n \in \mathbb{N}$, $P(n)$ is true.

Again, our starting point is that we know the principle of mathematical induction to be true. We will somehow have to use this as a building block in order to show that the above theorem – which gives rise to its own style of proof – is indeed correct.

Our proof will be similar to the one we did last time. The objective will be to define some new property $Q(n)$ in terms of $P(n)$ such that

- $Q(n)$ can be proven true for all $n \in \mathbb{N}$ using the principle of mathematical induction, and
- If $Q(n)$ is true for all $n \in \mathbb{N}$, then $P(n)$ is true for all $n \in \mathbb{N}$.

The trick will be choosing an appropriate $Q(n)$. To do so, let's review why this style of proof works in the first place. If you look at the informal logic we used on the previous page, you'll note that we started with $P(0)$ and $P(1)$ and used this to derive $P(2)$. Then, from $P(1)$ and $P(2)$, we derived $P(3)$. From $P(2)$ and $P(3)$, we derived $P(4)$. If you'll notice, each step in this proof works by assuming that $P(n)$ is true for the last two choices of $n$, then using that to prove that $P(n)$ holds for the next choice of $n$. This suggests that perhaps we'll want our choice of $Q(n)$ to encode the idea that two adjacent values of $P(n)$ still holds. For example, what if we try choosing $Q(n)$ as

$$Q(n) \equiv P(n) \text{ and } P(n + 1)$$

Could we use this to prove that $Q(n)$ is true for all $n \in \mathbb{N}$ by induction? Well, we'd first have to prove that $Q(0)$ holds, meaning that $P(0)$ and $P(1)$ are true. Fortunately, we already know that – this is one of the two properties we know of $P(n)$. Great!

So could we prove the inductive step? Our goal here would be to show that if $Q(n)$ holds (meaning that $P(n)$ and $P(n + 1)$ are true), then $Q(n + 1)$ holds (meaning that $P(n + 1)$ and $P(n + 2)$ are true). Half of this is easy – if we're already assuming that $P(n + 1)$ is true to begin with, then we don't need to prove $P(n + 1)$ is true again. However, we do need to prove that $P(n + 2)$ is true. But we've chosen $P(n)$ such that $P(n)$ and $P(n + 1)$ collectively imply $P(n + 2)$. This means that if we know that $P(n)$ and $P(n + 1)$ are true, we can easily get that $P(n + 1)$ and $P(n + 2)$ are true. In other words, if $Q(n)$ is true, then $Q(n + 1)$ must be true as well. Excellent!

We can formalize this intuition below in the following proof:

***Theorem:*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:

$P(0)$ is true.
$P(1)$ is true.
For any $n \in \mathbb{N}$, $P(n)$ and $P(n + 1) \to P(n + 2)$

Then for any $n \in \mathbb{N}$, $P(n)$ is true.

***Proof:*** Let $P(n)$ be an arbitrary property with the above set of traits. Let $Q(n) \equiv$ "$P(n)$ and $P(n + 1)$." We will prove that $Q(n)$ is true for all $n \in \mathbb{N}$ by induction on $n$. Once we have proven this, we will show that if $Q(n)$ is true for all $n \in \mathbb{N}$, it must be the case that $P(n)$ is true for all $n \in \mathbb{N}$.

To see that $Q(n)$ is true for all $n \in \mathbb{N}$, we proceed by induction. First, we prove that $Q(0)$ is true; namely, that $P(0)$ and $P(1)$ are true. By our choice of $P(n)$, we know these properties are true, so $Q(0)$ holds.

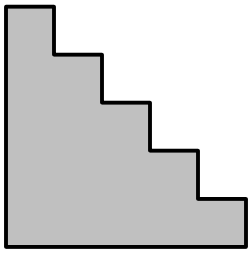
For the inductive step, assume that for some $n \in \mathbb{N}$ that $Q(n)$ is true, meaning that $P(n)$ and $P(n + 1)$ are true. We will prove $Q(n + 1)$ is true, meaning that $P(n + 1)$ and $P(n + 2)$ are true. By our inductive hypothesis, we already know that $P(n + 1)$ is true, so we just need to show that $P(n + 2)$ is true as well. By our choice of $P(n)$, we know that since $P(n)$ and $P(n + 1)$ are true, we have that $P(n + 2)$ is true as well. Thus we have that $P(n + 1)$ and $P(n + 2)$ hold, so $Q(n + 1)$ holds as well, completing the induction.

Finally, we need to show that because $Q(n)$ is true for all $n \in \mathbb{N}$, we know that $P(n)$ is true for all $n \in \mathbb{N}$. To see this, note that for any $n \in \mathbb{N}$, we know that $Q(n)$ is true, so $P(n)$ and $P(n + 1)$ are true. Ignoring the extra detail that $P(n + 1)$ is true, we now have that $P(n)$ is true as well. Since our choice of $n$ was arbitrary, this proves that $P(n)$ holds for all $n \in \mathbb{N}$. ■

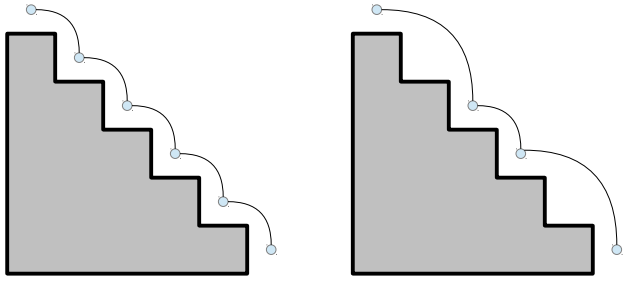
Cool! We now have a proof technique we can use to reason about the Fibonacci numbers. Let's not just leave this sitting on the shelf; instead, let's go and use it to prove some interesting properties about Fibonacci numbers and related problems!

### 3.4.2.1 Climbing Down Stairs

Let's start off by tackling a simple problem that has a rather surprising solution. Let's suppose that you're standing at the top of a staircase, like this one:

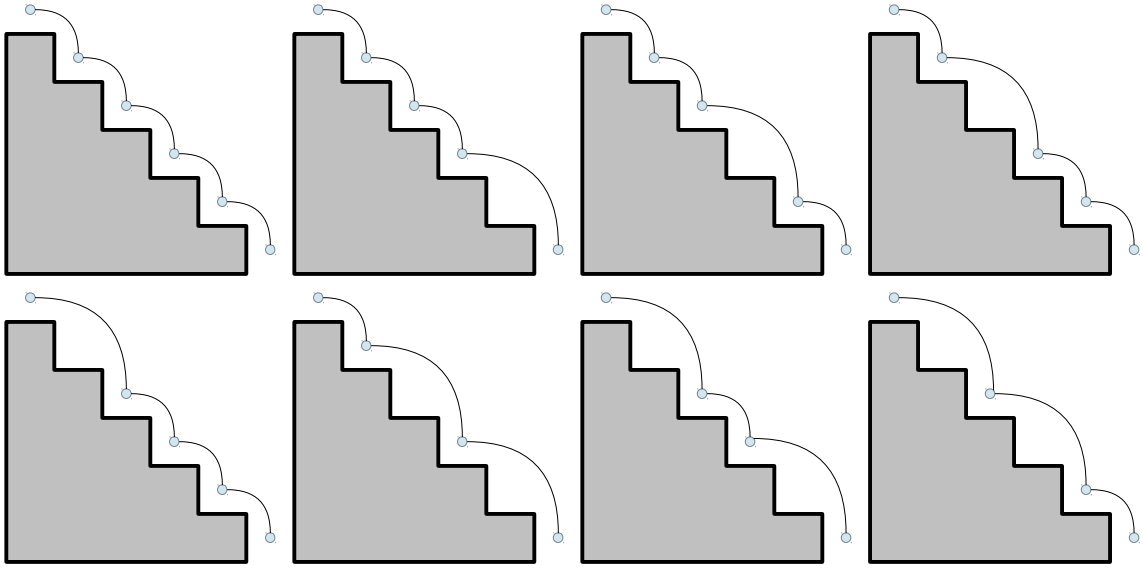


You want to get down to the base of the staircase. In doing so, you can walk down using steps of size one or steps of size two (but not any more – you don't want to come tumbling down the stairs!) For example, here are two different paths you could take down the above staircase:



Now, the question – how many paths down a staircase of size *n* can you take?

If you take any reasonable-sized staircase, you'll find that there can be a *lot* of paths down. For example, if you have the above staircase, there are eight different paths down:



The number of paths just keeps getting bigger and bigger as the staircase gets higher and higher. Can we find a nice expression that will tell us exactly how many paths down the stairs there are?

When faced with a problem like this, one reasonable approach would be to list off all the paths that we can find, then check whether or not we can find a pattern in how they're constructed. For the staircase with five stairs, we can write out those paths as a sequence of 1s and 2s,

- 1 1 1 1 1
- 1 1 1 2
- 1 1 2 1
- 1 2 1 1
- 2 1 1 1
- 2 2 1
- 2 1 2
- 1 2 2

There are a lot of patterns we can exploit here. Notice, for example, that the first path (1 1 1 1 1) is just all 1s. The next four paths consist of all possible permutations of three 1s and one 2. The last three paths consist of all possible permutations of two 2s and a 1. If we counted up how many ways there were to set up these sorts of permutations, we could arrive at our answer.

While the above approach works, it's a bit complicated and we might have better luck looking for other patterns. What if we sorted the above sequences lexicographically (the way that you would order words in a dictionary?) In that case, we get the following sequence of paths:

- 1 1 1 1 1
- 1 1 1 2
- 1 1 2 1
- 1 2 1 1
- 1 2 2
- 2 1 1 1
- 2 1 2
- 2 2 1

Now, we can start to see a different structure emerging. Notice that for each of the paths that start with a 1, the structure of the path is a 1 followed by some path from the fourth step all the way back down to the bottom of the stairs. Each of the paths that start with a 2 have the form of a path starting with a step of size 2, followed by a path from the third stair all the way down. Intuitively, this makes sense – to get down, you either take a step down one stair and then some path back down from there, or you take a step down two stairs and then some path back down from there.

This observation can actually be used to develop a way of computing the number of paths down the stairs. Assuming that we start out sufficiently "high enough" on the stairs that there's room to take steps of size one and size two, then the total number of paths back down the stairs must be equal to the number of paths that start with a step of size one, plus the number of paths that start with a step of size two. Since each path starting with a step of size one initially takes us from stair $n$ to stair $n - 1$, and each path starting with a step of size two initially takes us from stair $n$ to stair $n - 2$, this means that the number of paths down the stairs from stair $n$ is given by the number of paths down from stair $n - 1$, plus the number of paths down from stair $n - 2$.

What this says is that if we're sufficiently high up on the stairs that we can take steps of size 1 and size 2, we can determine the number of paths down based on the number of paths from the stairs one and two steps below us. But what if we're very near the bottom of the staircase and can't do this? Well, the only way that we couldn't take a step of size 1 and a step of size 2 would be if we are standing on stair 1, or we are at the bottom of the staircase. If we are at stair 1, then there is exactly one path down – just take a step of size one down to the bottom.

But what if we are at the base of the staircase? How many paths are there now? This is a subtle but important point. Initially, we might say that there are zero paths, since you can't take any steps here. But this would be mathematically misleading. If there are indeed zero paths once you're standing at the base of the staircase, then it would mean that there is no way to get to the bottom of the staircase once you're already there. This seems suspicious. Ask yourself the following question – can you get to where you are sitting right now from your current location? You'd probably think "yes – I'm already there!" In fact, for this very reason, it would be inappropriate to say that there are zero paths down the stairs from the bottom of the staircase. Rather, there is exactly one path, namely, not moving at all.

The thing to remember here is that there is a difference between "there are no paths" and "the only path is the empty path." When dealing with problems like these, it is critical to remember to maintain a distinction between "unsolvable" and "trivially solvable." Many problems have silly solutions for small cases, but those solutions are still indeed solutions!

Okay – at this point, we have a nice intuition for the number of paths down:

- There is exactly one path down from a staircase of height 0 or height 1.
- For staircases with two or more steps, the number of paths down is the sum of the number of paths down for a staircase of one fewer step plus the number of paths down for a staircase of two fewer steps.

Let's try to make this a bit more formal. Let's define a sequence $S_n$ representing the number of paths down from a staircase of height $n$. Translating the above intuition into something a bit more mathematically rigorous, we get that

- $S_0 = S_1 = 1$.
- $S_{n+2} = S_n + S_{n+1}$.

Now that we have this recurrence, we can start evaluating a few terms from it to see if we can recognize it from somewhere. If we start expanding this out, we get the sequence

$$1, 1, 2, 3, 5, 8, 13, 21, \ldots$$

This should look very familiar – it's the Fibonacci sequence, shifted over by one term! Now that is indeed surprising. The problem we described – walking down a staircase – superficially bears no resemblance at all to the Fibonacci sequence. This isn't particularly unusual, though, since the Fibonacci sequence tends to arise in all sorts of surprising contexts.

Are we satisfied that the number of paths down from a staircase of height $n$ is the $(n + 1)$st Fibonacci number? We've arrived at our result intuitively, but we haven't actually proven anything yet. To wrap up this problem, and to put our new Fibonacci induction proof technique to use, let's formally establish the above result:

***Theorem:*** On a staircase of $n$ stairs, there are $F_{n+1}$ paths from the top of the staircase down to the bottom using step sizes of 1 and 2.

***Proof:*** By induction. Let $P(n)$ be "On a staircase of $n$ stairs, there are $F_{n+1}$ paths from the top of the staircase down to the bottom using step sizes of 1 and 2." We will prove that $P(n)$ is true for all $n \in \mathbb{N}$.

As our base cases, we prove that $P(0)$ and $P(1)$ are true; that is, there are $F_1$ and $F_2$ paths down staircases of heights 0 and 1, respectively. In the case of a staircase of height 0, there is exactly one path down the staircase, namely, the path of no steps. Since $F_1 = 1$, the claim holds for $P(0)$. For a staircase of height 1, there is exactly one path down, which is to take a step of size one. Since $F_2 = 1$, the claim holds for $P(1)$.

For the inductive step, assume that for some $n \in \mathbb{N}$, that $P(n)$ and $P(n + 1)$ hold and that the number of paths down staircases of heights $n$ and $n + 1$ using only step sizes of 1 and 2 is $F_{n+1}$ and $F_{n+2}$, respectively. We want to prove $P(n + 2)$, namely that the number of paths down a staircase of height $n + 2$ using steps of size 1 and 2 is $F_{n+3}$. To see this, consider any path down such a staircase. This path must either begin with a step of size 1, in which case the path is formed by taking a path down a staircase of size $n + 1$ and extending it by one step, or it must begin with a step of size 2, in which case the path is formed by taking a path down a staircase of size $n$ and extending it by

one step. Consequently, the number of paths down the staircase of height $n + 2$ is given by the number of paths down staircases of heights $n$ and $n + 1$. By our inductive hypothesis, these numbers are $F_{n+1}$ and $F_{n+2}$. Consequently, the number of paths down is $F_{n+1} + F_{n+2} = F_{n+3}$, as required. Thus $P(n + 2)$ holds, completing the induction. ■

### 3.4.2.2 Computing Fibonacci Numbers

The Fibonacci numbers are often introduced as a simple function that can be computed easily with recursion. Typically, the recursive function is presented as follows:

```
int fib(int n) {
   if (n == 0) return 0;
   if (n == 1) return 1;
   return fib(n - 1) + fib(n - 2);
}
```

We can prove that this function is correct by induction on $n$:

***Theorem:*** For any $n \in \mathbb{N}$, **fib**$(n) = F_n$.

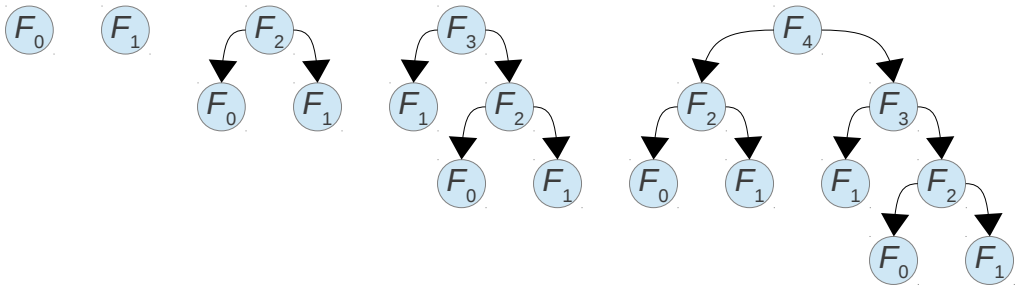
***Proof:*** By induction on $n$. Let $P(n) \equiv$ "**fib**$(n) = F_n$." We prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction.

For our base case, we prove $P(0)$ and $P(1)$; namely, that **fib**$(0) = F_0$ and that **fib**$(1) = F_1$. To see this, note that by construction, **fib**$(0) = 0 = F_0$ and **fib**$(1) = 1 = F_1$. Thus $P(0)$ and $P(1)$ hold.

For our inductive step, assume that for some $n$ that $P(n)$ and $P(n + 1)$ hold and that **fib**$(n) = F_n$ and **fib**$(n + 1) = F_{n+1}$. We prove $P(n + 2)$, that **fib**$(n + 2) = F_{n+2}$. To see this, note that since $n + 2 \geq 2$, $n + 2 \neq 0$ and $n + 2 \neq 1$. Consequently, **fib**$(n + 2) =$ **fib**$(n) +$ **fib**$(n + 1)$. By our inductive hypothesis, this means that **fib**$(n + 2) = F_n + F_{n+1} = F_{n+2}$, as required. Thus $P(n + 2)$ holds, completing the induction. ■

Great – so we now have a function for computing Fibonacci numbers! But how efficient is it? To measure the complexity of this **fib** function, we should count some quantity that measures the total amount of work being done. One measure of complexity which might be good here is how many total function calls end up being required. Since each function does at most some fixed amount of work (namely, declaring variables, checking a fixed number of if statements, making a fixed number of recursive calls, and extra logic like setting up and tearing down the stack frame), we can claim that the total work done is proportional to the number of function calls made.

To determine how many function calls are made, we can start off by drawing out a *recursion tree*, a diagram of which function calls invoke which other function calls. Here are the recursion trees for $n = 0$, 1, 2, 3, and 4:




If we count up the number of nodes in these trees, we get 1, 1, 3, 5, 9, …, which doesn't seem to be any sequence that we know so far. Perhaps we could investigate the structure of this sequence in more depth to try to arrive at a nice formula for it.

To start things off, let's see if we can write out some nice recurrence that describes the terms in the series. Looking over our recursive function, we can note the following:

- If $n = 0$ or $n = 1$, exactly one function call is necessary.
- Otherwise, we need one function call for the initial call, plus a number of calls necessary to evaluate **fib**($n - 1$), plus a number of calls necessary to evaluate **fib**($n - 2$).

Let's denote by $C_n$ the number of function calls required to compute **fib**($n$). Translating the above definition, we end up getting that $C_n$ is defined as follows:

- $C_0 = C_1 = 1$.
- $C_{n+2} = C_n + C_{n+1} + 1$.

This is similar to the Fibonacci series, but it's not quite the same. Specifically, the Fibonacci sequence starts off with the first two terms 0, 1, and doesn't have the +1 term in the recurrence step. In case you're curious, the first few terms of this series are

1, 1, 3, 5, 9, 15, 25, 41, 67, 109, 177, …

The question now becomes how we can try to get a value for $C_n$, preferably in terms of the Fibonacci sequence $F_n$. There are many different approaches we can take here, and in this section we'll see two approaches, each based on a different techniques.

The first technique that we can use is based on the following idea. The above recurrence looks a *lot* like the normal Fibonacci sequence, but with a few extra 1s thrown into the mix. Could we somehow separate out $C_n$ into two terms – one term based on the Fibonacci sequence, plus one extra term based on the extra +1's?

To apply this technique, we will do the following. First, let's try to identify how much of this recurrence we can attribute to the Fibonacci sequence. One observation we can have is that since the first two terms of the sequence are 1s and each successive term depends (partially) on the sum of the previous two terms, we could consider thinking of this sequence as the Fibonacci sequence shifted over one step (as in the staircase problem), plus some extra terms. Specifically, let's see if we can write

$$C_n = F_{n+1} + E_n$$

Where $E_n$ is some "extra" term thrown into the mix to account for the extra 1s that we keep accumulating at each step. In other words, we can write

$$E_n = C_n - F_{n+1}$$

If we can now find some value for $E_n$, then we will end up being able to compute a value for $C_n$ in terms of the $(n+1)$st Fibonacci number $F_{n+1}$ and the sequence $C_n$. We still don't know how to compute $E_n$ yet, but we've at least stripped away some of the complexity of our original problem.

In order to learn what $E_n$ is, we should probably try writing out some values for $C_n - F_{n+1}$. Below are the values for this sequence:

| $C_n$ | 1 | 1 | 3 | 5 | 9 | 15 | 25 | 41 | 67 |
|---|---|---|---|---|---|---|---|---|---|
| $F_{n+1}$ | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 |
| $E_n$ | 0 | 0 | 1 | 2 | 4 | 7 | 12 | 20 | 33 |

Now *this* is interesting. If you'll notice, the value of $E_n$, the extra number of 1s added into the sequence, is always exactly one less than $F_{n+1}$. In other words it appears that, $E_n = F_{n+1} - 1$. Given that $C_n = F_{n+1} + E_n$, this would mean that $C_n = F_{n+1} + (F_{n+1} - 1) = 2F_{n+1} - 1$.

We haven't actually proven this yet, nor do we have much of an intuition for why this would be true. If we are purely interested in coming up with an answer to the question "how many function calls are made?," however, we don't actually need to know why. We can use a quick inductive proof to show that we have to be correct. (Don't worry – we'll definitely come back to *why* this is true in a minute).

***Theorem:*** The number of function calls required to compute **fib**($n$) is $2F_{n+1} - 1$.

***Proof:*** By induction. Let $P(n)$ be "**fib**($n$) makes $2F_{n+1} - 1$ function calls." We will prove that $P(n)$ is true for all $n \in \mathbb{N}$ by induction.

For our base cases, we prove $P(0)$ and $P(1)$; namely, that **fib**(0) makes $2F_1 - 1$ function calls and that **fib**(1) makes $2F_2 - 1$ function calls. In the case of both **fib**(0) and **fib**(1), exactly one function call is made, specifically the initial calls to **fib**. Since $F_1 = F_2 = 1$ and $2F_1 - 1 = 2 - 1 = 1$, this means that **fib**(0) makes $2F_1 - 1$ calls and **fib**(1) makes $2F_2 - 1$ calls, as required. Thus $P(0)$ and $P(1)$ hold.

For the inductive step, assume that for some $n$ that $P(n)$ and $P(n + 1)$ hold, meaning that **fib**($n$) makes $2F_{n+1} - 1$ calls and **fib**($n + 1$) makes $2F_{n+2} - 1$ calls. We will prove $P(n + 2)$, that **fib**($n + 2$) makes $2F_{n+3} - 1$ calls. To see this, consider the number of calls required to evaluate **fib**($n + 2$). The number of calls required is 1 for the initial call to **fib**($n + 2$), plus the number of calls required to evaluate **fib**($n$) and **fib**($n + 1$). By the inductive hypothesis, these values are $2F_{n+1} - 1$ and $2F_{n+2} - 1$, respectively. Thus the total number of calls is

$$\begin{aligned} & 1 + 2F_{n+1} - 1 + 2F_{n+2} - 1 \\ &= 2F_{n+1} + 2F_{n+2} + 1 - 1 - 1 \\ &= 2(F_{n+1} + F_{n+2}) - 1 \\ &= 2F_{n+3} - 1 \end{aligned}$$

Thus $2F_{n+3} - 1$ calls are required to evaluate **fib**($n + 2$), so $P(n + 2)$ holds, completing the induction. ■

The numbers 1, 1, 3, 5, 9, 15, 25, … are important in computer science and are called the *Leonardo numbers*. We denote the nth Leonardo number by $L_n$. The previous proof shows that $L_n = 2F_{n+1} - 1$.

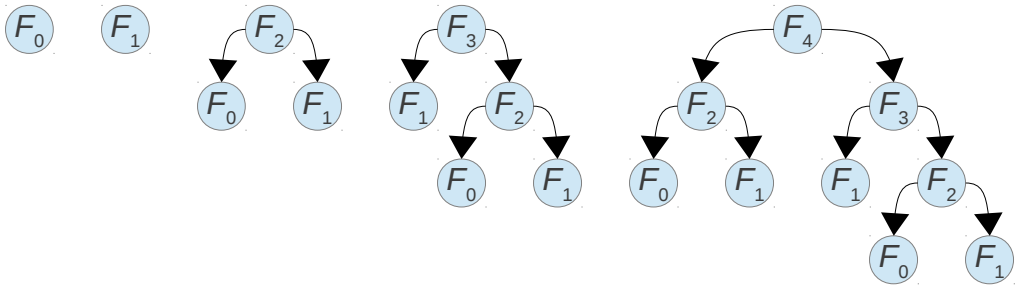
The above proof tells us that we at least have the right answer – the number of recursive calls is indeed $2F_{n+1} - 1$. However, it doesn't give us any insight whatsoever about where this number comes from. How on earth did we arrive at this figure?

Let's revisit the intuition that led us here. We separated $C_n$ into two terms – the $(n + 1)$st Fibonacci number $F_{n+1}$, plus some "extra" term $E_n$. Let's investigate exactly where each term comes from.

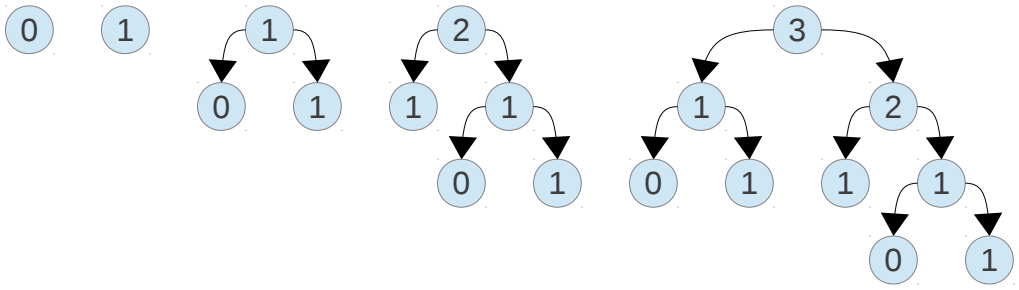
The key insight we need to have here is exactly how **fib**($n$) computes $F_n$. I've reprinted this function below for simplicity:

```
int fib(int n) {
   if (n == 0) return 0;
   if (n == 1) return 1;
   return fib(n - 1) + fib(n - 2);
}
```

Notice that the **fib** function works in one of two ways. First, for its base cases, it directly returns a value. For the recursive step, **fib** computes $F_n$ by computing $F_{n-2}$ and $F_{n-1}$ and adding those values together. But those values originally came from adding up even smaller Fibonacci numbers, which in turn came from adding up even smaller Fibonacci numbers, etc. Ultimately, the value returned by this function is derived by adding up the 0s and 1s returned in the base cases the appropriate number of times. You can see this below by reexamining the recursion trees for computing **fib**($n$):



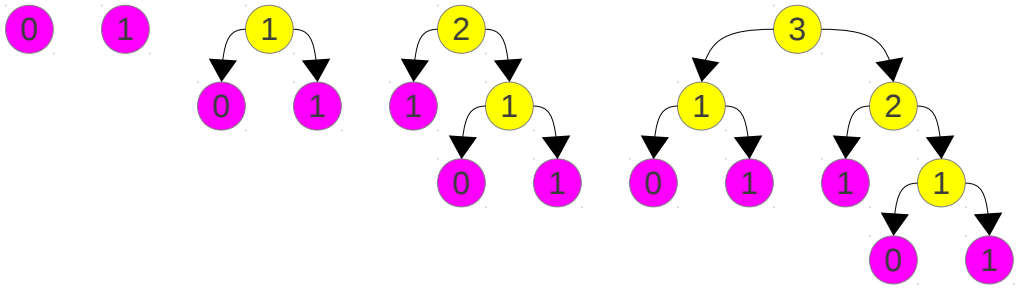

If we replace each function call with the value it returns, we get the following:




Each of the numbers is the sum of the numbers below it, which in turn are the sum of the numbers below them, until the recursion bottoms out into the base cases.

So now we can ask: how many of the function calls are base cases (values that actually produce the values), and how many of the function calls are recursive cases that just combine together previously-produced values?

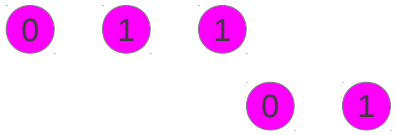
Below are the above recursion trees, with all of the recursive function calls highlighted in yellow, all of the base cases highlighted in magenta:



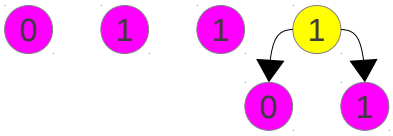

So how many of the function calls in each recursion tree are of each type? Notice that the number of magenta circles in each of the recursion trees is given by the following sequence:

1, 1, 2, 3, 5, 8, 13, …

This is the Fibonacci sequence shifted by one, which explains why we're getting some term that depends on $F_{n+1}$. Now, how many yellow circles are there? Notice that it's always equal to the number of magenta circles minus one. The reason for this is structural. Let's begin with any collection of $F_{n+1}$ magenta nodes representing the base cases; for example, like this:




Initially, all of these function calls are disconnected – there is nothing combining them together. In order to link them together into a recursion tree, we will need to link them together by the other function calls that called them. For example, we might add in this call:




Now, let's count how many different, disconnected "pieces" of the tree remain. Initially, we had five different function calls, each of which were isolated. Adding in this yellow function call reduces this down to four isolated pieces. In other words, adding in a yellow node decreased the number of disconnected pieces by one.

Every time we introduce one of the yellow internal nodes to connect together two trees that are previously disconnected, we decrease the number of disconnected trees by one. We ultimately need to end up with a single recursion tree, which means that we need to pairwise merge all $F_{n+1}$ disconnected trees together into a single tree. This means that we will need to do $F_{n+1} - 1$ merges, each of which introduces a yellow node. Consequently, the total number of nodes will be $F_{n+1} + (F_{n+1} - 1) = 2F_{n+1} - 1$. This gives a completely different but equally valid argument for why this must be the correct number of nodes in the tree.

# 3.5 Strong Induction

The flavors of induction we've seen so far – normal induction, induction starting at $k$, and Fibonacci induction – have enabled us to prove a variety of useful results. We will now turn to an even more powerful form of induction called *strong induction* that has numerous applications within computer science, from the analysis of algorithms to the understanding of the structure of numbers themselves.

To motivate strong induction, let's take a minute to think about how normal induction works. In a proof by induction, we first show that some property holds for 0 (or some other starting number, as you saw before). We then conclude "since it's true for 0, it's true for 1," then conclude "since it's true for 1, it's true for 2," then conclude "since it's true for 2, it's true for 3," etc. Notice that at each step in the induction, we only use the most recent result that we have proven in order to get to the next result. That is, to prove that the result is true for three, we only use the fact that the result is true for two, and not that it's true for zero or one. Similarly, if we wanted to prove that the result holds for a large number (say, 137), our proof would only rely on the fact that the result was true for 136.

In a sense, it seems like we're handicapping ourselves. Why must we only rely on the most recent result that we have proven? Why can't we use the entire set of all the results we've proven so far in order to establish the next result?

This is the intuition behind a powerful type of induction called *strong induction*:

> ***Theorem (strong induction):*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:
>
> $P(0)$ is true.
> For any $n \in \mathbb{N}$, if $P(0)$, $P(1)$, …, $P(n)$ are true, then $P(n + 1)$ is true.
>
> Then for any $n \in \mathbb{N}$, $P(n)$ is true.

Compare this to the normal principle of mathematical induction. In a regular induction, we would assume that $P(n)$ is true, then use it to show that $P(n + 1)$ is true. In strong induction, we assume that all of $P(0)$, $P(1)$, …, and $P(n)$ are true, then use this to show that $P(n + 1)$ is true.

To give a sort of intuition for strong induction, let's work through a simple example that shows how to use this style of proof technique. Suppose that you have a chocolate bar consisting of $n + 1$ smaller squares of chocolate, all in a line. For example, the candy bar might look like this:


You want to break this chocolate bar apart into $n + 1$ squares. How many breaks do you have to make in the chocolate bar in order to completely break it down?

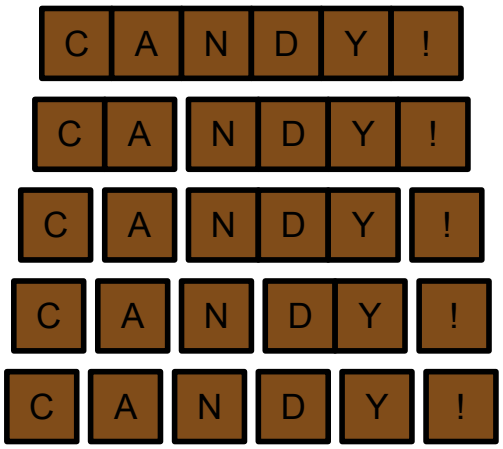
Let's try some examples. If you have a chocolate bar with 6 pieces, then you'll need to make five total breaks – one in-between each of the pieces of chocolate. If the chocolate bar has 137 pieces, you'd need 136 breaks. In general, it seems like you need to break the chocolate bar $n$ times if it has $n + 1$ squares, since there are $n$ separators.

This result might not seem all that impressive, but if we actually want to prove that this is optimal, we will need to be a bit more careful. Surely we can break the candy bar apart with $n$ breaks, but can we do it in *fewer* than $n$ breaks? The answer is no, and to prove this we will show the following result:

> ***Theorem:*** Breaking a linear candy bar with $n + 1$ pieces down into its individual pieces requires at least $n$ breaks.

How exactly can we show this? Well, we would somehow need to show that no matter how you try to break the candy bar apart, you always have to make at least *n* breaks. To do this, we can try the following line of reasoning – consider any possible way to break apart the candy bar, then show that no matter how it's done, it always uses at least *n* breaks.

So what is it like to break apart a candy bar this way? If you think about it, any way that we break apart the candy bar must start with some initial break, which will split the candy bar into two smaller pieces. From there, we can start breaking those smaller pieces down even further, and those smaller pieces down even further, etc. For example, here is one way of breaking down a candy bar with six pieces:




Now for the key insight: notice that as soon as we break the chocolate bar with the first break, we are left with two smaller chocolate bars. In order to break the overall chocolate bar down into individual squares, we will need to break those smaller pieces down into their individual parts. Consequently, we can think of any approach for breaking the chocolate bar down as follows:

- Make some initial break in the chocolate bar.
- Break the remaining pieces of chocolate down into their constituent pieces.

At some point this process has to stop, and indeed we can see that once we get down to a chocolate bar of size one, there is no longer any work to do.

Using this insight, we can prove by strong induction that the total number of breaks required is at least *n*. To do so, we'll initially prove the base case – that a chocolate bar with one piece requires no breaks – and from there will show that no matter how you break the chocolate bar into pieces, the total number of breaks required to subdivide the remaining pieces, plus the initial break, is always at least *n* if the chocolate bar has *n* + 1 pieces in it.

Here is the proof; we'll discuss it in some depth immediately afterwards:

***Proof:*** By strong induction. Let $P(n)$ be "breaking a linear chocolate bar with $n + 1$ pieces down into its constituent pieces requires at least $n$ breaks." We will prove $P(n)$ holds for all $n \in \mathbb{N}$ by strong induction on $n$.

For our base case, we prove $P(0)$, that any way of breaking a chocolate bar consisting of a single square into its constituent pieces takes at least zero breaks. This is true, since if there is just one square, it is already broken down as far is it can be, which requires no breaks.

For our inductive step, assume that for some $n \in \mathbb{N}$, that for any $n' \in \mathbb{N}$ with $n' \leq n$, that $P(n')$ holds and breaking a candy bar with $n' + 1$ pieces into its squares takes at least $n'$ breaks. We will prove $P(n + 1)$, that breaking a candy bar with $n + 2$ pieces requires at least $n + 1$ breaks. To see this, note that any way that we can break apart this candy bar will consist of an initial break that will split the candy bar into two pieces, followed by subsequent breaks of those smaller candy bars. Suppose that we break the candy bar such that there are $k + 1$ squares left in one smaller piece and $(n + 2) - (k + 1) = (n - k) + 1$ pieces in the second piece. Here, $k + 1$ must be no greater than $n + 1$, since if it were, we would have $n + 2$ squares in one smaller piece and 0 in the other, meaning that we didn't actually break anything. This means that $k + 1 \leq n + 1$, so $k \leq n$. Thus by our strong inductive hypothesis, we know that it takes at least $k$ breaks to split the piece of size $k + 1$ into its constituent pieces. Similarly, since $k \geq 0$, we know that $n - k \leq n$, so by our inductive hypothesis it takes at least $n - k$ breaks to break the piece of size $(n - k) + 1$ into its constituent pieces. This means that for any initial break, the total number of breaks required is at least $(n - k) + k + 1 = n + 1$, as required. Thus $P(n + 1)$ holds, completing the induction. ■

Let's dissect this proof and see exactly how it works. First, notice that we began the proof by announcing that we were going to use strong induction. Just as you should start a proof by induction, contradiction, or contrapositive by announcing how your proof will proceed, you should start proofs by strong induction with an explicit indication that this is what you are doing. It will make your proof much easier to read.

Notice that in the above proof, the proof of the base case proceeded as usual. We state what $P(0)$ is, then go prove it. However, the inductive step starts out noticeably differently from in our previous proofs by induction. Notice that it began as follows:

> For our inductive step, assume that for some $n \in \mathbb{N}$, that for any $n' \in \mathbb{N}$ with $n' \leq n$, that $P(n')$ holds and breaking a candy bar with $n' + 1$ pieces into its squares takes at least $n'$ breaks.

Here, we are not just assuming that $P(n)$ holds for some choice of $n$. Instead, we are assuming that we already know that $P(0)$, $P(1)$, $P(2)$, … $P(n)$ are true. Rather than writing this out longhand, we use a more compact notation and say that $P(n')$ is true for any $n' \leq n$. Most proofs by strong induction will proceed this way, and it is a good idea to make sure you understand why this notation with $n'$ is equivalent to listing off all the smaller choices of $n$.

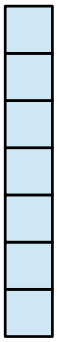
From there, the proof proceeds more or less as usual – we explain why we can reason about the initial break, and then introduce the variable $k$ to reason about how many squares are in each side. However, there is one key step to pay attention to. In the body of the inductive step, we invoke the inductive hypothesis twice, once for each piece of the chocolate bar. Notice how we do it:

Here, $k + 1$ must be no greater than $n + 1$, since if it were, we would have $n + 2$ squares in one smaller piece and 0 in the other, meaning that we didn't actually break anything. This means that $k + 1 \leq n + 1$, so $k \leq n$. Thus by our strong inductive hypothesis, we know that it takes at least $k$ breaks to split the piece of size $k + 1$ into its constituent pieces. Similarly, since $k \geq 0$, we know that $n - k \leq n$, so by our inductive hypothesis it takes at least $n - k$ breaks to break the piece of size $(n - k) + 1$ into its constituent pieces.

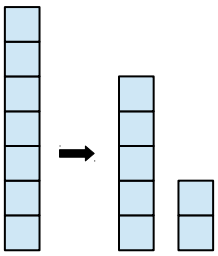
Before we claim that we can use the inductive hypothesis on the smaller pieces, we first verify that the size of each smaller piece is indeed no greater than $n$. It is **critical** that you do something like this in any proof by strong induction. The inductive hypothesis only applies to natural numbers that are less than or equal to $n$, and if you want to apply the inductive hypothesis to something of size $n'$, you need to first demonstrate that $n' \leq n$.

### 3.5.1 The Unstacking Game

Let's continue our exploration of strong induction with a simple game with a surprising strategy. This game, called the *unstacking game*, works as follows.* At the start of the game, you are presented a stack of $n + 1$ identical blocks. Your goal is to unstack all the boxes so that you are left with $n + 1$ stacks consisting of one block each. To do so, you are allowed to take a stack of at least one block, then "unstack" that block by splitting it into two stacks, where each stack has at least one block in it. For example, if you are playing this game with seven blocks, the game might start out like this:

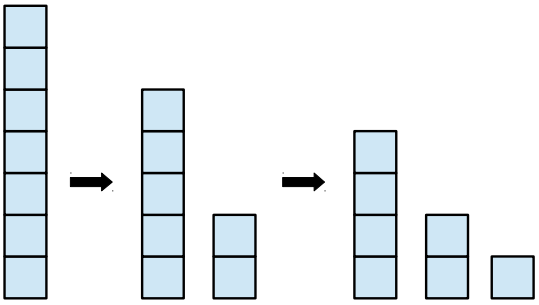


As your first move, you might split this tower into two stacks, one with two blocks and one with five blocks, as shown here:



Your next move might be to split the stack of five blocks into two stacks, one with four blocks and one with one block, as seen here:

* I first heard this problem from Prof. Dieter van Melkebeek of the University of Wisconsin.

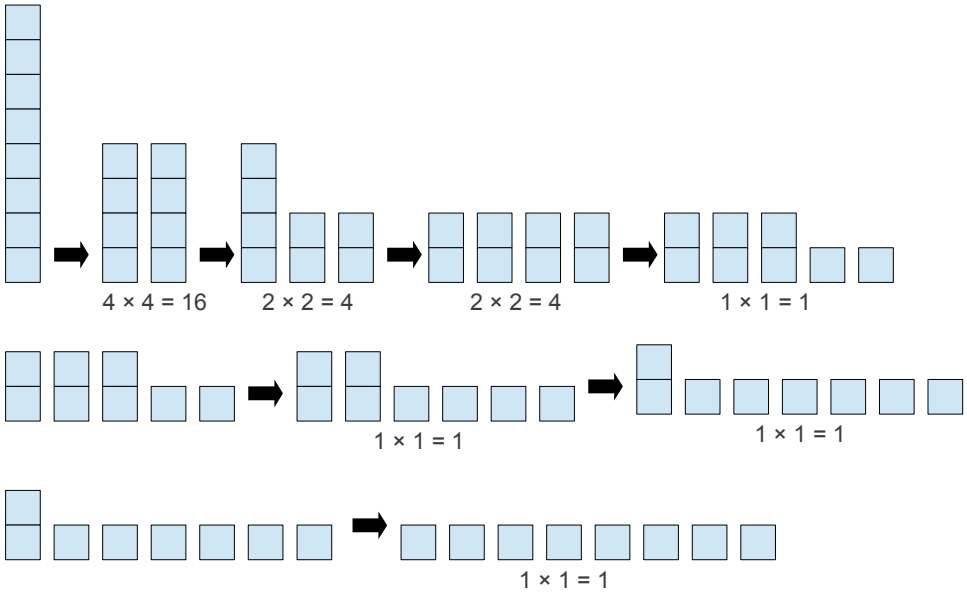


If you'll notice, so far this game is identically the same as the breaking chocolate problem – we have a linear stack of blocks, and keep breaking that stack into smaller and smaller pieces. What differentiates this game from the chocolate bar problem is that you are awarded a different number of points based on how you break the stack into two pieces. In particular, with each move, you earn a number of points equal to the product of the number of blocks in each of the smaller stacks that you create. For example, in the first move, above, you would earn ten points ($5 \times 2$), and in the second move you would earn four points ($4 \times 1$).

Now, for the key question: what strategy should you use to maximize the number of points that you earn?

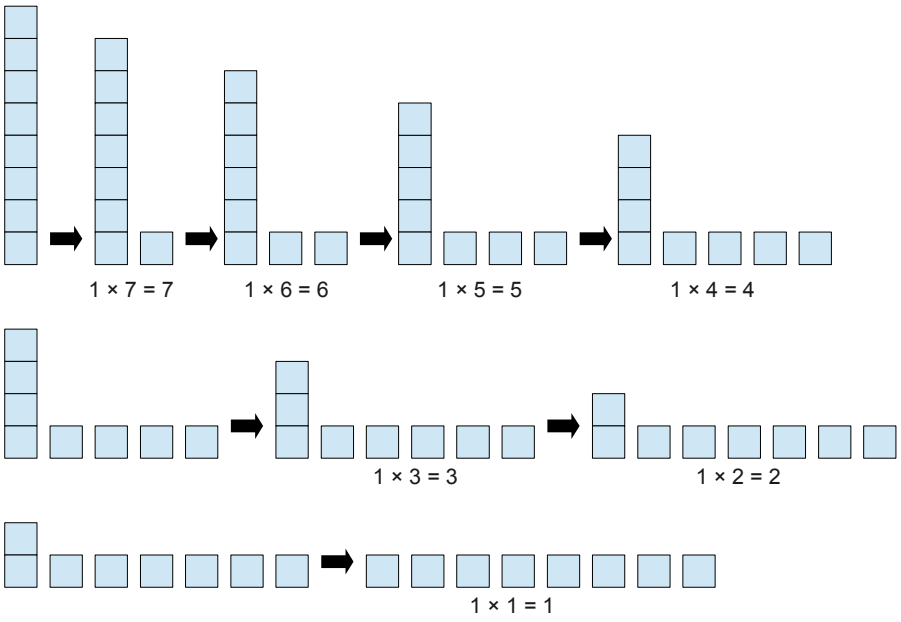
When confronted with a problem like this one, sometimes the best option is to try out a bunch of different ideas and see how they pan out. Let's consider, for example, a stack of eight blocks. If we are trying to maximize our score, then we might consider a few different strategies. Since our score depends on the product of the sizes of the splits we make, one option might be, at each point, to split the largest tower we have in half. This maximizes the score we get at each step, though it rapidly decreases the size of the remaining towers. Another option might be to always just peel off one block from the tower at a time. This would mean that each turn individually doesn't give us that many points back, but would mean that the rate at which the tower shrinks is minimized, thus giving us more points for a longer period of time.

Let's try these strategies out! If we adopt the first strategy and keep splitting the largest tower cleanly in half, then we get the following game:




16 + 4 + 4 + 1 + 1 + 1 + 1 = **28**

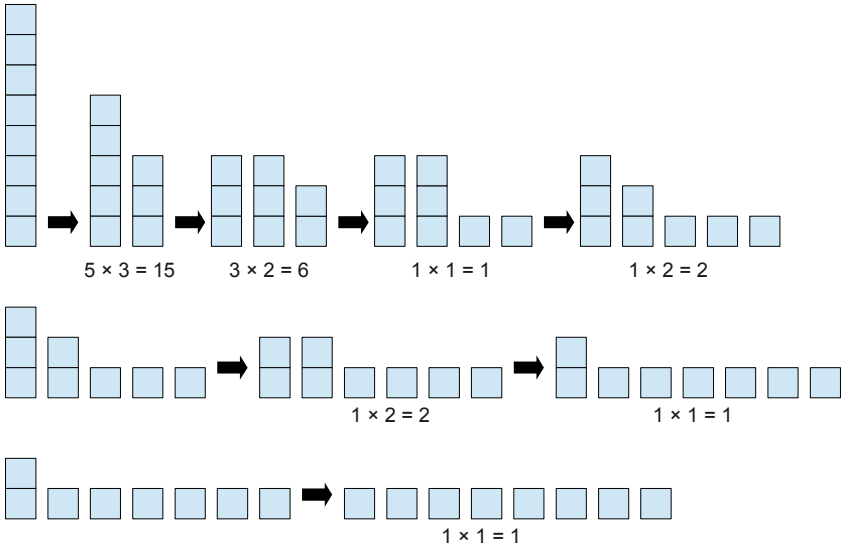
This gives us a net of 28 points. If we adopt the second strategy, though, we get this game:




7 + 6 + 5 + 4 + 3 + 2 + 1 = **28**

Interestingly, we end up with 28 points again, which is the same as before. That's an odd coincidence – it doesn't initially seem like these strategies should give the same score.

Let's try out a different approach. Eight happens to be a Fibonacci number, so perhaps we could split the blocks apart using the patterns of the Fibonacci sequence. First, we split the 8 into a 5 and a 3, then the 5 into a 3 and a 2, then the 3 into a 2 and a 1, etc. This strategy combines the previous two strategies nicely – we break the towers apart into large chunks, but don't split the tower too fast. How many points will we get? If we play with this strategy, we get this result:




15 + 6 + 1 + 2 + 2 + 1 + 1 = **28**

Amazing! Somehow we get 28 points once again. This is starting to seem a bit suspicious – we have come up with three totally different strategies, and each time end up with exactly the same score. Is this a coincidence? Or is there something deeper going on?

Before moving on, let's make the following conjecture:

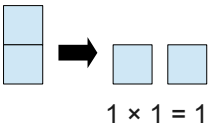
*No matter how you play the unstacking game, you always get the same score.*

Is it premature of us to conclude this? Possibly. We could end up being wrong and find that there actually are strategies that give you more points than others. But at this point we're still exploring.

In order for us to explore this conjecture, we will need to do more than just play the game on a stack of size eight. Instead, let's try playing the game on smaller stacks. That way, we can actually exhaustively list of all possible ways that the game could be played, which would let us either (1) gather supporting evidence that our conjecture is correct, or (2) find a counterexample that might tell us something more about the game.

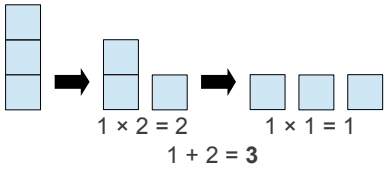
Well, let's begin with a very simple game, where we have exactly one block in the initial stack. In this case, the game immediately ends with us scoring 0 points. In that case, the claim "every strategy produces the same score" is pretty obviously true.

If we have two blocks, then there is just one strategy:



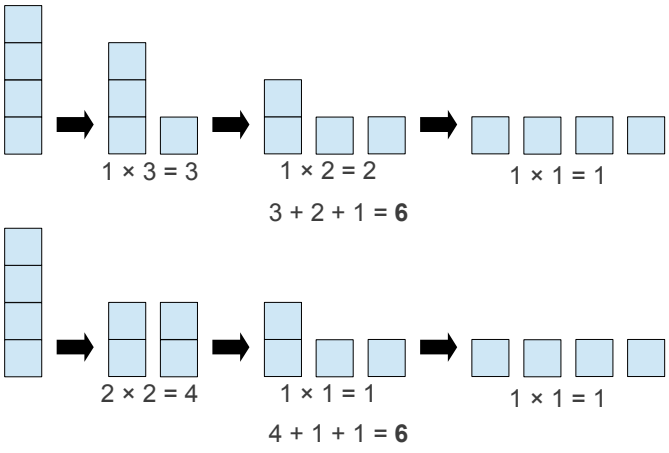

We end up with one point, and this was the only strategy.

What about three blocks? It turns out that there's only one strategy here as well:




And we earn three points. So far this isn't particularly interesting, since there's only one strategy we can use in these cases.

The first interesting case we find is when there are four blocks in the stack, because now we actually have a choice of what to do. One option would be to split the stack into two stacks of size two, while the other would be to split it into a stack of size three and a stack of size one. Both strategies are shown here:




In both cases, we end up with six points. Our conjecture is starting to seem like it might actually be true!

At this point we can start to seriously believe that this conjecture might be true. Our next step will be to think about how exactly we're supposed to prove it.

There are many paths we can use to prove that the score is always the same. Here's one particularly useful idea: since we're claiming that the score is always the same, we might start by asking exactly how many points you will end up getting in a game with $n$ blocks in the initial stack. So far, we have this data:

| n | Total Score |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 3 | 3 |
| 4 | 6 |
| 5 | ? |
| 6 | ? |
| 7 | ? |
| 8 | 28 |

This sequence should seem familiar; we've encountered it before. If you'll recall, the sequence 0, 1, 3, 6, ?, ?, ?, 28, …, is the sequence you get if you sum up the first $n$ natural numbers. After all:

- The sum of the first 1 natural numbers is 0.
- The sum of the first 2 natural numbers is 0 + 1 = 1.
- The sum of the first 3 natural numbers is 0 + 1 + 2 = 3.

In other words, if we have a game with $n$ blocks in the initial stack, the score we would expect to get is equal to the sum of the first $n$ natural numbers. As we saw before, this sum is $n(n - 1) / 2$. Rephrasing this, if we have $n + 1$ blocks in the initial stack, our score should be $n(n + 1) / 2$.

This might initially seem completely unexpected, but this particular sequence of numbers is not completely unexpected. After all, one strategy that we can use works by always splitting the stack by pulling off exactly one block at a time. This means that if we have $n + 1$ blocks in the initial stack, we'll get $n$ points on the first move, $n - 1$ points on the second move, $n - 3$ points on the third move, etc., giving us a net of $1 + 2 + \ldots + n$ points. We still haven't accounted for why the score is always exactly the same – that's a much deeper question – but we at least aren't completely in the dark about where this number is coming from.

We now have a stronger version of our initial conjecture from before:

**If you play the unstacking game with $n + 1$ blocks, you will get $n(n + 1) / 2$ points.**

There are two questions left to consider – first, how do we prove this? Second, why is this true? These are completely separate questions! It's often possible to prove results without having a deep understanding about why they are true. In an unusual twist, I'd like to first go and prove that the result is true. We'll then come back and try to figure out exactly why this result is true. My reasoning for this is twofold. First, starting off with an initial proof helps guide our intuition as to why the result might be true. Second, in the

course of exploring why this result happens to be true, we will be able to look back at our initial proof a second time and explore exactly why it works.

So, how might we try proving this result? For this, we can return to the proof technique we used in the chocolate bar problem. If you'll notice, when playing the unstacking game, every possible strategy consists of a first move, in which we split the stack into two pieces. From there, there is no further overlap between the points earned from those two stacks; we could think of the game as being two completely independent games, each played on a stack whose size is determined by how we cut the initial stacks apart.

Given that this is true, we could proceed as follows. First, we'll show that the score obtained by any strategy when there is exactly one block is always 0. Next, we'll assume that for any stacks of size 1, 2, 3, …, $n + 1$, that the claim holds, and will consider a stack of size $n + 2$. The first move we make on such a stack will split it into two smaller stacks, one of which we'll say has size $k + 1$, and the other of which has size $(n - k) + 1$. From there, we can apply the inductive hypothesis to get the total number of points from the subgames, and can add in the score we got from making this particular move. If we can show that this sum comes out correctly, then we will have a valid induction proof.

We can turn this proof idea into an actual proof here:

***Theorem:*** No matter what strategy is used, the score for the unstacking game with $n + 1$ blocks is $n(n + 1) / 2$.

***Proof:*** By strong induction. Let $P(n)$ be "no matter what strategy is used, the score for the unstacking game with $n + 1$ blocks is $n(n + 1) / 2$." We will prove that $P(n)$ holds for all $n \in \mathbb{N}$ by strong induction.

For our base case, we prove $P(0)$, that any strategy for the unstacking game with one block will always yield $0( + 1) / 2 = 0$ points. This is true because the game immediately ends if the only stack has size one, so all strategies immediately yield 0 points.

For the inductive hypothesis, assume that for some $n \in \mathbb{N}$, that for all $n' \in \mathbb{N}$ with $n' \leq n$, that $P(n')$ holds and the score for the unstacking game played with $n' + 1$ blocks is always $n'(n' + 1) / 2$. We will prove that $P(n + 1)$ holds, meaning that the score for the unstacking game played with $n + 2$ blocks is always $(n + 1)(n + 2) / 2$. To see this, consider any strategy for the unstacking game with $n + 2$ blocks. This strategy consists of making some initial split, producing two smaller stacks, then splitting those smaller stacks down. Suppose that the initial split places $k + 1$ blocks into one stack, which leaves $n + 2 - (k + 1) = (n - k) + 1$ blocks in the other stack. Since each stack must have at least one block in it, this means that $k \geq 0$ (so that $k + 1 \geq 1$) and that $k \leq n$ (so that $(n - k) + 1 \geq 1$). Consequently, we know that $0 \leq k \leq n$, so by the inductive hypothesis we know that the total number of points earned from splitting the stack of $k + 1$ blocks down must be $k(k + 1) / 2$. Similarly, since $0 \leq k \leq n$, we know that $0 \leq n - k \leq n$, and so by the inductive hypothesis the total score for the stack of $(n - k) + 1$ blocks must be $(n - k)(n - k + 1) / 2$.

Let us consider the total score for this game. The initial move yields $(k + 1)(n - k + 1)$ points. The two subgames yield $k(k + 1) / 2$ and $(n - k)(n - k + 1) / 2$ points, respectively. This means that the total number of points earned is

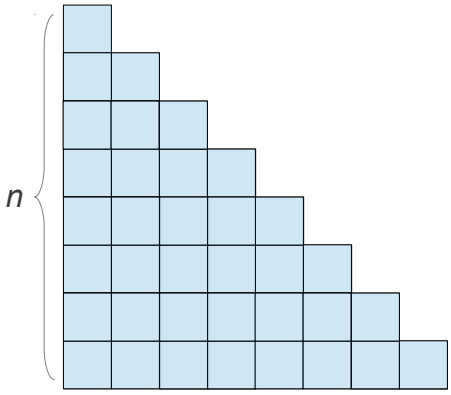
$$
\begin{aligned}
& (k + 1)(n - k + 1) + k(k + 1) / 2 + (n - k)(n - k + 1) / 2 \\
= \ & 2(k + 1)(n - k + 1)/2 + k(k + 1) / 2 + (n - k)(n - k + 1) / 2 \\
= \ & (2(k + 1)(n - k + 1) + k(k + 1) + (n - k)(n - k + 1)) / 2 \\
= \ & (2kn - 2k^2 + 2k + 2n - 2k + 2 + k(k + 1) + (n - k)(n - k + 1)) / 2 \\
= \ & (2kn - 2k^2 + 2n + 2 + k(k + 1) + (n - k)(n - k + 1)) / 2 \\
= \ & (2kn - 2k^2 + 2n + 2 + k^2 + k + (n - k)(n - k + 1)) / 2 \\
= \ & (2kn - k^2 + 2n + 2 + k + (n - k)(n - k + 1)) / 2 \\
= \ & (2kn - k^2 + 2n + 2 + k + n^2 - kn + n - kn + k^2 - k) / 2 \\
= \ & (2n + 2 + n^2 + n) / 2 \\
= \ & (n^2 + 3n + 2) / 2 \\
= \ & (n + 1)(n + 2) / 2
\end{aligned}
$$

As required. Since this result holds for any valid choice of $k$, we have that $P(n + 1)$ holds, completing the proof. ■

No doubt about it – this proof is dense. The math in the middle section is very difficult, and seems to work out through pure magic. There has to be a better way to prove this result, but to find it, we're going to need to develop a better understanding of what's going on. The key to simplifying this proof will be to find a better way to understand why on earth the math in the middle section happens to work out correctly.

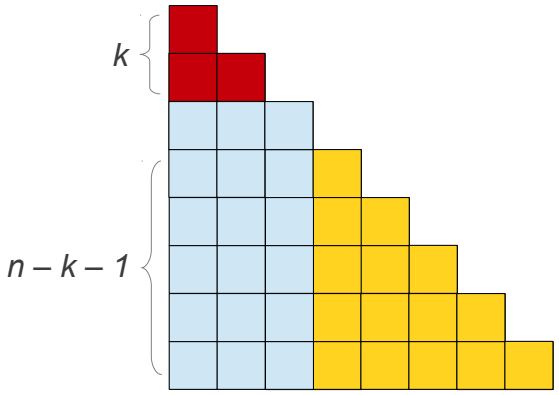
At this point, let's perform a somewhat unusual step. We built up a huge amount of machinery at the start of this chapter to be able to replace summations with nice closed form expressions that don't involve any sums. In this proof, we ended up showing that the game, played with $n + 1$ blocks, always produces a score of $n(n + 1) / 2$. Equivalently, though, we could have proven that the score, when the game is played with $n + 1$ blocks, is equal to $\sum_{i=0}^{n} i$. This might at first seem like we're making things more complicated – after all, we've replaced a simple polynomial with a summation – but this might not actually be all that bad an idea. After all, remember that the game score is formed by summing together a lot of smaller parts. Perhaps putting our game total into the form of a summation will make things easier.

By rewriting the total score for $n + 1$ blocks using a summation, we can also use some of the geometric techniques from earlier in order to reason about the sum. If you'll recall, the above summation is equal to the number of squares in this "boxy triangle:"




Now, let's suppose that you're playing with $n + 1$ blocks and split the stack into two pieces. One of these pieces will have $k + 1$ blocks in it; the other will have $n - k$ blocks in it. This means that the total number of points that you will earn will be $\sum_{i=0}^{k} i$ from the tower of size $k$, $\sum_{i=0}^{n-k-1} i$ from the tower of size $n - k$,

and $(k + 1)(n - k)$ from the move itself. The first two of these sums have nice geometric intuitions – they're the number of square in "boxy triangles" of height $k$ and $n - k - 1$, respectively. In fact, we can superimpose these triangles on top of the original boxy triangle from before:




Now, look at what's left uncovered in the above triangle – it's a rectangle of dimension $(k + 1) \times (n - k)$. Such a rectangle has area $(k + 1)(n - k)$, which is precisely the number of points we earned from making this move.

This gives us a completely different intuition for what's going on. When we make a move, we earn a number of points equal to the area of some rectangle. If we then cover up a part of the boxy triangle with that rectangle, we're left with two smaller triangles that are still yet to be covered – one representing the points we'll earn from the first stack created, and one representing the points we'll earn from the second stack we created. Beautiful, isn't it?

What remains now is to repeat the previous proof using this geometric intuition. To do so, let's start off by writing out the total points as a sum of the two summations and the points earned from the single turn:

$$\sum_{i=0}^{k-1} i + \sum_{i=0}^{n-k} i + k(n - k + 1)$$

We'd like to somehow show that we can rewrite this sum as the simpler sum

$$\sum_{i=0}^{n} i$$

How exactly might we do this? Well, we arrived at this summation by taking some quantities that we already knew quite well, then replacing them with (allegedly) more complex summations. What if we rewrite that final product $(k(n - k) + 1)$ not as a product, but as a sum of a number of terms? In particular, we can treat this sum as the sum of $n - k + 1$ copies of the number $k$. If we rewrite the product this way, we get

$$\sum_{i=0}^{k-1} i + \sum_{i=0}^{n-k} i + \sum_{i=0}^{n-k} k$$

Notice that this summation runs from 0 to $n - k$, which includes $n - k + 1$ different terms.

Next, we can use the properties of summations we proved earlier in the chapter to simplify the above expression. In particular, notice that the last two sums run over the exact same indices. As we saw from before, this enables us to combine them together into a single sum, whose summand is just the sum of the two smaller summands (it might help to reread that once or twice to make sure you can parse it correctly). This means that we can rewrite the above as

$$\sum_{i=0}^{k-1} i + \sum_{i=0}^{n-k} (i + k)$$

And now for the final step. This second summation ranges from $i = 0$ to $n - k$, and at each point sums up the value $i + k$. Let's suppose that we define a new variable $j$ and say that $j = i + k$. If we do this, then we'll find that as $i$ ranges from 0 to $n - k$, this variable $j$ ranges from $k$ to $n$. Consequently, we have the following:

$$\sum_{i=0}^{n-k} (i+k) = \sum_{j=k}^{n} j$$

This means that we can rewrite our sum as

$$\sum_{i=0}^{k-1} i + \sum_{j=k}^{n} j$$

Finally, we'll do one more simplification. This first sum computes $0 + 1 + 2 + \ldots + k - 1$. The second sum computes $k + (k + 1) + (k + 2) + \ldots + n$. We can therefore just combine the sums together to get

$$\sum_{i=0}^{n} i$$

*Et voilà*. We've got the sum that we wanted to achieve.

Using summations rather than the explicit formula n(n + 1) / 2, let's rewrite our initial proof about the unstacking game. This new proof is much shorter, much cleaner, and gives a better intuition for what's going on.

***Theorem:*** No matter what strategy is used, the score for the unstacking game with $n + 1$ blocks is $\sum_{i=0}^{n} i$.

***Proof:*** By strong induction. Let $P(n)$ be "no matter what strategy is used, the score for the unstacking game with $n + 1$ blocks is $\sum_{i=0}^{n} i$." We will prove that $P(n)$ holds for all $n \in \mathbb{N}$ by strong induction.

For our base case, we prove $P(0)$, that any strategy for the unstacking game with one block will always yield $\sum_{i=0}^{0} i = 0$ points. This is true because the game immediately ends if the only stack has size one, so all strategies immediately yield 0 points.

For the inductive hypothesis, assume that for some $n \in \mathbb{N}$, that for all $n' \in \mathbb{N}$ with $n' \leq n$, that $P(n')$ holds and the score for the unstacking game played with $n' + 1$ blocks is always $\sum_{i=0}^{n'} i$. We will prove that $P(n + 1)$ holds, meaning that the score for the unstacking game played with $n + 2$ blocks is always $\sum_{i=0}^{n+1} i$. To see this, consider any strategy for the unstacking game with $n + 2$ blocks. This strategy consists of making some initial split, producing two smaller stacks, then splitting those smaller stacks down. Suppose that the initial split places $k + 1$ blocks into one stack, which leaves $n + 2 - (k + 1) = (n - k) + 1$ blocks in the other stack. Since each stack must have at least one block in it, this means that $k \geq 0$ (so that $k + 1 \geq 1$) and that $k \leq n$ (so that $(n - k) + 1 \geq 1$). Con-

sequently, we know that $0 \le k \le n$, so by the inductive hypothesis we know that the total number of points earned from splitting the stack of $k + 1$ blocks down must be $\sum_{i=0}^{k} i$. Similarly, since $0 \le k \le n$, we know that $0 \le n - k \le n$, and so by the inductive hypothesis the total score for the stack of $(n - k) + 1$ blocks must be $\sum_{i=0}^{n-k} i$.

Now, let us consider the total score for this game. Splitting the stack with the initial move yields $(k + 1)(n - k + 1)$ points. The two subgames yield $\sum_{i=0}^{k} i$ and $\sum_{i=0}^{n-k} i$ points, respectively. This means that the total number of points earned is

$$\begin{aligned}
&\sum_{i=0}^{k} i + \sum_{i=0}^{n-k} i + (k+1)(n-k+1) \\
&= \sum_{i=0}^{k} i + \sum_{i=0}^{n-k} i + \sum_{i=0}^{n-k} (k+1) \quad = \sum_{i=0}^{k} i + \sum_{i=0}^{n-k} (i+k+1) \\
&= \sum_{i=0}^{k} i + \sum_{i=k+1}^{n-k+1} i \quad = \sum_{i=0}^{n-k+1+k} i \quad = \sum_{i=0}^{n+1} i
\end{aligned}$$

As required. Since this result holds for any valid choice of $k$, we have that $P(n + 1)$ holds, completing the proof. ■

## 3.5.2 Variations on Strong Induction ★

Just as we saw several variants on normal induction, there are many ways that we can slightly modify strong induction in order to make it easier to use in our proofs. This section surveys some of these new approaches.

One of the first modifications we made to normal induction was the ability to start the induction at a value other than 0. We saw, and formally proved, that we can begin normal induction from some value $k$ if we want to prove that some result holds for all values greater than or equal to $k$. We can similarly do this to strong induction:

***Theorem (strong induction from k):*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:

> $P(k)$ is true.
> For any $n \in \mathbb{N}$, if $P(k)$, $P(k + 1)$, …, $P(n)$ are true, then $P(n + 1)$ is true.

Then for any $n \in \mathbb{N}$ with $n \ge k$, $P(n)$ is true.

The proof of this result is left as an exercise at the end of the chapter. Demonstrating that this is true is similar to demonstrating that the result is true for weak induction – first, invent a new predicate $Q(n)$ that can be proven by strong induction, then use the fact that $Q(n)$ is true for all $n \in \mathbb{N}$ to show that $P(n)$ holds for all $n \ge k$.

A much more interesting thing to do with this new proof technique is to repeat the proof of the unstacking game one last time with a few simplifications. Notice that the proof of the unstacking game score always revolved around a stack of $n + 1$ blocks. This may have seemed strange, and with good reason. Talking about stacks of $n + 1$ blocks is a mathematically sneaky way of allowing us to only reason about stacks

that have one or more blocks in them. A much nicer version of the result works as follows – instead of proving that the score for a stack of $n + 1$ blocks is $\sum_{i=0}^{n} i$, instead we will prove that the score for a stack of $n$ blocks, where $n \geq 1$, is given by $\sum_{i=0}^{n-1} i$. This dramatically simplifies the proof, because a lot of the math required to handle the "+1" term suddenly vanishes.

The proof is given here:

**Theorem:** No matter what strategy is used, the score for the unstacking game with $n$ blocks, with $n \geq 1$, is $\sum_{i=0}^{n-1} i$.

***Proof:*** By strong induction. Let $P(n)$ be "no matter what strategy is used, the score for the unstacking game with $n + 1$ blocks is $\sum_{i=0}^{n} i$." We will prove that $P(n)$ holds for all $n \in \mathbb{N}^+$ by strong induction.

For our base case, we prove $P(1)$, that any strategy for the unstacking game with one block will always yield $\sum_{i=0}^{1-1} i = \sum_{i=0}^{0} i = 0$ points. This is true because the game immediately ends if the only stack has size one, so all strategies immediately yield 0 points.

For the inductive hypothesis, assume that for some $n \in \mathbb{N}^+$, that for all $n' \in \mathbb{N}^+$ with $n' \leq n$, that $P(n')$ holds and the score for the unstacking game played with $n'$ blocks is always $\sum_{i=0}^{n'-1} i$. We will prove that $P(n + 1)$ holds, meaning that the score for the unstacking game played with $n + 1$ blocks is always $\sum_{i=0}^{n} i$. To see this, consider any strategy for the unstacking game with $n + 1$ blocks. This strategy consists of making some initial split, producing two smaller stacks, then splitting those smaller stacks down. Suppose that the initial split places $k$ blocks into one stack, which leaves $n + 1 - k = n - k + 1$ blocks in the other stack. Since each stack must have at least one block in it, this means that $k \geq 1$ and that $k \leq n$ (so that $n - k + 1 \geq 1$). Consequently, we know that $1 \leq k \leq n$, so by the inductive hypothesis we know that the total number of points earned from splitting the stack of $k$ blocks down must be $\sum_{i=0}^{k-1} i$. Similarly, since $1 \leq k \leq n$, we have $1 \leq n - k + 1 \leq n$, and so by the inductive hypothesis the total score for the stack of $(n - k) + 1$ blocks must be $\sum_{i=0}^{n-k} i$.

Now, let us consider the total score for this game. Splitting the stack with the initial move yields $k(n - k + 1)$ points. The two subgames yield $\sum_{i=0}^{k-1} i$ and $\sum_{i=0}^{n-k} i$ points, respectively. This means that the total number of points is

$$\begin{aligned}
&\sum_{i=0}^{k-1} i + \sum_{i=0}^{n-k} i + k(n-k+1) \\
&= \sum_{i=0}^{k-1} i + \sum_{i=0}^{n-k} i + \sum_{i=0}^{n-k} k \quad = \sum_{i=0}^{k-1} i + \sum_{i=0}^{n-k} (i+k) \\
&= \sum_{i=0}^{k} i + \sum_{i=k}^{n-k} i \quad = \sum_{i=0}^{n-k+k} i \quad = \sum_{i=0}^{n} i
\end{aligned}$$

As required. Since this result holds for any valid choice of $k$, we have that $P(n + 1)$ holds, completing the proof. ■

One final variant on strong induction that is often useful from an algorithmic perspective recasts the form of strong induction in a new light. With strong induction, we use the fact that $P(0)$, $P(1)$, $P(2)$, …, $P(n)$ all hold, then use these smaller results to conclude that $P(n + 1)$ holds as well. A slightly different way of phrasing this is as follows: we can assume that $P(0)$, $P(1)$, …, $P(n – 1)$ all hold, then prove that $P(n)$ holds as well. In other words, instead of assuming that the claim holds up to and including $n$, we can assume that that claim holds for everything smaller than $n$, and then show that the result holds for $n$ as well. In many mathematics textbooks, this is the preferred version of induction. You'll see an example of why this is later on.

Formally speaking, we could write this as follows:

***Theorem:*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:

$P(0)$ is true.
For any $n \in \mathbb{N}$, if $P(0)$, $P(1)$, …, $P(n – 1)$ are true, then $P(n)$ is true.

Then for any $n \in \mathbb{N}$, $P(n)$ is true.

However, we can simplify this even further. Let's be a bit more formal in this definition. Instead of saying the somewhat hand-wavy "$P(0)$, …, $P(n – 1)$ are true," let's rewrite this as

***Theorem:*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:

$P(0)$ is true.
For any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true,
then $P(n)$ is true.

Then for any $n \in \mathbb{N}$, $P(n)$ is true.

At this point, we can make a clever (some might say *too clever*) observation. Think about this second claim as applied to 0. Under this claim, we would be doing the following: assume that $P(n')$ holds for all natural numbers $n'$ less than 0, then use this to prove $P(0)$. In this case, the statement "$P(n')$ holds for all natural numbers $n'$ less than 0" is vacuously true, because there *are* no natural numbers less than 0. Consequently, the statement "If $P(n')$ holds for all natural numbers $n'$ less than 0, then $P(0)$ holds" is completely equivalent to "$P(0)$ holds."

Given this, we can rewrite the above theorem one last time:

***Theorem:*** Let $P(n)$ be a property that applies to natural numbers. If for any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true, then $P(n)$ is true, then we can conclude that for any $n \in \mathbb{N}$, $P(n)$ is true.

With this new style of induction, we have not eliminated the need for a base case. Instead, we have folded the base case into the general structure of how the proof proceeds.

To see an example of this style of induction in action, let's go back and reprove something that we already know to be true – that the sum of the first $n$ positive natural numbers is $n(n + 1) / 2$. In this proof, we'll use the style of induction described above. Take note of how this proof works:

***Theorem:*** $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

***Proof:*** By strong induction. Let $P(n)$ be

$$P(n) \equiv \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

We will prove that $P(n)$ holds for all $n \in \mathbb{N}$ by strong induction.

Assume that for some $n \in \mathbb{N}$, that for all $n' \in \mathbb{N}$ where $n' < n$, that

$$\sum_{i=1}^{n'} i = \frac{n'(n'+1)}{2}$$

We will prove that in this case, $P(n)$ holds. We consider two cases. First, if $n = 0$, then we see that

$$\sum_{i=1}^{0} i = 0 = \frac{0(0+1)}{2}$$

So $P(n)$ holds. Otherwise, if $n > 0$, then we know that $n \geq 1$. Consequently,

$$\sum_{i=1}^{n} i = \sum_{i=1}^{n-1+1} i = \sum_{i=1}^{n-1} i + n = \frac{(n-1)n}{2} + n = \frac{n(n-1)+2n}{2} = \frac{n(n+1)}{2}$$

As required. Thus $P(n)$ holds, completing the induction. ■

This proof is as subtle as it is elegant. Notice that we don't explicitly state anywhere that we're going to prove the base case. We start off with the inductive assumption and proceed from there. That isn't to say that we don't prove the base case – we do – but it's not labeled as such. Instead, our proof works by cases. In the first case, we check out what happens if $n = 0$. This serves as the "stand-in" for our base case. In the second case, we can proceed confident that $n > 0$, meaning that $n \geq 1$. This case is similar to the inductive step from normal inductive proofs, except that we want to prove $P(n)$ instead of $P(n + 1)$.

Looking at the above proof, you might wonder why we need to have two cases at all. Why can't we always use the logic from the second case? The answer is critically important and surprisingly subtle. Notice that in this part of the proof, we pull a term off of the sum:

$$\sum_{i=1}^{n} i = \sum_{i=1}^{n-1+1} i = \sum_{i=1}^{n-1} i + n$$

The only reason that we can pull a term off of this sum is because we know that the sum actually has at least one term in it. If $n = 0$, then this is the empty sum, and we can't just go pulling terms off of it. For example, the following reasoning is *not sound*:

$$\sum_{i=1}^{0} 2^n = \sum_{i=1}^{-1} 2^n + 2^0$$

This absolutely does not work, because the left-hand side is the empty sum (0), while the right-hand side is the empty sum (0) plus 1. Obviously $0 \neq 1$, so something must be wrong here. The problem here lies in the fact that we pulled a term off from the empty sum, something we're not allowed to do. For this reason, in our previous proof, in order to peel a term off of the sum, we first had to show that $n \geq 1$, meaning that there actually was at least one term to peel off. Consequently, we had to separate out the logic for the case where $n = 0$ from the rest of the proof.

## 3.5.3 A Foray into Number Theory

To see just how powerful strong induction is, we will turn to a problem explored by ancient mathematicians in Greece and India. This problem concerns finding a way to measure two different quantities using a single common measure. Suppose, for example, that two drummers are each hitting a drum at different rates. The first drummer (call her drummer A) hits her drum once every four seconds, and the second drummer (call him drummer B) hits his drum once every six seconds. If these drummers start at the same time, then the series of drum hits will be as follows:

- Time 0: A, B
- Time 4: A
- Time 6: B
- Time 8: A
- Time 12: A, B
- Time 16: A
- Time 18: B
- Time 20: A
- Time 24: A, B

As you can see, there is a repeating pattern here – at precisely determined intervals, A and B will hit their drums, sometimes hitting the drum together, and sometimes hitting the drum at different times. One question we might ask is the following – given the time delays between when drummers A and B hit their drums, at what time intervals will A and B simultaneously hit their drums?

Let's consider a different, related problem. Suppose that you have a room of dimensions 60m × 105m and you want to tile the floor with square tiles. In doing so, you want to use the smallest number of square tiles possible. What dimension should you make the square tiles in order to minimize the total number of tiles needed? Well, you can always just use tiles of size 1m × 1m, which would require you to use 60 × 105 = 6300 tiles. That's probably not a good idea. You could also use 5m × 5m tiles, which would have fifteen tiles one side side and twenty-one tiles on the other side, which comes out to 252 tiles. You could not use 6m × 6m tiles, however, because if you tried to do this you would have excess tiles hanging over one side of the room. The reason for this is that 6 does not divide 105 cleanly. To minimize the number of tiles used, the ideal solution is to use tiles of size 15m × 15m, which requires four tiles on one side and seven tiles on the other, for a total of only twenty-eight tiles required.

So what is the connection between the two problems? In the problem with the drummers, we have two different numbers (the delay time in-between the drum beats) and want to find the shortest time required before the two drums will beat at the same time. Since each drum beats at a fixed interval, any time at which the drums can beat together must be a multiple of both drum intervals. We therefore want to find the smallest time that is a multiple of both intervals.

In the problem with tilings, note that the size of any square tile must cleanly divide the lengths of both sides of the room. Otherwise, trying to tile the room with tiles of that size would fail, since there would be some extra space overhanging the side of the room. Consequently, we want to find the largest number that cleanly divides both sides of the room.

We can formalize these two intuitions by introducing two new concepts: the *least common multiple* and the *greatest common divisor*. But first, we need to introduce what it means for one number to be a multiple of another, or for one number to divide another.

Let $m, n \in \mathbb{N}$ be natural numbers. We say that $m$ ***divides*** $n$, denoted $m \mid n$, iff there is a natural number $q$ such that $n = qm$. We say that $n$ is a ***multiple*** of $m$ iff $m$ divides $n$.

Intuitively, $m$ divides $n$ means that we can multiply $m$ by some other natural number to get $n$. For example, 2 divides 10 because $10 = 2 \cdot 5$. Similarly, 15 divides 45 because $45 = 3 \cdot 15$. Note that *any* number is a divisor of 0. A quick proof, just to see the definition in action:

Theorem: For any $n \in \mathbb{N}$, we have that $n \mid 0$.

Proof: We need to show that $n \mid 0$, meaning that there is some $q \in \mathbb{N}$ such that $0 = nq$. Take $m = 0$. Then we have that $nq = n \cdot 0 = 0$, as required. ■

This proof isn't the most scintillating argument, but it's good to see exactly how you would structure a proof involving divisibility.

Let's consider any two natural numbers $m$ and $n$. $m$ has some divisors, as does $n$. We might therefore find some $q$ such that $q \mid m$ and $q \mid n$. These numbers are called *common divisors*:

If $m, n \in \mathbb{N}$, then the number $q$ is a ***common divisor*** of $m$ and $n$ iff $q \mid m$ and $q \mid n$.

In our question about tiling the floor of the room, we were searching for a common divisor of the lengths of the sides of the room that was as large as possible. We call this number – which is a common divisor of the room lengths and the largest possible such divisor, the *greatest common divisor* of the room lengths:

If $m, n \in \mathbb{N}$, then the ***greatest common divisor*** of $m$ and $n$, denoted $gcd(m, n)$, is the largest natural number $d$ such that $d \mid m$ and $d \mid n$.

Before we move on and start reasoning about the greatest common divisor, we have to pause and ask ourselves a serious question – how do we know that any two numbers even have a great common divisor in the first place? In case this seems obvious, I'd like to assure you that it's not, and in fact there is a slight error in the above definition.

What would it mean for two numbers $m$ and $n$ to not have a greatest common divisor? There would be two possible options here. First, it might be the case that $m$ and $n$ have no divisors in common at all. In that case, there isn't a "greatest" common divisor, since there wouldn't be any common divisors in the first place! Second, it might be the case that $m$ and $n$ do indeed have some divisors in common, but there are infinitely many such divisors. In that case, no one of them would be the "greatest," since for any divisor we could always find another divisor that was greater than the one we had previously chosen.

In order to show that greatest common divisors actually exist, we will need to show that the two above concerns are not actually anything to worry about. First, let's allay our first concern, namely, that there could be two numbers that have no common divisors at all. To do this, we'll start off by proving the following (hopefully obvious!) result:

***Theorem:*** For any $n \in \mathbb{N}$, we have that $1 \mid n$.

***Proof:*** We need to show that $1 \mid n$, meaning that there is some $q \in \mathbb{N}$ such that $1 \cdot q = n$. Take $q = n$. Then $1 \cdot q = 1 \cdot n = n$, as required. ■

This means that given any $m$ and $n$, there is always at least one divisor in common, namely 1. There could be many more, though, and this brings us to our second concern. Can we find two numbers for which there is no greatest common divisor because there are infinitely many common divisors?

The answer, unfortunately, is yes. Consider this question:

What is $gcd(0, 0)$?

We know from before that *any* natural number $n$ is a divisor of 0. This means that the set of natural numbers that are common divisors of 0 and 0 is just the set of all natural numbers, $\mathbb{N}$. Consequently, there is no one greatest divisor of 0 and 0. If there were some greatest divisor $n$, then we could always pick $n + 1$ as a larger divisor. Consequently, $gcd(0, 0)$ is *undefined*! Is this a bizarre edge case, or are there other pairs of numbers that have no greatest common divisor?

One important property of divisibility is the relative sizes of a number and its divisor. Initially, it might be tempting to say that if $m$ divides $n$, then $m$ must be no bigger than $n$. For example, 5 divides 20, and $5 < 20$, and 137 divides itself, and surely $137 \leq 137$. However, this is not in general true. For example, 137 divides 0, since *every* natural number divides zero, but we know that $137 > 0$.

However, if we treat 0 as a special case, then we have the following result:

***Theorem:*** If $m \mid n$ and $n \neq 0$, then $m \leq n$.

How might we go about proving this? In a sense, this is an "obvious" result, and it seems like we should be able to demonstrate it directly. But doing so ends up being a bit trickier than first glance might suggest. Instead, we'll do a proof by contradiction, showing that if $m \mid n$ (with $n$ not equal to 0) and $m > n$, we can arrive at a contradiction about the value of $n$.

***Proof:*** By contradiction; assume that $m \mid n$, that $n \neq 0$, but that $m > n$. Since $m \mid n$, we know that there must be some $q$ such that $n = qm$. This $q$ cannot be 0, since otherwise we would have that $n = qm = 0 \cdot m = 0$, contradicting the fact that $n \neq 0$. Similarly, this $q$ cannot be 1, since then we would have that $n = qm = 1 \cdot m = m$, which contradicts the fact that $m > n$. So now we have that $q \geq 2$. As one additional detail, note that $m > n$. Since $n \neq 0$, this means that $m \neq 0$ either. We thus have that $m \neq 0$.

Since $m > n$, this means that $qm = n < m$, meaning that $qm < m$. Since $m \neq 0$, we can divide both sides of this inequality by $m$ to get that $q < 1$. But this is impossible, since we know that $q \geq 2$.

We have reached a contradiction, so our assumption must have been wrong. Thus if $m \mid n$ and $n \neq 0$, we must have that $m \leq n$. ■

This proof is a bit more elaborate than our previous proofs. We needed to establish several smaller results along the way – namely, that the quotient q must be at least two, and that m itself could not be zero. Once we have these results, however, the result follows from a simple contradiction.

The reason that this result is important is that it allows us to conclude that there must indeed be a greatest common divisor for any pair of natural numbers other than (0, 0). The reason for this is that for any numbers $m$ and $n$ that are not both identically zero, at least one of these numbers must be nonzero, and thus cannot have any divisors larger than itself. Consequently, one of the numbers $\{1, 2, 3, \ldots, n\}$ must be the greatest common divisor of $m$ and $n$. We're not sure which one it is, but since there are only finitely many numbers to consider, we can guarantee that one of them must be the largest.

But how does the *gcd* relate to our question about drumming? In the case of the drummers, we wanted to find the smallest number that was a multiple of two other numbers. This is called the *least common multiple* of the two numbers:

For any natural numbers $n$ and $m$, a number $k$ is called a ***common multiple*** of $n$ and $m$ if $n \mid k$ and $m \mid k$. The ***least common multiple*** of $m$ and $n$, denoted $lcm(m, n)$, is the smallest of all common multiples of $m$ and $n$.

We are still tasked with proving that the least common multiple of any two numbers $m$ and $n$ actually exist. We'll defer this proof until later, when we've built up a few more mathematical tools. However, we do have the following result:

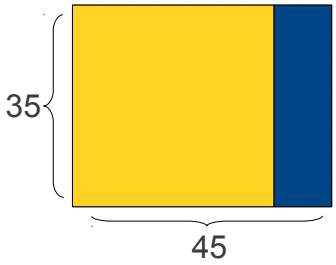
***Theorem:*** For any $m, n \in \mathbb{N}$, where $m$ and $n$ are not both zero, $lcm(m, n) = mn / gcd(m, n)$.

In the interests of time and space, this proof is left as an exercise at the end of this chapter. However, this result does connect the *lcm* and *gcd* of two numbers together. By computing the *gcd* of two numbers, we can compute the *lcm* of those numbers. In other words, we can study the properties of the greatest common divisor in order to study the properties of the least common multiple.

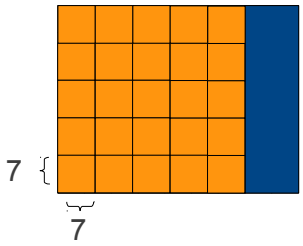
We have now defined *gcd*, but have not provided any way to actually find what the *gcd* of two natural numbers is. This is a step in the right direction – at least we've identified what it is that we're looking for – but otherwise is not particularly useful. To wrap up our treatment of greatest common divisors, let's explore some algorithms that can be used to compute the *gcd* of two natural numbers (that aren't both zero, of course).

To motivate our first algorithm, let's return to the original motivating question we had for the greatest common divisor. Suppose that you have a room whose side lengths are $m$ and $n$. What is the largest size of square tile that you can use to use to tile the room?

This problem dates back to the ancient Greeks, who made a very clever observation about it. Suppose that we have an $m \times n$ rectangle, where $m \geq n$. If we want to find the largest size of the square tiles that we can use to cover the room, one idea is to place a $n \times n$ square tile over the rectangle, flush up against one side of the room. For example, given the rectangle below, which has size $45 \times 35$, we would begin by placing one $35 \times 35$ square tile into the rectangle, like this:

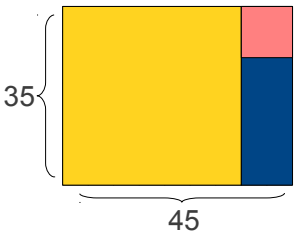



You might wonder why we're doing this – after all, in this case, we can't tile the room with 35×35 tiles! Although this is absolutely true, we can make the following observation. Since any tile that we do use to cover the room must have a side length that divides 35, once we find the actual maximum size of the tile that we're going to use, we can always replace that $35 \times 35$ tile with a bunch of smaller square tiles. For example, if we discovered that the actual tile size is $7 \times 7$ (it isn't, but humor me for a minute), then we could always "retile" the $35 \times 35$ square with $7 \times 7$ tiles like this:

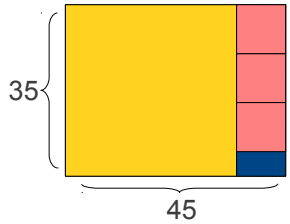



There is one more key observation we can have. After we place down this $35 \times 35$ tile in the room, we're left with a $35 \times 10$ rectangle that isn't tiled. Now, remember that our goal is to find the largest square tile size that we can use to tile the entire room. We just saw that once we've found that tile size, we can always replace the large tile we've placed with a collection of smaller tiles. Consequently, if we think about what the room would look like once we've tiled it appropriately, we know that it must consist of a tiling of the large square we placed down, plus a tiling of the remaining space in the room.

Okay, so how might we find the size of the largest square tiles we could use in the $35 \times 10$ room? Using exactly the same logic as before, we could place a $10 \times 10$ tile in this section of the room, yielding this setup:




But why stop here? We can fit two more $10 \times 10$ tiles in here. If we place these down, we end up with the following:




We're now left with a 10 × 5 rectangle. At this point, we can note that since five cleanly divides ten, we can just drop down two 5 × 5 rectangles into what remains. These are the largest square tiles that we can fit into this 10 × 5 space. Using our previous logic, this means that 5 × 5 tiles are the largest square tiles we can use to tile the 35 × 10 rectangle, and therefore 5 × 5 tiles are the largest square tiles we can use to tile the 45 × 35 rectangle. Consequently, we should have that $gcd(45, 35) = 5$, which indeed it is.

This algorithm is generally attributed to Euclid, the ancient Greek mathematician, and is sometimes called the *Euclidean algorithm*. The rest of this section formalizes exactly how this algorithm works.

To understand Euclid's algorithm in more depth, let us abstract away from rectangles and squares and try to determine mathematically how this algorithm works. Take the above room as an example. Initially, we wanted to compute $gcd(45, 35)$. To do so, we placed a 35 × 35 tile in the room, which left us with a smaller room of size 35 × 10. We then tried to compute the greatest common divisor of those two numbers, $gcd(35, 10)$. Next, we placed three 10 × 10 tiles into the room (the largest number of 10 × 10 tiles that fit), leaving us with a 10 × 5 room.

It seems that the algorithm works according to the following principle: given an $m \times n$ room, subtract out as many copies of $n$ as is possible from $m$. This leaves a room of size $n \times r$, for some natural number $r$. From there, we then subtract out as many copies of $r$ as we can from $n$, leaving us with a room of size $r \times s$ for some natural number $s$. We keep on repeating this process until we end up with a pair of numbers from which we can immediately read off the greatest common divisor.

Now, given a pair of (nonzero) natural numbers $m$ and $n$, where $m \geq n$, what number $r$ do we get when we continuously subtract out $n$ from $m$ until we cannot do so any more? Well, we know that this number $r$ must satisfy $m - nq = r$ for some natural number $q$, since $r$ is what's left after we pull out as many copies of $n$ as we can. We also know that $r$ must satisfy $0 \leq r < n$. The reason for this is simple – if $r < 0$, then we pulled out too many copies of $n$, and if $r \geq n$, then we didn't pull out enough copies.

Let's rewrite the expression from above by moving $nq$ over to the other side. This gives us that $m = nq + r$, where $0 \leq r < n$. With a bit of thought, we can realize that these values $q$ and $r$ are actually meaningful quantities. Specifically, $r$ is the remainder when $m$ is divided by $n$, and $q$ is the (integer) part of the quotient of $m$ and $n$. In other words, this algorithm that works by tiling a rectangle with a lot of squares is really just a fancy way of doing division with remainders!

Before we move on, let's introduce an important theorem:

***Theorem (the division algorithm):*** For any natural numbers $m$ and $n$, with $n \neq 0$, there exist unique natural numbers $q$ and $r$ such that

$m = nq + r$, and
$0 \leq r < n$

Here, $q$ is called the *quotient*, and $r$ is called the *remainder*.

This theorem is called the division algorithm, which is a bit of a misnomer. It's definitely related to division, but there's nothing algorithmic about it. The theorem asserts that there is a unique way to divide two natural numbers to produce a quotient and a remainder. The uniqueness here is important – it's not just that we can do the division, but that there is exactly one way to do this division.

Given that we always compute a quotient and remainder, let's introduce one more piece of terminology:

> For any natural numbers $m$ and $n$, if $n \neq 0$, then the remainder of $m$ when divided by $n$ is denoted $m$ rem $n$. Specifically, $m$ rem $n$ is the unique choice of $r$ guaranteed by the division algorithm such that $m = qn + r$.

For example, 5 rem 3 = 2, and 137 rem 42 = 11. However, 11 rem 0 is not defined. In many programming languages, the remainder operation would be expressed using the % operator.

Now that we have this terminology, we can start to talk about how the Euclidean algorithm actually works. When given $m$ and $n$, the act of adding as many $n \times n$ tiles as possible is equivalent to computing $m$ rem $n$, since we're eliminating as many copies of $n$ from $m$ as follows. In other words, the Euclidean algorithm tries to compute $gcd(m, n)$ by computing $gcd(n, m \text{ rem } n)$.

We already discussed a geometric intuition for why this would work, but can we somehow formalize this argument? It turns out that the answer is "yes," thanks to a clever and ancient theorem.

> ***Theorem:*** For any natural numbers $m$ and $n$, with $n \neq 0$, $gcd(m, n) = gcd(n, m \text{ rem } n)$.

Before proceeding, let's see what this theorem tells us. According to this theorem, we should have that $gcd(105, 60) = gcd(60, 45)$. We should then have that $gcd(60, 45) = gcd(45, 15)$. At this point, we can easily conclude that $gcd(45, 15) = 15$, since 15 cleanly divides 45. This means that $gcd(105, 60) = 15$, which indeed it is.

So how do we prove this theorem? Given just what we know about *gcd* so far, this might initially appear quite tricky. However, there is one nice technique we can use to try to establish this fact. Recall that $gcd(m, n)$ is the largest divisor in the set of all common divisors of $m$ and $n$. If we can show that the pairs $(m, n)$ and $(n, m \text{ rem } n)$ have exactly the same common divisors, it would immediately follow that $gcd(m, n) = gcd(n, m \text{ rem } n)$, since in each case we are taking the largest element out of the same set. Consequently, we can prove that $gcd(m, n) = gcd(n, m \text{ rem } n)$ by proving that any common divisor of $(m, n)$ is a common divisor of $(n, m \text{ rem } n)$ and vice-versa.

***Proof:*** Consider any $m, n \in \mathbb{N}$ with $n \neq 0$. We will prove that any common divisor $d$ of $m$ and $n$ is a common divisor of $n$ and $m$ rem $n$ and vice-versa. From this, the claim that $gcd(m, n) = gcd(n, m \text{ rem } n)$ follows by the definition of *gcd*.

First, we show that any common divisor $d$ of $m$ and $n$ is also a common divisor of $n$ and $m$ rem $n$. Since $d$ is a common divisor of $m$ and $n$, we know that $d \mid m$ and $d \mid n$. Since $d \mid m$, there exists a natural number $q_0$ such that $m = dq_0$. Since $d \mid n$, there exists a natural number $q_1$ such that $n = dq_1$. We need to show that $d \mid n$ and that $d \mid m$ rem $n$. The first of these two claims is immediately satisfied, since we already know $d \mid n$, so we just need to show $d \mid m$ rem $n$, meaning that there is some $q'$ such that $m$ rem $n = dq'$. Using the division algorithm, write $m = nq + m$ rem $n$. We can rewrite this as $m - nq = m$ rem $n$. Since $m = dq_0$ and $n = dq_1$, this means that $dq_0 - dq_1q = m$ rem $n$, meaning that $d(q_0 - q_1q) = m$ rem $n$. Taking $q' = q_0 - q_1q$, we have that $m$ rem $n = dq'$, so $d \mid m$ rem $n$ as required.

Next, we show that any common divisor $d$ of $n$ and $m$ rem $n$ is a common divisor of $m$ and $n$. Since $d$ is a common divisor of $n$ and $m$ rem $n$, we know that $d \mid n$ and $d \mid m$ rem $n$. We need to show that $d \mid m$ and $d \mid n$. This second claim we already know to be true, so we just need to prove the first. Now, since $d \mid n$ and since $d \mid m$ rem $n$, there exist natural numbers $q_0$ and $q_1$ such that $n = dq_0$ and $m$ rem $n = dq_1$. To show that $d \mid m$, we need to show that there is a $q'$ such that $m = dq'$. Using the division algorithm, write $m = nq + m$ rem $n$. Consequently, we have that $m = dq_0q + dq_1 = d(q_0q + q_1)$. Taking $q' = q_0q + q_1$, we have that $m = dq'$, so $d \mid m$ as required. ■

This proof is a bit long, but conceptually is not very difficult. We keep applying definitions in each case to write two of $m$, $n$, and $m$ rem $n$ as multiples of $d$, then use the division algorithm to show that the third is a multiple of $d$ as well.

We now have a theorem that says that $gcd(m, n) = gcd(n, m \text{ rem } n)$, assuming that $n \neq 0$. However, we haven't discussed what happens when $n = 0$. In that case, we are trying to compute $gcd(m, 0)$. If $m = 0$, then this is mathematically undefined. However, if $m \neq 0$, then this is mathematically legal. In fact, we have that $gcd(m, 0) = m$, since $m$ is the largest number that divides $m$, and any number divides 0. Let's quickly formalize this:

***Theorem:*** For any $m \in \mathbb{N}^+$, $gcd(m, 0) = m$.

***Proof:*** Let $m$ be any arbitrary positive natural number. Then $gcd(m, 0)$ is the largest common divisor of $m$ and 0. We know that $m \mid m$, since $m = 1 \cdot m$, and by our previous result all divisors of $m$ must be no greater than $m$. Thus $m$ is the greatest divisor of $m$. Since we also know that $m \mid 0$, $m$ is a common divisor of $m$ and 0, and there are no greater common divisors. Thus $gcd(m, 0) = m$. ■

We now have two key theorems about *gcd*. The first one says that the *gcd* stays the same after you compute the remainder of the two arguments. The second one says that once we reduce the second argument to 0, we know that the *gcd* is just the first value. This means that we can finally introduce a description of the Euclidean algorithm. Consider the following function:

```
int euclideanGCD(int m, int n) {
    if (n == 0) return m;
    return euclideanGCD(n, m rem n);
}
```

This recursive algorithm computes *gcd*(*m*, *n*), using the above lemma to continuously simplify the expression. For example, suppose that we want to compute the *gcd* of two very large numbers, say, 32,340 and 10,010. Using the above code, we have the following:

```
  euclideanGCD(32340, 10010)
= euclideanGCD(10010, 2310)
= euclideanGCD(2310,  770)
= euclideanGCD(770,   0)
= 770
```

And indeed, 770 is the *gcd* of the two numbers.

Now, how would we prove that this algorithm is actually correct? As you might have suspected, we're going to use induction. However, doing so seems tricky in this case – there are now two parameters to the function, *m* and *n*, but induction only works on one variable. The key observation we can have is that the second parameter to this function continuously gets smaller and smaller as the recursion progresses. The reason for this is that whenever we compute the remainder *m* rem *n*, we end up with a value that is strictly less than *n*. Consequently, we will prove the following claim: for any natural number *n*, the algorithm works regardless of which *m* you choose. We can prove that *this* claim is true by induction on *n*. The logic will be the following:

- When $n = 0$, the algorithm works regardless of which *m* you choose.
- When $n = 1$, the algorithm works regardless of which *m* you choose.
- When $n = 2$, the algorithm works regardless of which *m* you choose.
- etc.

In the course of doing so, we'll use the modified version of strong induction that we developed in the previous section. The resulting proof is remarkably short, and is given here:

***Theorem:*** For any $m, n \in \mathbb{N}$, if *m* and *n* are not both zero, then **euclideanGCD**(*m*, *n*) = *gcd*(*m*, *n*).

***Proof:*** By strong induction. Let $P(n)$ be "for any $m \in \mathbb{N}$, if *m* and *n* are not both zero, then **euclideanGCD**(*m*, *n*) = *gcd*(*m*, *n*)." We will prove that $P(n)$ holds for all $n \in \mathbb{N}$ by strong induction.

Assume that for some $n \in \mathbb{N}$, that for all $n' \in \mathbb{N}$ with $n' < n$, that $P(n')$ holds, so for any *m*, if *m* and $n'$ are not both zero, then **euclideanGCD**(*m*, $n'$) = *gcd*(*m*, $n'$). We will prove $P(n)$, that for any *m*, if *m* and *n* are not both zero, then **euclideanGCD**(*m*, *n*) = *gcd*(*m*, *n*).

First, we consider the case where $n = 0$. In this case, for any $m \in \mathbb{N}^+$, we have that **euclideanGCD**(*m*, *n*) = *m* = *gcd*(*m*, *n*). Thus $P(n)$ holds.

Otherwise, $n > 0$. Then for any $m \in \mathbb{N}$, we have that **euclideanGCD**(*m*, *n*) = **euclideanGCD**(*n*, *m* rem *n*). Since *m* rem *n* satisfies $0 \leq m \text{ rem } n < n$, by our inductive hypothesis we have that **euclideanGCD**(*n*, *m* rem *n*) = *gcd*(*n*, *m* rem *n*). By our earlier theorem, we know that *gcd*(*n*, *m* rem *n*) = *gcd*(*m*, *n*). Consequently, **euclideanGCD**(*m*, *n*) = *gcd*(*m*, *n*). Thus $P(n)$ holds, completing the induction. ■

### 3.5.4 Why Strong Induction Works ★

Before we conclude this section on strong induction, we should go over exactly why strong induction works. We wrote proofs to show that we could use Fibonacci induction or induction starting from some number $k$, and it would be a serious asymmetry to omit the proof that we can indeed use strong induction in the first place.

Our goal will be to prove the following theorem, which represents the simplest version of strong induction:

> ***Theorem:*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:
>
> $P(0)$ is true.
> For any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true,
> then $P(n)$ is true.
>
> Then for any $n \in \mathbb{N}$, $P(n)$ is true.

How exactly can we do this? As before, our goal will be to invent some new property $Q(n)$ such that $Q(n)$ can be proven using normal (not strong) induction, and which has the property that if $Q(n)$ is true for all $n \in \mathbb{N}$, then $P(n)$ is true for all $n \in \mathbb{N}$.

The key difference between normal induction and strong induction is what we carry along with us in the inductive step. In normal induction, we just remember the last result we have proven. In strong induction, we remember all the results that we have proven so far. Given this, one idea for how we might pick $Q(n)$ is the following. What if we choose $Q(n)$ to mean "$P(n')$ is true for all $n' \leq n$"?* In this case, the statement that $Q(n)$ is true just for some particular $n$ means that we still remember all of the previous $P(n)$ results.

Given this, the proof is actually quite simple:

* Yes, I know the ? should be in the double-quotes, but that just looks really weird here!

***Theorem:*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:

$P(0)$ is true.
For any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true,
then $P(n)$ is true.

Then for any $n \in \mathbb{N}$, $P(n)$ is true.

***Proof:*** Let $P(n)$ be any property satisfying the requisite conditions. Define the property $Q(n) \equiv$ "for all $n' \in \mathbb{N}$ with $n' \leq n$, $P(n')$ holds." This proof proceeds in two parts. First, we will prove that $Q(n)$ holds for all $n \in \mathbb{N}$ by induction. Second, we will prove that if $Q(n)$ holds for all $n \in \mathbb{N}$, then $P(n)$ holds for all $n \in \mathbb{N}$.

First, we prove that $Q(n)$ holds for all $n \in \mathbb{N}$ by induction. For our base case, we prove $Q(0)$, that for all $n' \in \mathbb{N}$ with $n' \leq 0$, $P(n')$ holds. Since the only natural number $n' \leq 0$ is 0 itself, this means that we need to show that $P(0)$ holds. By our choice of $P$, we know this to be true, so $Q(0)$ holds.

For the inductive step, assume that for some $n \in \mathbb{N}$, that $Q(n)$ holds, meaning that for all $n' \in \mathbb{N}$ with $n' \leq n$, $P(n)$ holds. We want to prove that $Q(n + 1)$ holds, meaning that for all $n' \in \mathbb{N}$ with $n' \leq n + 1$, $P(n)$ holds. Under the assumption that $Q(n)$ is true, we already know that $P(n')$ holds for all $n' \leq n$, so we only need to show that $P(n + 1)$ holds. By our choice of $P$, we know that since $P(n')$ holds for all $n' \leq n$, it must be true that $P(n + 1)$ holds. Thus $Q(n + 1)$ holds, completing the induction.

Finally, we need to show that since $Q(n)$ holds for all $n \in \mathbb{N}$, that $P(n)$ holds for any $n \in \mathbb{N}$. To do this, consider any $n \in \mathbb{N}$. Since $Q(n)$ holds, this means that $P(n')$ holds for all $n' \leq n$. In particular, this means that $P(n)$ holds, since $n \leq n$. Since our choice of $n$ was arbitrary, this means that $P(n)$ holds for all $n \in \mathbb{N}$, as required. ■

# 3.6 The Well-Ordering Principle ★

To wrap up our treatment of induction, let's explore something that at face value has nothing to do with induction.

The following fact might seem obvious, but it's actually surprisingly subtle:

***Theorem (the well-ordering principle):*** Any nonempty set of natural numbers has a least element.

The well-ordering principle says that if you take any set of natural numbers (that is, a set $S \subseteq \mathbb{N}$), then that set contains a least element (some natural number smaller than all the other natural numbers in the set. For example, the set { 0, 1, 2, 3 } has least element 0, while the set { $n \in \mathbb{N}$ | $n$ is a prime number } has least element 2. However, the well-ordering principle doesn't guarantee anything about $\varnothing$, since the empty set is (unsurprisingly!) empty. It also says nothing about $\mathbb{R}$, because $\mathbb{R}$ is not a set of natural numbers.

The subtle aspect of this theorem is that it applies to *all* sets of natural numbers, including infinite sets. This means that if we are ever working with infinite sets of natural numbers, we can always speak of the least element of the set, since it's guaranteed to exist.

### 3.6.1 Proof by Infinite Descent

What's truly amazing about the well-ordering principle is that *all* of the inductive proofs we have done in this chapter can be rewritten using the well-ordering principle, rather than using induction. How can this be? To prove a property using the well-ordering principle, you can use the following setup:

1. Define your property $P(n)$ to be proven true for all $n \in \mathbb{N}$.
2. Consider the set $S = \{ n \in \mathbb{N} \mid P(n) \text{ is false} \}$ of all natural numbers for which $P(n)$ is not true. Note that if $S$ is empty, then $P(n)$ is true for all natural numbers.
3. Assume, for the sake of contradiction, that $S$ is nonempty. It therefore contains a least element, which we will call $n_0$. Note that because $n_0$ is the least element for which $P(n)$ is false, we are guaranteed that $P(n)$ is true for all $n < n_0$.
4. Using the fact that $n_0$ is the least natural number for which $P(n)$ is false, derive a contradiction. This means that our assumption is wrong, so $S$ must be empty, and therefore $P(n)$ is true for all $n \in \mathbb{N}$.

This might seem somewhat convoluted, but in many cases these proofs can be quite clean. As an example, let's prove a result that we already know is true: that the sum of the first $n$ powers of two is $2^{n+1} - 1$. We have proven this before using telescoping series, which indirectly relied on induction, which gives a very different flavor of proof from the one below.

***Theorem:*** $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

***Proof:*** Consider the set $S = \{ n \in \mathbb{N} \mid \sum_{i=0}^{n} 2^i \neq 2^{n+1} - 1 \}$. If this set $S$ is empty, then it must be true that $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all $n \in \mathbb{N}$, from which the theorem follows. So assume for the sake of contradiction that $S$ is nonempty. Since $S$ is a nonempty set of natural numbers, by the well-ordering principle it has a least element; let this element be $n_0$. Now, either $n_0 = 0$, or $n_0 \geq 1$. We consider these cases separately.

First, suppose that $n_0 = 0$. But we can check that $\sum_{i=0}^{0} 2^i = 2^0 = 1 = 2^1 - 1$, meaning that $n_0 \notin S$, contradicting our initial assumption.

So it must be the case that $n_0 \geq 1$. Since the sum $\sum_{i=0}^{n_0} 2^i$ is nonempty, we know that $\sum_{i=0}^{n_0} 2^i = \sum_{i=0}^{n_0-1} 2^i + 2^{n_0}$. Thus $\sum_{i=0}^{n_0-1} 2^i + 2^{n_0} \neq 2^{n_0+1} - 1$. Consequently, $\sum_{i=0}^{n_0-1} 2^i \neq 2^{n_0+1} - 2^{n_0} - 1$. Since we have that $2^{n_0+1} - 2^{n_0} - 1 = 2^{n_0} - 1$, this means that $\sum_{i=0}^{n_0-1} 2^i \neq 2^{n_0} - 1$. But since $n_0 \geq 1$, this means that $n_0 - 1 \geq 0$, so $n_0 - 1$ is a natural number. Since $n_0 - 1 \in \mathbb{N}$ and $\sum_{i=0}^{n_0-1} 2^i \neq 2^{n_0} - 1$, this means that $n_0 - 1 \in S$, contradicting the fact that $n_0$ is the least element of $S$. We have reached a contradiction, so our assumption must have been wrong and $S$ is empty. ■

This proof is in many ways quite subtle, and deserves a closer look.

The key idea behind the proof is the following idea – we find the set of all counterexamples to the theorem, then show that it must be empty. To do so, we consider what happens if it is nonempty. If the set is nonempty, then the well-ordering principle guarantees us that it must have a least element. We can then think about what this least element is. Intuitively, this element cannot exist for the following reason – if it did, we could peel off the last term of the sum, then show that the number that comes before the least element *also* must be a counterexample to the theorem. In other words, we started with the least element, then showed that there would have to be an element that's even smaller than it. This style of proof is sometimes called a *proof by infinite descent*, since we show that we can always keep making a counterexample smaller and smaller.

Of course, before we do this, we have to check that $n_0$ is not zero. If $n_0$ is zero, then the fact that the claim is not true for $n_0 - 1$ is meaningless, since $n_0 - 1 = -1$, which is not a natural number. Consequently, it can't be contained in the set $S$ in the first place. We thus have to split our proof into two cases, one where $n_0$ is zero, and one where it is nonzero. Here, we have a strong parallel to proof by induction – our "base case" handles the case where $n_0$ is zero, and our "inductive step" works when $n_0$ is greater than zero.

Typically, when writing a proof that uses the well-ordering principle, we would not actually explicitly construct the set $S$. Instead, we would just say that, if the theorem were false, there would be some counterexample, and consequently we could consider the smallest counterexample. Here is an alternative version of the above proof, which is more condensed but still mathematically rigorous:

***Proof:*** By contradiction; assume that there is some $n \in \mathbb{N}$ such that $\sum_{i=0}^{n} 2^i \neq 2^{n+1} - 1$. Since there is at least one $n$ for which this is true, there must be a smallest $n_0 \in \mathbb{N}$ for which $\sum_{i=0}^{n_0} 2^i \neq 2^{n_0+1} - 1$. Now, either $n_0 = 0$, or $n_0 \geq 1$. We consider each case separately.

First, suppose that $n_0 = 0$. But we can check that $\sum_{i=0}^{0} 2^i = 2^0 = 1 = 2^1 - 1$, contradicting our initial assumption.

So it must be the case that $n_0 \geq 1$. Since the sum $\sum_{i=0}^{n_0} 2^i$ is nonempty, we know that $\sum_{i=0}^{n_0} 2^i = \sum_{i=0}^{n_0-1} 2^i + 2^{n_0}$. Thus $\sum_{i=0}^{n_0-1} 2^i + 2^{n_0} \neq 2^{n_0+1} - 1$. Consequently, $\sum_{i=0}^{n_0-1} 2^i \neq 2^{n_0+1} - 2^{n_0} - 1$. Since we have that $2^{n_0+1} - 2^{n_0} - 1 = 2^{n_0} - 1$, this means that $\sum_{i=0}^{n_0-1} 2^i \neq 2^{n_0} - 1$. But since $n_0 \geq 1$, this means that $n_0 - 1 \geq 0$, so $n_0 - 1$ is a natural number. This contradicts the fact that $n_0$ is the smallest natural number for which $\sum_{i=0}^{n_0} 2^i \neq 2^{n_0+1} - 1$. We have reached a contradiction, so our assumption must have been wrong, so $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$ for all $n \in \mathbb{N}$. ■

At this point, the well-ordering principle might not seem particularly useful – after all, we already knew of two different ways to prove the above result (telescoping series and a direct proof by induction). The real strength of the well-ordering principle shows up in certain sorts of proof by contradiction that might otherwise feel hand-wavy or insufficiently rigorous.

As an example, let's return to one of the proofs we did in the previous chapter – that the square root of two is irrational. (I would strongly suggest reviewing that proof at this point before proceeding; we're going to be talking about it in some detail). If you'll recall, we said that a nonnegative number $r$ is rational iff there exists natural numbers $p$ and $q$ such that

- $q \neq 0$,
- $p / q = r$, and
- $p$ and $q$ have no common divisors other than 1 and -1.

Of these three restrictions, the first two seem essential to the definition – after all, we're trying to write $r$ as the ratio of two other numbers – but the third seems suspicious. Why is it that we need to have this restriction? Why does it matter if $p$ and $q$ have any common factors?

Let's see what happens if we completely drop this restriction. Let's say that a nonnegative number $r$ is rational if we can write $r = p / q$, where $q \neq 0$ and $p, q \in \mathbb{N}$. If we use this new definition at face value, then our previous proof that the square root of two is irrational breaks down. That proof worked by showing that if we can ever write $\sqrt{2} = p / q$, then it would have to be that both $p$ and $q$ have two as a common factor, resulting in our contradiction. But without this extra clause in our definition, we can't use this proof anymore. We no longer get a contradiction.

However, now that we have the well-ordering principle in place, we actually can prove that $\sqrt{2}$ is irrational even if we drop the last restriction from our definition. The proof will be similar to the one we did in Chapter 2. Specifically, we will assume that $\sqrt{2} = p / q$, then show that both $p$ and $q$ must have two as a com-

mon factor. From this, we get that both $p / 2$ and $q / 2$ must be natural numbers. However, we know that since $q \neq 0$, $q / 2$ is strictly less than $q$. This means that if we can ever find a choice of $p$ and $q$ for which $p / q = \sqrt{2}$, then we can find a smaller choice of $p$ and $q$ that works as well. But this is impossible, since eventually we'll find the smallest choice of $p$ and $q$, and we won't be able to make them any smaller.

We can formalize this intuition using the well-ordering principle. We'll assume, for the sake of contradiction, that $\sqrt{2} = p / q$ for natural numbers $p$ and $q$, with $q \neq 0$. Since there is at least one choice of $q$ that works here, there must be a smallest such choice of $q$ that works. If we then start off with $p$ and $q$ being as small as possible and show that we can keep making them smaller, we will have arrived at the contradiction we desire.

Using the well-ordering principle, here is a succinct proof that $\sqrt{2}$ is irrational, without relying on the "no common factors" definition of rationality.

***Proof:*** By contradiction; assume that $\sqrt{2}$ is rational. Then there exists integers $p$ and $q$ such that $q \neq 0$ and $p / q = \sqrt{2}$. Since there is at least one natural number $q$ that acts as a denominator in this case, there must be a least such $q$. Call this number $q_0$, and let $p_0$ be the natural number such that $p_0 / q_0 = \sqrt{2}$. Note that in this case, $q_0 \neq 0$.

Since $p_0 / q_0 = \sqrt{2}$, this means that $p_0^2 / q_0^2 = 2$, which means that $p_0^2 = 2q_0^2$. This means that $p_0^2$ is even, so by our earlier result $p_0$ must be even as well. Consequently, there exists some integer $k$ such that $p_0 = 2k$.

Since $p_0 = 2k$, we have that $2q_0^2 = p_0^2 = (2k)^2 = 4k^2$, so $q_0^2 = 2k^2$. This means that $q_0^2$ is even, so by our earlier result $q_0$ must be even as well. Since both $p_0$ and $q_0$ are even, this means that $p_0 / 2$ and $q_0 / 2$ are natural numbers, and we have that $(p_0 / 2) / (q_0 / 2) = p_0 / q_0 = \sqrt{2}$. But since $q_0 \neq 0$, we know that $q_0 / 2 < q_0$, contradicting the fact that $q_0$ is the least denominator in a rational representation of $\sqrt{2}$.
We have reached a contradiction, so our assumption must have been incorrect. Thus $\sqrt{2}$ is irrational. ■

## 3.6.2 Proving the Well-Ordering Principle

To finalize our treatment of the well-ordering principle, let us take a few minutes to prove that it is true. It turns out that this proof is not as obvious as it might seem, and in fact working through this proof will shed some new light on what induction can and cannot prove.

### 3.6.2.1 An Incorrect Proof

One initial idea that we might consider as a proof technique here would be the following. Why don't we show, by induction, that any set of natural numbers of size 1, 2, 3, …, etc. contains a least element? Our proof would work as follows – we'll start off by showing that any set of one natural number contains a least element, then will proceed by induction to show that if any set of $n$ natural numbers contains a least element, then any set of $n + 1$ natural numbers has a least element.

**Unfortunately, the above proof technique does not work**. The reason is extremely subtle. At face value, it might seem like this should work out just great. Below is an **incorrect proof** that proceeds along these lines. Although it's completely incorrect, you should look over it anyway. The next part of this section goes over exactly what's wrong with it.

***Incorrect Proof:*** By induction. Let $P(n)$ be "Any set $S$ of $n$ natural numbers contains a least element." We will prove that $n$ holds for all $n \in \mathbb{N}^+$ by induction on $n$.

As our base case, we prove $P(1)$, that any set $S$ of one natural number contains a least element. To see this, let $S = \{ k \}$ be an arbitrary set of one natural number. Then $k$ is the least element of $S$, since it is the only such element.

For our inductive step, assume that for some $n \in \mathbb{N}^+$ that $P(n)$ holds and any set of $n$ natural numbers has a least element. We will prove $P(n + 1)$, that any set of $n + 1$ natural numbers has a least element. To do this, consider any set $S \subseteq \mathbb{N}$ of $n + 1$ natural numbers. Choose some natural number $k \in S$ and let $S' = S - \{k\}$. Then $S'$ has size $n$ and is a set of natural numbers, so by our inductive hypothesis $S$ contains a least element, let it be $r$. Now, since $S = S' \cup \{ k \}$ and $k \notin S'$, we know that $k \neq r$. Thus either $r < k$ or $k < r$. If $r < k$, then $r$ is the least element of $S$, since it is smaller than all other elements of $S'$ and is smaller than $k$. Otherwise, if $k < r$, then $k$ is the least element of $S$, since $k < r$ and $r$ is smaller than all other elements of $S'$. Thus $S$ contains a least element, so $P(n + 1)$ holds, as required. ■

This proof is deceptive. It looks like it should be absolutely correct, but this proof is completely and totally wrong. The problem with it is very subtle, and rather than ruining the surprise, let's walk through it and try to see if we can figure out what's wrong.

One indicator that this proof might be wrong is the following. We know that while any set of natural numbers must have a least element, some sets of integers or real numbers might not. For example, the set $\mathbb{Z}$ itself has no least element, nor does the set $\mathbb{R}$. The set $\{ x \in \mathbb{R} \mid x > 0 \}$ also has no least element. However, we can easily adapt the above proof to show that, allegedly, any set of real numbers or integers must have a least element! In fact, here's the modified proof:

***Incorrect Proof:*** By induction. Let $P(n)$ be "Any set $S$ of $n$ {integers, real numbers} contains a least element." We will prove that $n$ holds for all $n \in \mathbb{N}^+$ by induction on $n$.

As our base case, we prove $P(1)$, that any set $S$ of one {integer, real number} contains a least element. To see this, let $S = \{ k \}$ be an arbitrary set of one {integer, real number}. Then $k$ is the least element of $S$, since it is the only such element.

For our inductive step, assume that for some $n \in \mathbb{N}^+$ that $P(n)$ holds and any set of $n$ {integers, real numbers} has a least element. We will prove $P(n + 1)$, that any set of $n + 1$ {integers, real numbers} has a least element. To do this, consider any set $S \subseteq \mathbb{N}$ of $n + 1$ {integers, real numbers}. Choose some {integers, real numbers} $k \in S$ and let $S' = S - \{k\}$. Then $S'$ has size $n$ and is a set of {integers, real numbers}, so by our inductive hypothesis $S$ contains a least element, let it be $r$. Now, since $S = S' \cup \{ k \}$ and $k \notin S'$, we know that $k \neq r$. Thus either $r < k$ or $k < r$. If $r < k$, then $r$ is the least element of $S$, since it is smaller than all other elements of $S'$ and is smaller than $k$. Otherwise, if $k < r$, then $k$ is the least element of $S$, since $k < r$ and $r$ is smaller than all other elements of $S'$. Thus $S$ contains a least element, so $P(n + 1)$ holds, as required. ■

This can't be right. Something must be amiss here.

The problem with these two "proofs" has to do with how induction works. Recall that induction works as follows – if we can prove $P(0)$ (or, in this case, $P(1)$) and that $P(n) \to P(n + 1)$ for all $n \in \mathbb{N}$, then we can

conclude that $P(n)$ holds for all $n \in \mathbb{N}$ (or, in this case, all $n \in \mathbb{N}^+$). What this means is that the above two "proofs" have demonstrated the following:

For any $n \in \mathbb{N}$, any set $S$ of { natural numbers, integers, real numbers }
of size $n$ has a least element.

Notice that this statement is *not* the same as the following:

Any set $S$ of { natural numbers, integers, real numbers } has a least element.

The subtle difference here is that the first statement only applies to sets whose sizes are natural numbers. In other words, it only applies to *finite sets*. The second statement, on the other hand, applies to *all* sets, whether or not that set is infinite. In other words, both of the above proofs by induction correctly demonstrate that any *finite* set of natural numbers, integers, or real numbers has a least element. However, they do not say anything about sets of infinite size, because those sets do not have size equal to any natural number.

When writing out proofs by induction, be careful to remember that you are only proving that your property $P(n)$ holds for all natural numbers $n$. This means that you are proving a result that holds for infinitely many different choices of $n$, but for which each choice of $n$ is itself finite. This might seem counterintuitive at first, so be sure to think about why this is. We will dedicate more time to playing around with infinity in a later chapter, but for now make sure you understand the distinction between "it is true for infinitely many different values" and "it is true for infinite values."

#### 3.6.2.2 A Correct Proof

Given that our previous proof completely failed, how exactly would we prove that the well-ordering principle is true? We will need to find a proof that works for both finite and infinite sets, meaning that we cannot use induction on the size of the sets in order to complete our proof.

Perhaps we could try out a proof by contradiction. Let's suppose that there is a nonempty set $S$ of natural numbers that doesn't contain a least element. In that case, what could this set contain? Well, it certainly couldn't contain 0, because if it did, 0 would be the least element (because $0 \leq n$ for all $n \in \mathbb{N}$). Given that $0 \notin S$, we also know that $1 \notin S$ either, because if $S$ contained 1, then 1 would be the least element (since the only possible smaller value is 0, which we know isn't in $S$). Given that $0 \notin S$ and $1 \notin S$, we would then get that $2 \notin S$, since if $2 \in S$, it would be the least element (because $0 \notin S$ and $1 \notin S$). We can then just repeat this logic over and over again to show that there can't be *any* natural numbers in $S$, since if there were, one of them would have to be the least element. This intuition can actually be formalized as a proof, which is given below:

***Theorem:*** Every nonempty set of natural numbers contain a least element.

***Proof:*** By contradiction; assume there is a nonempty set $S \subseteq \mathbb{N}$ such that $S$ has no least element. We will derive a contradiction by proving that, in this case, $S$ must be empty. To do so, let $P(n)$ be "$n \notin S$." We will prove that $P(n)$ is true for all $n \in \mathbb{N}$, meaning that $S$ does not contain any natural numbers. Since $S \subseteq \mathbb{N}$, but no natural numbers are in $S$, this means that $S = \varnothing$, a contradiction.

To prove that $P(n)$ is true for all $n \in \mathbb{N}$, we proceed by strong induction. As our base case, we prove $P(0)$, that $0 \notin S$. To see this, note that if $0 \in S$, then 0 would be the least element of $S$, since $0 \leq n$ for all $n \in \mathbb{N}$, contradicting the fact that $S$ has no least element.

For the inductive step, assume that for some $n \in \mathbb{N}$, that for all $n' \in \mathbb{N}$ with $0 \leq n' \leq n$, we have that $P(n')$ holds and $n' \notin S$. We now prove $P(n + 1)$, meaning that $n + 1 \notin S$. To see this, suppose for the sake of contradiction that $n + 1 \in S$. Note that the only way that there could be a smaller element of $S$ would be if some value $n'$ satisfying $n' < n + 1$ is contained in $S$. Since all $n'$ satisfying $n' < n + 1$ also satisfies $n' \leq n$, this means that, by the inductive hypothesis, we know that $n' \notin S$. Consequently, there are no smaller natural numbers in $S$, and therefore $n + 1$ is the least element of $S$. But this is impossible, since $S$ has no least element. We therefore have a contradiction, so $n + 1 \notin S$, and $P(n + 1)$ holds, completing the induction. ■

# 3.7 Chapter Summary

- The *principle of mathematical induction* states that if a property holds for 0, and if whenever that property holds for $n$ it holds for $n + 1$ as well, then the property holds for all natural numbers.
- The *base case* of an inductive proof is the proof that the claim holds for 0 (that is, that $P(0)$ is true). The *inductive step* of an inductive proof is the assumption that $P(n)$ holds for some $n \in \mathbb{N}$, then proving that $P(n + 1)$ holds. The assumption of $P(n)$ is called the *inductive hypothesis*.
- *Summation notation* can be used to compactly represent a sum of multiple values. Summations are closely connected to induction.
- The *empty sum* of no numbers is 0. The empty product of no numbers is 1.
- The sum of the first $n$ natural numbers is $n(n - 1) / 2$.
- It is possible to simplify many sums into a closed-form by splitting or joining the sums together.
- A *telescoping series* is a sum of differences of terms in a sequence. They can be used to determine the values of many unknown sums.
- Induction and recursion are closely connected. Recursive functions can be formally verified with an inductive proof.
- *Monoids* are binary operations that are associative and have an identity element. Recursion makes it possible to compute the *fold* of a monoid, which can be formally proven correct with induction.
- Induction can be started from any natural number, not just 0.

- *Fibonacci induction* makes it possible to prove more complex results with induction.
- *Strong induction* allows us to strengthen our inductive hypothesis by assuming that the property $P$ holds for all numbers smaller than some value, not just for that value itself.
- The *Euclidean algorithm* can be used to compute the greatest common divisor of two natural numbers, assuming both of the numbers are not zero.
- The *well-ordering principle* states that any nonempty subset of natural numbers has a least element. This principle can be used in place of induction if desired.

# 3.8 Chapter Exercises

1. Consider the sum of $k$ odd natural numbers $n_1$, …, $n_k$. Prove that the parity of this sum is equal to the parity of $k$.
2. Consider the product of $k$ natural numbers $n_1$, $n_2$, …, $n_k$. Prove that the parity of this product is odd iff each of the $n_i$'s are.
3. Using the previous two results, prove that $\log_2 3$ is irrational. ★
4. Let's look at the parities of the values in the Fibonacci sequence. Notice that the terms 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, … follow the pattern of even, odd, odd, even, odd, odd, even, odd, odd, etc. In other words, $F_n$ is even iff $n$ is a multiple of three.

   Prove, by induction, that $F_n$ is even iff $3 \mid n$.
5. A ***triangular number*** is a number of the form $n(n + 1) / 2$. Intuitively, triangular numbers are numbers that represent the sum of the first $n$ positive natural numbers, since those sums can be visualized as a triangle of $n(n + 1) / 2$ points. We'll denote the $n$th triangular number using the notation $\Delta_n = n(n + 1) / 2$. The first triangular numbers are 0, 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, …

   Prove that $\Delta_n$ is even iff $n$ leaves a remainder of 1 or 2 when divided by four.
6. Prove, by induction, that for any finite set $A$, that $|\wp(A)| = 2^{|A|}$.
7. Prove that the sum of the first $n$ Fibonacci numbers is $F_{n+1} - 1$. Do this proof twice – once using Fibonacci induction, and once using the sum of a telescoping series.
8. Find a formula for the sum of the first $n$ Leonardo numbers, then prove that it is correct.
9. Find a formula for the sum of the first $n$ fourth powers ($0^4 + 1^4 + 2^4 + \ldots (n - 1)^4$). Prove that it is correct.
10. For $n \geq 0$, what is $L_{n+3} - L_{n+2}$? Using this fact, find another proof that $L_n = 2F_{n+1} - 1$.
11. Suppose that we change the recursion in **fib** by adding a base case that has **fib** return 1 if the input is 2. With this new definition of **fib**, how many calls are necessary to evaluate **fib**($n$)?
12. Prove by induction that for any $m$, $n \in \mathbb{N}$, that $m!n! \leq (m + n)!$. Explain, intuitively, why this is.
13. Prove that for any natural numbers $m$ and $n$, that $lcm(m, n)$ exists. That is, there is some natural number that is a common multiple of $m$ and $n$, and that it is the least such number.
14. ***Binet's formula*** is a remarkable result about Fibonacci numbers. Specifically, it states that

$$F_n = \frac{1}{\sqrt{5}}\left(\varphi^n - (1 - \varphi)^n\right)$$

Here, φ is the *golden ratio*, $\frac{1+\sqrt{5}}{2}$. Using Fibonacci induction, prove Binet's formula.

15. Prove that every nonzero natural number is the product of prime numbers, numbers with no divisors other than one and themselves.

16. ***Euclid's lemma*** states that for any prime number $p$, if $p \mid mn$ for natural numbers $m$ and $n$, then either $p \mid m$ or $p \mid n$. Using Euclid's lemma, prove the ***fundamental theorem of arithmetic***, that every nonzero natural number can be written *uniquely* as the product of prime numbers.

17. Using the fundamental theorem of arithmetic, prove that for any natural numbers $m$ and $n$ such that $m$ and $n$ are not both zero, that $lcm(m, n) = mn / gcd(m, n)$. ★

18. Formally prove that it is legal to start strong induction at any number $k$, not just zero, using a proof along the lines of the one used to prove that we can start off regular induction at any number, not just 0.

19. Suppose that there is a property $P(x)$ of *integers* such that $P(0)$ holds and, for any $x \in \mathbb{Z}$, $P(x) \to P(x + 1)$ and $P(x) \to P(x - 1)$. Prove that in this case, $P(x)$ holds for all $x \in \mathbb{Z}$.

20. Suppose that there is a property $P(n)$ of natural numbers such that $P(0)$ holds and, for any $n \in \mathbb{N}$, $P(n) \to P(2n)$ and $P(n) \to P(2n + 1)$. Prove that in this case, $P(n)$ holds for all $n \in \mathbb{N}$.

21. Suppose that we modify the unstacking game so that your score is given by the *sum* of the two heights of the towers formed. Do you always get the same score in this case? If so, what is that score? Prove that your answer is correct. If not, what is the optimal strategy, and how many points do you get? Prove that your answer is correct.

22. Suppose that you have an $m \times n$ rectangular candy bar formed from $mn$ smaller squares. How many breaks are necessary to break the candy bar down into all its constituent pieces? Prove that your answer is correct.

23. Generalize your result from the previous question to the case where you have a three-dimensional block of chocolate formed from cubes of chocolate.

24. Suppose that you have a linear candy bar with $n + 1$ squares in it and you want to break it down into its constituent pieces. However, this time you are allowed to break multiple pieces of the candy bar at the same time. For example, if you had a piece of size two and a piece of size three, with one break you could break the piece of size two into two pieces of size one and the piece of size three into one piece of size one and one piece of size three. In that case, what is the minimum number of breaks required to split the chocolate bar into its constituent pieces? ★

25. Prove that every natural number $n$ can be written as the sum of distinct powers of two. This proves that every number has a binary representation.

26. Prove that every natural number $n$ can be written *uniquely* as the sum of distinct powers of two. This proves that every number has a *unique* binary representation.

27. Prove that every natural number $n$ can be written as the sum of distinct Fibonacci numbers.

28. Prove that every natural number $n$ can be written as the sum of distinct Leonardo numbers.

29. Prove that every natural number greater than or equal to six can be written as $3x + 4y$, where $x$ and $y$ are natural numbers.

30. When discussing the well-ordering principle, we proved the well-ordering principle from the principle of strong induction. In other words, if we only knew that strong induction was true, we could

prove the well-ordering principle. Now, prove the opposite result, that if the well-ordering principle is true, then the principle of strong induction must be true. This shows that the two principles are equivalent to one another, in that any proof done by strong induction can be rewritten to use the well-ordering principle and vice-versa. ★

31. Since well-ordering and strong induction are equivalent to one another, it should be possible to re-write the proofs from the tail end of the chapter using strong induction. Try this out by proving that the square root of two is irrational using strong induction rather than well-ordering.

32. The ***binary search algorithm*** is a fast algorithm for searching a sorted list of values for a specific element. It is defined as follows:

```
bool binarySearch(list L, int value) {
    if L is empty, return false.


    if length(L) is even, let mid = length(L) / 2
    else, let mid = (length(L) - 1) / 2.


    if L[mid] == value:
        return true.
    if L[mid] < value:
        return binarySearch(L[mid + 1:], value).
    if L[mid] > value:
        return binarySearch(L[:mid - 1], value).
}
```

Prove, by strong induction on the length of $L$, that if $L$ is stored in sorted order, then **`binarySearch`**($L$, *value*) returns whether *value* is contained in $L$.

33. Prove that if $n \leq F_k$, then evaluating **`euclideanGCD`**($m$, $n$) makes at most $k + 1$ function calls. Because Binet's formula shows that the Fibonacci numbers grow exponentially quickly, this shows that computing $gcd(m, n)$ with the Euclidean algorithm requires at most logarithmically many steps. ★

34. This question asks you to prove the division algorithm.

    1. Prove that for any natural numbers $m$ and $n$, where $n \neq 0$, that there exists at least one choice of $q$ and $r$ such that $m = nq + r$, where $0 \leq r < n$. As a hint, consider the set of natural numbers $\{ q \in \mathbb{N} \mid m - nq \geq 0 \}$.
    2. Now, prove that there is a *unique* choice of $q$ and $r$ with this property. ★

35. ***Pascal's triangle*** is a mathematical object that arises in a surprising number of contexts. Below are the first few rows of Pascal's triangle, though it continues infinitely:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

…

The first and last elements of each row are always 1. Every internal number is the sum of the two numbers above it in the triangle. Mathematically, we can write this as follows:

$$P(m, 0) = P(0, m) = 1 \text{ for all } m \geq 0.$$

$$P(m, n) = P(m - 1, n) + P(m - 1, n - 1).$$

Interestingly, it's possible to directly compute the values in Pascal's triangle using the following formula:

$$P(m, n) = \frac{m!}{n!(m-n)!}$$

Prove that the above formula is true by induction. There are two variables here, so you'll have to think about how you want to structure your inductive proof.

36. Prove that if $A$ is a finite set of cardinality $n$, that for any $k \in \mathbb{N}$ with $0 \leq k \leq n$, the number of subsets of $A$ of cardinality exactly $k$ is $P(m, n)$, where $P(m, n)$ is defined above.

37. Suppose that you draw $n$ infinite straight lines in a plane such that no two lines are parallel and no two lines intersect at a single point. This will partition the plane into multiple different regions, some of which are bounded, and some of which are unbounded. How many total regions is the plane split into, as a function of $n$?

38. Suppose that you draw $n$ infinite straight lines in a plane such that no two lines are parallel and no two lines intersect at a single point. This will partition the plane into multiple different regions, some of which are bounded, and some of which are unbounded. How many *bounded* regions does the plane contain, as a function of $n$? ★

39. If you've taken a calculus course, you've probably seen the following rule for computing the derivative of $x^n$, for any $n > 0$:

$$\frac{d}{dx} x^n = n x^{n-1}$$

Prove, by induction on $n$, that this is the case. You might find the following two laws of derivatives useful:

$$\frac{d}{dx} x = 1 \qquad\qquad \frac{d}{dx}(f(x)g(x)) = \frac{d\,f(x)}{dx} g(x) + \frac{d\,g(x)}{dx} f(x)$$

# Chapter 4 Graph Theory

Abstraction is one of the key ideas in software engineering. Rather than building multiple different pieces of code to store a list of numbers, a list of names, a list of DNA sequences, etc., we simply build one single piece of code representing "a list of objects," then use that list to store data of all different types.

In this chapter, we will introduce a powerful abstraction that will appear repeatedly throughout our exploration of the mathematical foundations of computing: the *graph*. Graphs make it possible to reason about the relations between objects and how individual connections between objects can give rise to larger structures. For example, studying graphs makes it possible to reason about the overall distances between two cities, given their pairwise distances, or to find an optimal allocation of workers to tasks, given information on each individual worker's preferences.

Intuitively, a graph is a collection of objects (usually referred to as *nodes* or *vertices*) that can be connected to one another. Specifically, any pair of nodes may be connected by an *edge* (sometimes called an *arc*). Here are some examples of graphs:

- Molecules in chemistry can be thought of as graphs – each atom in the molecule is a node, and each bond between two atoms is an edge.
- Social networks like Facebook are graphs – each person is a node, and each friendship between two people is an edge.
- The Internet is a graph. Each computer is a node, and there is an edge between two computers iff the first computer can directly communicate with the second.
- Polyhedra like cubes, pyramids, and dodecahedrons are graphs. Each corner of the polyhedron is a vertex, and there is an edge between vertices if there is an edge of the solid connecting them.
- Highway systems are graphs. Each node represents a place where two highways meet, and there are edges between two connections iff the junctions have a highway between them.
- Your brain is a graph. Each neuron is a node, and there are edges between neurons if they meet at a synapse.

This chapter explores graphs and their properties. We'll start off by going over some basic formalisms necessary to precisely define a graph. Then, we'll explore certain types of graphs that arise frequently in computer science, along with various properties of graphs that are useful from an algorithmic and theoretical perspective. Later on, when we discuss computational complexity, we will explore how certain properties of graphs are known to be efficiently computable, while other properties are conjectured to be hard to compute.

## 4.1 Basic Definitions

### 4.1.1 Ordered and Unordered Pairs

In order to discuss graphs, we need a way of formalizing what a node and edge are. Typically, anything can act as nodes in a graph: people, places, words, numbers, neurons, etc. Since edges run between two nodes, we often represent edges as which pair of nodes they connect. For example, if we have a graph where each node is a person and each edge represents a friendship, we would represent the edge indicating that Alice is friends with Bob with the pair "Alice, Bob."

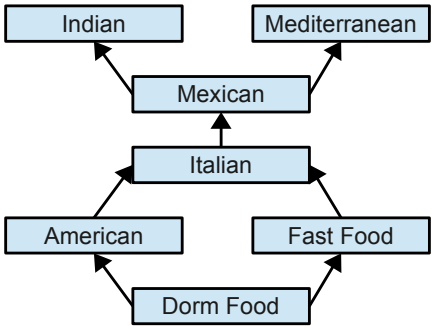
We can formalize the idea of a pair with two definitions below:

> An ***unordered pair*** is a set { *a*, *b* } representing the two objects *a* and *b*.

Intuitively, an unordered pair represents two objects without specifying that one of these objects is the "first" object and that one of these is the "second" object. For example, the friendship between Alice and Bob might be represented as the unordered pair { Alice, Bob }. Since sets are unordered collections of distinct elements, this means that { Alice, Bob } = { Bob, Alice }.

In some cases, we might have that a node in a graph is connected to itself. For example, let's consider a graph where each node represents a computer on a network and each edge between computers represents a pair of computers that can directly communicate with one another. Each computer is capable of communicating with itself, and so each computer would have an edge to itself. For example, HAL9000 would be connected to HAL9000, and GLaDOS would be connected to GLaDOS. We would represent these connections with the unordered pairs { HAL9000, HAL9000 } and { GLaDOS, GLaDOS }. If you'll remember, sets are unordered collections of *distinct* elements, which means that the set { HAL9000, HAL9000 } is the exact same set as { HAL9000 }. However, we will still consider this singleton set to be an unordered pair. Specifically, any set { *a* } can be thought of as an unordered pair containing two copies of *a*.

Unordered pairs are useful if we want to pair up objects together such that neither is "first" or "second." However, in some cases we may want to have two objects where there is a clear "first" and "second." For example, suppose that we have a graph where each node represents a type of food. Some types of food are tastier than others. If one type of food is tastier than another, then we will represent this by drawing an edge from the tastier food to the less tasty food. For example, if I rank food according to my preferences, we would get this graph:




In this graph, it's very important which directions these arrows go. I definitely prefer Italian food to fast food, and not the other way around! If we want to represent this graph, we will need a way to specify edges such that we don't lose this information. For this, let's introduce another definition:

> An ***ordered pair*** is a collection of two objects *a* and *b* in order. We denote the ordered pair consisting first of *a*, then of *b* as (*a*, *b*). Two ordered pairs ($a_0$, $b_0$) and ($a_1$, $b_1$) are equal iff $a_0 = a_1$ and $b_0 = b_1$.

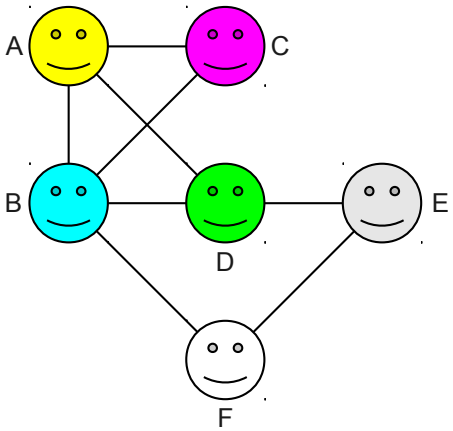
For example, I would represent that I like Italian more than fast food with the ordered pair (Italian, Fast Food). Similarly, I would represent that I like Thai more than American with the ordered pair (Thai, American).*

### 4.1.2 A Formal Definition of Graphs

Now that we have a formal definition of nodes and edges, we can formally define a graph.

> A ***graph*** $G$ is an ordered pair $G = (V, E)$, where $V$ is a set of vertices and $E$ is a set of edges.

This definition of a graph is very flexible, and we can use whatever objects we'd like as the vertices. For example, consider the following graph, which represents friendships in a group of people:



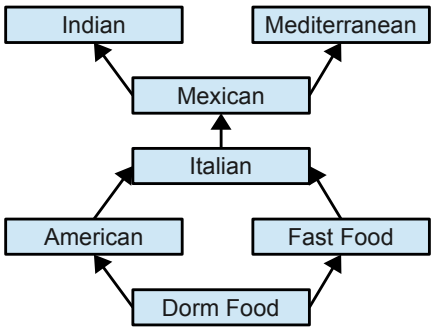

In this case, we could represent the graph as follows:

$$V = \{ A, B, C, D, E, F \}$$

$$E = \{ \{A, B\}, \{A, C\}, \{A, D\}, \{B, C\}, \{B, D\}, \{B, F\}, \{D, E\}, \{E, F\} \}$$

$$G = (V, E)$$

Similarly, if we have the following graph, which represents food preferences:




We would represent the graph as follows:

$$V = \{ \text{Indian, Mediterranean, Mexican, Italian, American, Fast Food, Dorm Food} \}$$

---

* I think I shouldn't write this when I'm hungry. If you'll excuse me, I'm going to get some dinner now.

$$E = \{ \text{(Mexican, Indian), (Mexican, Mediterranean), (Italian, Mexican), (American, Italian), (Fast Food, Italian), (Dorm Food, American), (Dorm Food, Fast Food)} \}$$

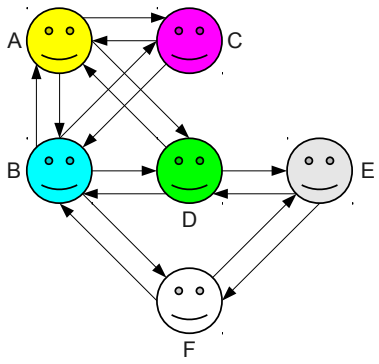
$$G = (V, E)$$

Notice that in the first case, our edges were represented as a set of *unordered* pairs, because friendship is bidirectional – if person A is a friend of person B, then person B is a friend of person A. In the second case, our edges were represented as a set of *ordered* pairs, because preferences have a definite ordering to them. Both of these are perfectly legal graphs, though in this respect they are quite different. In fact, we can think of these graphs as representatives of larger classes of graphs. In some graphs, the edges are directed, meaning that they flow from one node to another, while in other graphs the edges are undirected and link both nodes equally. This distinction is important, which leads to these definitions:

> An ***undirected graph*** is a graph $G = (V, E)$, where $E$ is a set of unordered pairs. A ***directed graph*** (or ***digraph***) is a graph $G = (V, E)$, where $E$ is a set of ordered pairs.

The term "graph" without qualification often refers to both directed and undirected graphs. In the cases where it matters, I will explicitly disambiguate between the two of them.

That said, we can think of undirected graphs as just a special case of directed graphs. For example, we can redraw the above undirected graph representing friendships as the following directed graph:



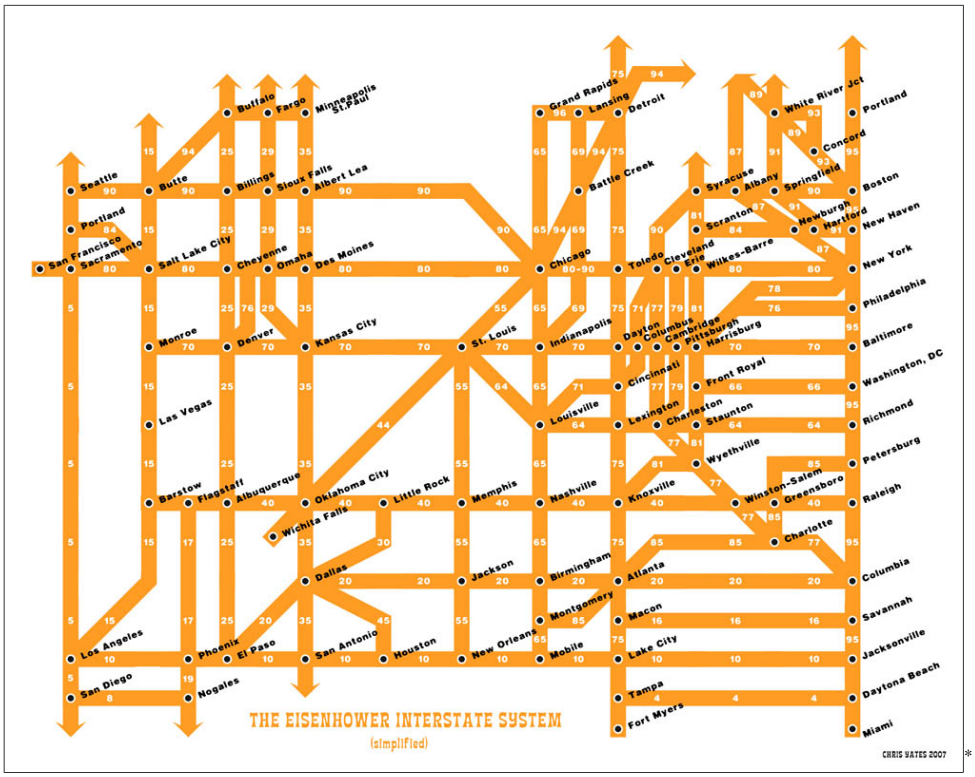

Here, we have edges going both directions between the pairs of nodes. Consequently, in many cases, we can just discuss directed graphs without needing to discuss undirected graphs as a special case.

## 4.1.3 Navigating a Graph

Now that we have a formal definition of a graph, we can start exploring their properties.

We have defined a graph as simply a set of objects that act as nodes and a set of edges linking individual pairs of nodes. However, most of the interesting operations on graphs, and many of the applications best modeled by graphs, focus on the interactions between multiple nodes and multiple edges. In fact, the entire remainder of this chapter explores properties of graphs that are too global to be understood simply by looking at a single node or a single edge.

To start off, let's consider the following graph, which shows the Eisenhower freeway system and how it connects various cities:




Some pairs of cities are directly connected to one another by highways. For example, there is a direct connection from San Francisco to Sacramento, and from Sacramento to Los Angeles. However, other pairs of cities are connected by the highway system, but not directly connected. For example, it's possible to reach Chicago from San Francisco by going from San Francisco to Sacramento, Sacramento to Salt Lake City, Salt Lake City to Cheyenne, Cheyenne to Omaha, Omaha to Des Moines, and finally from Des Moines to Chicago.

This sequence of cities – San Francisco, Sacramento, Salt Lake City, Cheyenne, Omaha, Des Moines – represents a way of getting from one city to another along a series of hops, rather than just a single hop. We can generalize this to a larger setting: given an arbitrary graph $G = (V, E)$, is it possible to get from one node $s$ to another node $t$ by following a series of edges? If the nodes are directly connected, we can just follow the edge between them, but otherwise we might have to take a longer series of edges to arrive at the destination.

We can formalize this idea here:

> A ***path*** in a graph $G = (V, E)$ is a series of nodes $(v_1, v_2, \ldots, v_n)$ where for any $i \in \mathbb{N}$ with $1 \leq i < n$, there is an edge from $v_i$ to $v_{i+1}$.

In other words, a path is a series of nodes where each adjacent pair of nodes has an edge connecting them. It's legal to have a path consisting of any nonzero number of nodes. The above path in the highway graph has six nodes in it. We can consider a short path of just two nodes, such as the path from New York to Philadelphia, or even a trivial path of just one node from any city to itself.

---

* This image taken from http://www.chrisyates.net/reprographics/index.php?page=424.

The above definition of a path only says that any adjacent pair of nodes must have an edge connecting them. This means that, according to our definition, the following is a legal path from San Francisco to Los Angeles:

(SF, LA, SF, LA, SF, LA, SF, LA)

This is a very silly path, since it just keeps going around and around and around. In many cases when we discuss paths in a graph, we want to disallow paths that repeatedly revisit the same locations over and over again. This gives rise to the definition of a simple path:

> A ***simple path*** is a path with no repeated nodes.

Under this definition, the path (SF, LA, SF, LA) is indeed a legal path, but it is not a *simple* path. The initial path we took from San Francisco to Chicago is also simple path.

Notice that this definition just says that a simple path cannot repeat any nodes. This implicitly also means that the path cannot repeat any edges, since if that were the case the path would have to repeat that edge's endpoints.

The notion of a path gives us a way of formalizing a trip from one city to another. Now, let's suppose that we want to take a short vacation from some city (say, Knoxville). This means that we want to leave Knoxville, go visit a few other cities, and then ultimately return back to Knoxville. For example, one possible trip would be to go from Knoxville to Nashville, from there to Birmingham, then to Atlanta, then to Charlotte, Winston-Salem, and then back to Knoxville. We could describe this trip as a path from Knoxville back to itself:

(Knoxville, Nashville, Birmingham, Atlanta, Charlotte, Winston-Salem, Knoxville)

This path is not a simple path, because Knoxville appears twice. However, it is an important type of path, in that it starts and ends at the same location. We call such a path a *cycle*:

> A ***cycle*** is a path that starts and ends at the same node.

As with our definition of a path, notice that this definition of cycle says nothing about what nodes are along the path, as long as we start and end at the same node. Sometimes, we want to talk about cycles that, in a spirit similar to that of simple paths, don't end up repeating themselves unnecessarily. That is, we want to consider cycles that start at a given node, return to that node, and don't end up retracing any steps or repeating any other vertices. For example, we wouldn't want to talk about cycles like this one:

(SF, LA, Phoenix, LA, SF)

which retrace the same edges multiple times, or this one:

(LA, San Diego, Nogales, Phoenix, Flagstaff, Albuquerque, El Paso, Phoenix, LA)

which never ends up retracing the same stretch of highway, but does indeed revisit some intermediate city (in this case, Phoenix) twice. To distinguish cycles like these, which end up retracing the same node or edge twice, from “simpler” cycles that don't retrace any steps, we have this definition:

A ***simple cycle*** is a cycle that does not contain any duplicate nodes (except for the very last node) or duplicate edges.

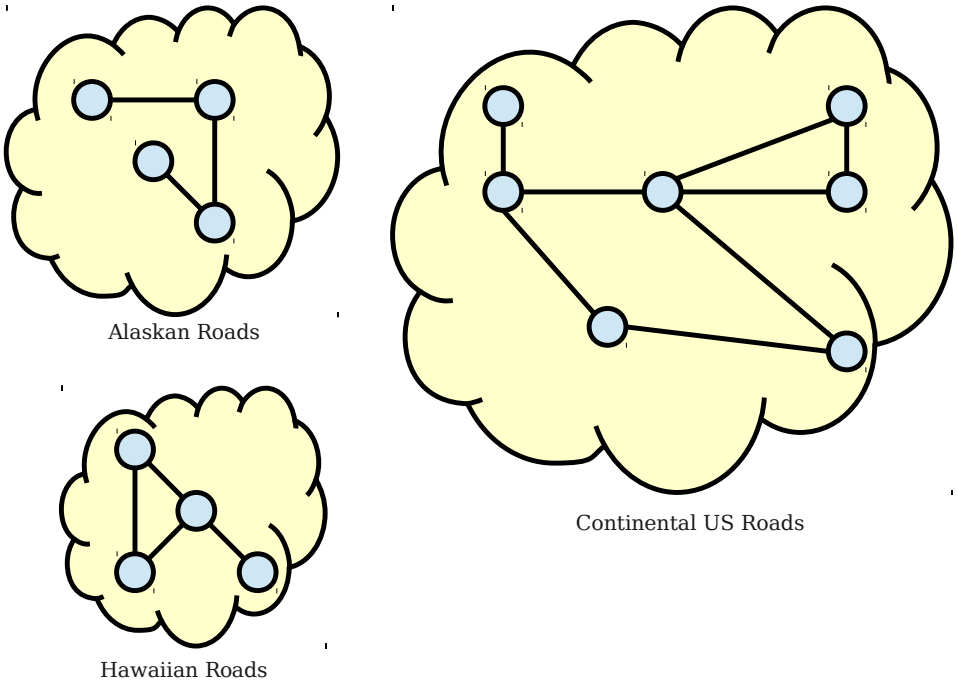
There is a subtle asymmetry between this definition of a simple cycle and our previous definition of a simple path. When describing simple paths, we didn't need to specify that no edges could be repeated, whereas in this definition we have such a restriction. The reason for this is that we want to be able to consider cycles like (SF, LA, SF) to not be simple cycles, since this cycle just follows the same edge forwards and backwards.

## 4.2 Graph Connectivity

Here's a classic riddle – what two US states have no interstate highways? The answer: Hawaii and Alaska, since they don't border any other states![*]

Suppose you were to draw a graph of the road systems in the entire United States. You would end up with a picture containing several smaller road systems that are all independent of one another. The roads within the contiguous United States are likely all connected to one another, but the roads in Hawaii, Alaska, and other US territories (Puerto Rico, Guam, etc.) would be off on their own, unconnected to the roads of the rest of the United States.

Let's consider this from a graph-theoretic perspective. We already saw that we could represent the Eisenhower freeway system as a graph; could we generalize this to the entire road transportation grid for the US? The answer is yes; the graph might look something like this:




Although there are many different “pieces” here, what you are looking at is still one graph. This graph looks different from the other graphs we have seen in this chapter in that it has several smaller pieces, none of

* See http://en.wikipedia.org/wiki/List_of_Interstate_Highways for a discussion of why this isn't quite true.

which touch one another. If we look back at our definition of a graph (a set of nodes and a set of edges), nothing says that we can't have graphs like these. In fact, important graphs often consist of smaller pieces.

This section explores *graph connectivity* and how we can measure how "connected" the nodes in a graph are. Does a graph have just one piece, or does it consist of several smaller pieces? If the graph consists of just one piece, how much "damage" can we do to that graph before we split it into multiple pieces?

### 4.2.1 Connected Components

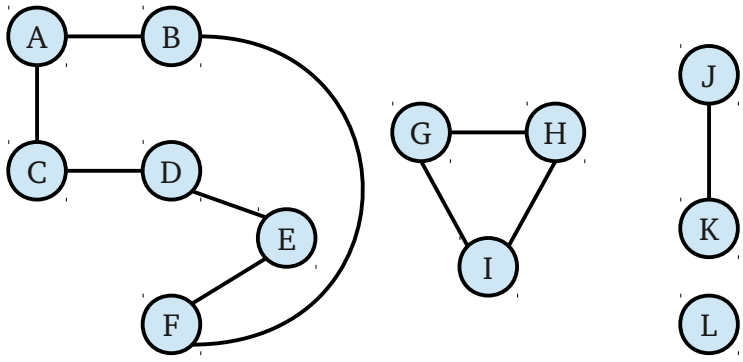
Let's begin this discussion with some basic terms and definitions. First, we need to have a way of talking about what the "pieces" of a graph are. We'll motivate this with a series of definitions.

Look at the above graph of the full US road system. How can we tell, for example, that Honolulu and San Francisco are in different pieces of the graph? One way that we can see this is that there is no way to start at Honolulu, follow a series of edges, and then end up at San Francisco. In other words, there is no path in the graph from Honolulu to San Francisco. This looks like a reasonable way of detecting whether two nodes are in different "pieces" of the graph – if there is no path between them, then they can't belong to the same piece.

This discussion motivates the following definition:

> Let $G$ be an undirected graph. Two nodes $u$ and $v$ are called ***connected*** iff there is a path from $u$ to $v$ in $G$. If $u$ and $v$ are connected, we denote this by writing $u \leftrightarrow v$. If $u$ and $v$ are not connected, we denote this by writing $u \nleftrightarrow v$.

For example, consider the following graph:




In this graph, not all nodes are connected. For example, $A \nleftrightarrow G$. However, many nodes are connected. For example, $A \leftrightarrow B$ and $B \leftrightarrow F$, since there are direct edges between those nodes. Additionally, although there is no edge between them, $A \leftrightarrow E$ because we can follow the path ($A$, $B$, $F$, $E$) to get from $A$ to $E$. We could also have taken the path ($A$, $C$, $D$, $E$) if we had liked. Sitting by its lonesome self is node $L$. We call node $L$ an ***isolated vertex*** or ***isolated node*** because it has no connections to any other nodes in the graph.

Connectivity between nodes has several nice properties. For starters, every node is connected to itself – we can just take the trivial path of starting at a node, not following any edges, and then ending up at that node. This means that $v \leftrightarrow v$ for any node $v$. Similarly, if we know that $u$ is connected to $v$, then it's also true that $v$ is connected to $u$, since we can just follow the edges in the path from $u$ to $v$ in reverse order. In other words, if $u \leftrightarrow v$, then $v \leftrightarrow u$. Finally, if $u$ is connected to $v$ and $v$ is connected to $w$, then we know that $u$ is

connected to $w$, since we can start at $u$, follow the path to $v$, then follow the path to $w$. Consequently, if $u \leftrightarrow v$ and $v \leftrightarrow w$, then $u \leftrightarrow w$.

These three results are summarized here, along with more formal proofs:

***Theorem:*** Let $G = (V, E)$ be an undirected graph. Then:

(1) If $v \in V$, then $v \leftrightarrow v$.
(2) If $u, v \in V$ and $u \leftrightarrow v$, then $v \leftrightarrow u$.
(3) If $u, v, w \in V$, then if $u \leftrightarrow v$ and $v \leftrightarrow w$, then $u \leftrightarrow w$.

***Proof:*** We prove each part independently.

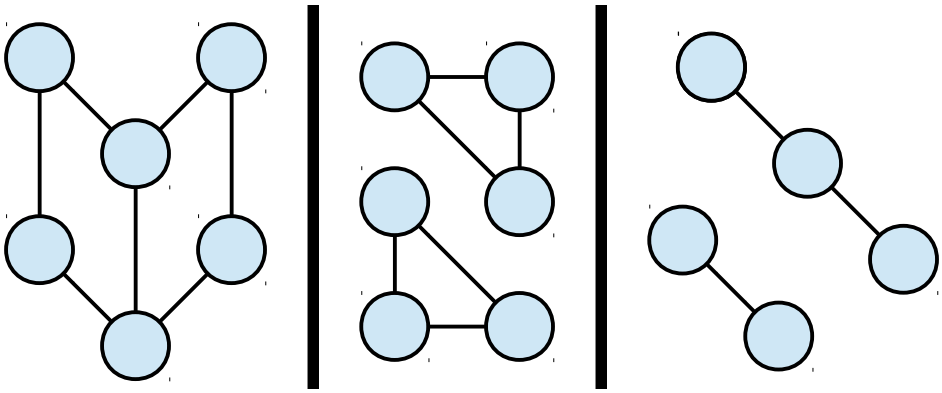
To prove (1), note that for any $v \in V$, the trivial path $(v)$ is a path from $v$ to itself. Thus $v \leftrightarrow v$.

To prove (2), consider any $u, v \in V$ where $u \leftrightarrow v$. Then there must be some path $(u, x_1, x_2, \ldots, x_n, v)$. Since $G$ is an undirected graph, this means $v, x_n, \ldots, x_1, u$ is a path from $v$ to $u$. Thus $v \leftrightarrow u$.

To prove (3), consider any $u, v, w \in V$ where $u \leftrightarrow v$ and $v \leftrightarrow w$. Then there must be paths $u, x_1, x_2, \ldots, x_n, v$ and $v, y_1, y_2, \ldots, y_m, w$. Consequently, $(u, x_1, \ldots, x_n, v, y_1, \ldots, y_m, w)$ is a path from $u$ to $w$. Thus $u \leftrightarrow w$. ■

The definition of connectivity we have just defined works pairwise and can be used to talk about whether two nodes in a graph are in the same "piece" of the graph. However, we still do not have a way of talking about what those "pieces" actually are. Using our previous definition, let's see if we can find a suitable definition for a "piece" of a graph.

First, let's see if we can find a way to talk about graphs that have just one piece. We motivated connectivity by remarking that two nodes must be in different pieces from one another if there is no path from one to the other. If the entire graph is one "piece," then this can't happen. In other words, we should have that every node in the graph is connected to every other node. We can try this definition out on a few graphs. For example, of the following three graphs:



The graph on the left seems like it's one big piece, and you can verify that every node is indeed connected to each other node. The other two graphs are in multiple pieces, and you can inspect them to check that they each contain at least one pair of nodes that are not connected to one another. Consequently, this notion of

the graph being one "piece" (namely, that every node is connected to every other node) seems like it's a very reasonable idea. In fact, this is precisely how we define graphs of one piece, which we call *connected graphs*:

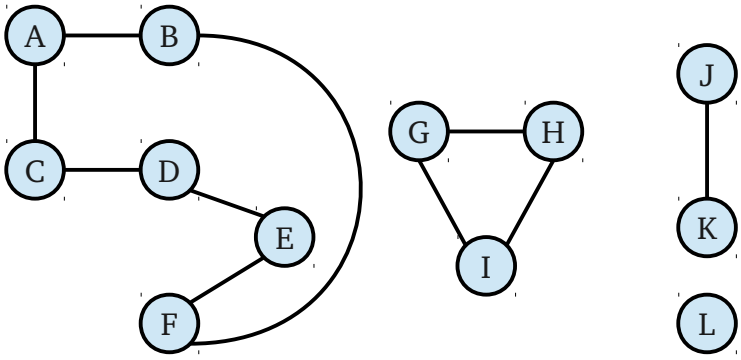
> An undirected graph $G = (V, E)$ is called ***connected*** if for any $u, v \in V$, we have $u \leftrightarrow v$. If $G$ is an undirected graph that is not connected, we say that $G$ is ***disconnected***.

An important note here is that this definition only applies to *undirected* graphs. We'll go over the corresponding definition for directed graphs, which is a bit more involved, later in the chapter.

We now have a way of talking about whether a graph is in just one piece. If the graph is disconnected and consists of several smaller pieces, we still do not have a way of talking about what those pieces are. Let's see if we can come up with one.

Our definition of connectivity gives us a way to talk about whether two nodes are in the same piece as one another; specifically, if $u \leftrightarrow v$, then $u$ and $v$ are in the same piece, and if $u \nleftrightarrow v$, then $u$ and $v$ belong to different pieces. This is a good first step – it gives us a way of checking whether two nodes are in the same piece – but it's not necessarily clear how we can scale this up into a full definition of what a "piece" of a graph is.

Intuitively, a piece of a graph is a bunch of nodes that are all connected to one another. Perhaps this can serve as a good definition for a piece of a graph? Initially, this might seem like exactly what we want, but unfortunately it doesn't quite give us what we're looking for. For example, consider this graph from earlier:




If you'll note, the nodes $A$, $B$, $C$, and $D$ are all connected to one another, but they don't really form a complete piece of the graph. After all, $E$ and $F$ are also in the same piece as them. So it seems like we have to update our definition. Specifically, what if we strengthen our definition of a piece of a graph so that we require it to contain as many nodes as it possibly can? In other words, a piece of a graph is a set of nodes, where each pair of nodes is connected to one another, that is as large as possible. Here, "as large as possible" means that every node in the graph connected to some node within this set of nodes must also be contained within that set. That is, if we have a set of nodes that are all connected together and find some other node connected to one of the nodes in the set, then we go and add that node into the set as well. Now, the pieces of the graph start to look more like real pieces.

Let's formalize our definition of the pieces of a graph, which are more commonly called *connected components*:

Let $G = (V, E)$ be an undirected graph. A ***connected component*** of $G$ is a nonempty set of nodes $C$ (that is, $C \subseteq V$), such that

(1) For any $u, v \in C$, we have $u \leftrightarrow v$.
(2) For any $u \in C$ and $v \in V - C$, we have $u \nleftrightarrow v$.

Let's take a minute to see what this definition says. By definition, a connected component is a set of nodes where all the nodes within that set are connected (rule 1). To ensure that this set is as large as possible, rule 2 says that if you pick any node in $C$ (call it $u$) and any node not in $C$ (call it $v$), then $u$ and $v$ are not connected. The notation $v \in V - C$ just means that $v$ is a node in the graph (it's in $V$) but not contained within $C$.

Our definition of a connected component now gives us a way to formally talk about the pieces of a graph. Remember that we arrived at this definition through several small steps – we first defined connectivity between nodes, then considered sets of nodes that were all connected, then finally arrived at this definition by considering the largest sets of connected nodes that we could. Much of mathematics proceeds this way – we identify some property that makes intuitive sense, then try to pin it down as precisely as possible in a way that is completely formal.

To ensure that our definition actually makes sense, we should take a minute to confirm that the object we have just defined corresponds to what we think it should. Specifically, we need to resolve some very important questions:

- How do we know that connected components even exist? That is, can we even be sure that any graph can be broken up into connected components? Or might there be some "exotic" graphs that don't have any connected components?
- How do we know that there is just one way of breaking any graph down into connected components? We motivated the discussion of connected components by talking about the "pieces" of a graph; are we sure that there's only one way of breaking the graph down into "pieces" this way?

To address these questions, we will prove two important theorems. First, we will prove that connected components cannot overlap. This means that when we split a graph apart into connected components, we really are splitting it into separate pieces. Second, we will show that it is always possible to break a graph apart into connected components. This means that our definition must not be totally arbitrary, since *any* graph can be split into different connected components.

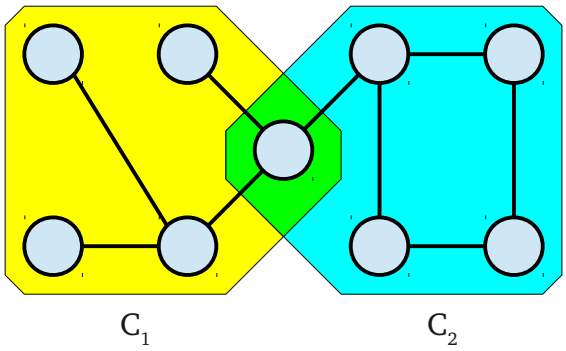
First, let us prove this theorem:

***Theorem:*** Let $G$ be an undirected graph and let $C_1$ and $C_2$ be connected components of $G$. If $C_1 \neq C_2$, then $C_1 \cap C_2 = \emptyset$.

Before we prove this theorem, let's take a minute to see what this means. The above theorem says that if we find any two connected components of a graph that aren't the same connected component, then the two connected components have no nodes in common (that is, their intersection has no nodes in it). As a quick aside, two sets that have no elements in common are said to be *disjoint*:

Two sets $A$ and $B$ are said to be ***disjoint*** iff $A \cap B = \varnothing$.

So how do we prove the above theorem? Intuitively, this makes sense, since a connected component is a "piece" of a graph, and we can't have two overlapping pieces. But in order to prove the result, we will need to argue from the definition of connected components. How might we do this? Well, let's think about what would happen if we had two overlapping connected components. In that case, there would have to be at least one node in common to the connected components (let's call it $v$), meaning that we'd have a picture like this one:




Here, we can make an observation. Since the two connected components $C_1$ and $C_2$ are not equal to one another, there must be some node in one of the connected components that isn't in the other connected component. Let's call this node $u$, and let's suppose that it's in $C_1$ but not $C_2$ (we can do this without loss of generality; if it's really in $C_2$ but not $C_1$, we can just relabel the connected components the other way). Since both $u$ and $v$ are contained within $C_1$, by the definition of a connected component, we can guarantee that $u \leftrightarrow v$. But there's another part of the definition of connected components, namely, that if we can find a node within some connected component $C$ and a node outside of that connected component, we can guarantee that those two nodes are not connected to one another. In particular, this would mean that since $v$ is contained in $C_2$ and $u$ is not contained with $C_2$, it must be the case that $u \nleftrightarrow v$. But that can't be true, since we know that $u \leftrightarrow v$!

We can formalize this reasoning below:

***Theorem:*** Let $G$ be an undirected graph and let $C_1$ and $C_2$ be connected components of $G$. If $C_1 \neq C_2$, then $C_1 \cap C_2 = \emptyset$.

***Proof:*** By contradiction. Suppose that $C_1$ and $C_2$ are connected components of some undirected graph $G$, that $C_1 \neq C_2$, but that $C_1 \cap C_2 \neq \emptyset$. Since $C_1 \cap C_2 \neq \emptyset$, there must be some node $v$ such that $v \in C_1$ and $v \in C_2$. Furthermore, since $C_1 \neq C_2$, there must be some node $u$ that either $u \in C_1$ or $u \in C_2$, but not both. Without loss of generality, assume that $u \in C_1$ and $u \notin C_2$.

By the definition of a connected component, since $u \in C_1$ and $v \in C_1$, we know $u \leftrightarrow v$. Similarly, by the definition of a connected component, since $v \in C_2$ and $u \notin C_2$, we know that $u \not\leftrightarrow v$, contradicting our previous assertion. We have reached a contradiction, so our assumption must have been wrong. Thus $C_1 \cap C_2 = \emptyset$, as required. ■

We have just shown that if we split a graph into connected components, we can guarantee that those connected components don't overlap one another. In other words, we really are splitting the graph into disjoint pieces.

Another way of thinking about this theorem is the following: no node in a graph can belong to more than one connected component. You can see this as follows – suppose that a node could actually be in two different connected components. But by the previous theorem, we know that the intersection of those connected components must be empty, contradicting the fact that our chosen node belongs to both connected components.

We have now established that each node in a graph belongs to *at most* one connected component, but we haven't yet confirmed that every node in the graph must belong to *at least* one connected component. In other words, we can't necessarily guarantee that there isn't some exotic graph in which some node doesn't actually belong to a connected component. Let's round out our treatment of connected components by showing that, indeed, each node belongs to at least one connected component.

So how exactly would we do this? Notice that this proof is an *existence proof* – for any node, we want to show that there is some connected component containing that node. If you'll recall, all we need to do for this proof is to show, for any node $v$, how to find some mathematical object that is a connected component containing $v$. How might we find such a set?

Intuitively, we can think about a connected component containing $v$ as the piece of the graph that $v$ resides in. This "piece" corresponds to all of the nodes that are connected to $v$. This means that we could approach this proof as follows: consider the set of all the nodes connected to $v$. If we can prove that this set is a connected component and that it contains $v$, then we're done: since our choice of $v$ here is totally arbitrary, we can conclude that any node $v$ belongs to some connected component.

Let's now start thinking about what this proof might look like. First, let's formalize our intuition about "the set of all nodes connected to $v$." Using set-builder notation, we can define this set as

$$C = \{ u \in V \mid u \leftrightarrow v \}$$

Now, we need to prove three facts about this set:

1. $v \in C$. Otherwise, this set can't possibly be a connected component containing $v$.
2. For any nodes $u_1, u_2 \in C$, $u_1 \leftrightarrow u_2$. In other words, any pair of nodes in this set are connected to one another.
3. For any nodes $u_1 \in C$ and $u_2 \in V - C$, $u_1 \nleftrightarrow u_2$. In other words, no nodes outside of $C$ are connected to any of the nodes in $C$.

We're making progress – we now have three smaller results that, if proven, collectively prove that $C$ is a connected component containing $v$. Let's now explore how we might prove them.

To prove fact (1), we need to show that $v \in C$. This seems silly – after all, why wouldn't $v$ be in its own connected component? – but remember that we've arbitrarily defined $C$. Given an arbitrary definition, we can't assume much about it. Fortunately, this step is not too hard. Note that by definition of $C$, we have $v \in V$ iff $v \leftrightarrow v$. Earlier in this section, we proved that for any node $v$ in any graph $G$, that $v \leftrightarrow v$. Consequently, we immediately get that $v \in C$.

To prove fact (2), we need to show that any pair of nodes in $C$ are connected by proving that for any $u_1, u_2 \in C$, that $u_1 \leftrightarrow u_2$. Intuitively, this should be true. We've defined $C$ as the set of all nodes connected to $v$. Consequently, any two nodes in $C$ must be connected to one another, since we can find a path between them by starting at the first node, taking a path to $v$, then taking a path from $v$ to the second node. Formally, we can use two results from earlier in the chapter: first, that if $x \leftrightarrow y$ and $y \leftrightarrow z$, then $x \leftrightarrow z$; second, that if $x \leftrightarrow y$, then $y \leftrightarrow x$. We can then say that since $u_1 \leftrightarrow v$ and $u_2 \leftrightarrow v$, we know that $v \leftrightarrow u_2$, and consequently, $u_1 \leftrightarrow u_2$.

The last step is to show that if we take any node $u_1$ in $C$ and any node $u_2$ not in $C$, that $u_1$ is not connected to $u_2$. We can reason about this by contradiction – suppose that there is a $u_1$ in $C$ and a $u_2$ not in $C$, but that $u_1 \leftrightarrow u_2$. Since $v \leftrightarrow u_1$, this would mean that $v \leftrightarrow u_2$, meaning that $u_2$ should be contained in $C$, contradicting the fact that it is not.

Let's formalize all of these ideas in a proof of the following theorem:

***Theorem:*** Let $G = (V, E)$ be an undirected graph. Then for any $v \in V$, there is a connected component $C$ such that $v \in C$.

***Proof:*** Let $G = (V, E)$ be any undirected graph and let $v \in V$ be an arbitrary node in the graph. Consider the set $C = \{ u \in V \mid u \leftrightarrow v \}$. We will prove $C$ is a connected component containing $v$.

First, we prove that $v \in C$. To see this, note that by construction, $v \in C$ iff $v \leftrightarrow v$. As proven earlier, $v \leftrightarrow v$ is always true. Consequently, $v \in C$.

Next, we prove that $C$ is a connected component. This proof proceeds in two steps: first, we prove that for any $u_1, u_2 \in C$, that $u_1 \leftrightarrow u_2$; second, we prove that for any $u_1 \in C$ and $u_2 \in V - C$, that $u_1 \not\leftrightarrow u_2$.

To prove that for any $u_1, u_2 \in C$, that $u_1 \leftrightarrow u_2$, consider any $u_1, u_2 \in C$. By construction, this means that $u_1 \leftrightarrow v$ and $u_2 \leftrightarrow v$. As proven earlier, since $u_2 \leftrightarrow v$, we know that $v \leftrightarrow u_2$. Also as proven earlier, since $u_1 \leftrightarrow v$ and $v \leftrightarrow u_2$, this means that $u_1 \leftrightarrow u_2$.

Finally, to prove that for any $u_1 \in C$ and $u_2 \in V - C$, that $u_1 \not\leftrightarrow u_2$, consider any $u_1 \in C$ and $u_2 \in V - C$. Assume for the sake of contradiction that $u_1 \leftrightarrow u_2$. Since $u_1 \in C$, we know that $u_1 \leftrightarrow v$. Since $u_1 \leftrightarrow u_2$, we know $u_2 \leftrightarrow u_1$. Therefore, since $u_2 \leftrightarrow u_1$ and $u_1 \leftrightarrow v$, we have that $u_2 \leftrightarrow v$. Thus by definition of $C$, this means that $u_2 \in C$, contradicting the fact that $u_2 \in V - C$. We have reached a contradiction, so our assumption must have been wrong. Thus $u_1 \not\leftrightarrow u_2$.

Thus $C$ is a connected component containing $v$. Since our choice of $v$ and $G$ were arbitrary, any node in any graph belongs to at least one connected component. ■

This theorem, combined with the one before, gives us this result:

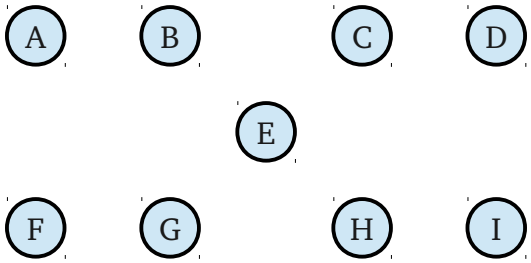
***Theorem:*** Every node in an undirected graph belongs to exactly one connected component.

In other words, it's always meaningful to talk about "the connected component containing $v$" or "$v$'s connected component." We have successfully found a way to break all of the nodes in a graph apart into pieces based on how they are connected to one another!

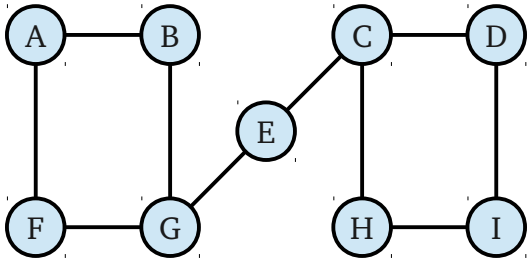
### 4.2.2 2-Edge-Connected Graphs ★

Our discussion of connected components in the previous section gives us a precise way of pinning down the pieces of a graph. However, this definition just says what nodes are connected to which other nodes. It says nothing about how tightly those nodes are connected, or how fragile that connectivity might be.

As an example, suppose that you have been tasked with laying out highways that connect various cities together. These cities are shown below:

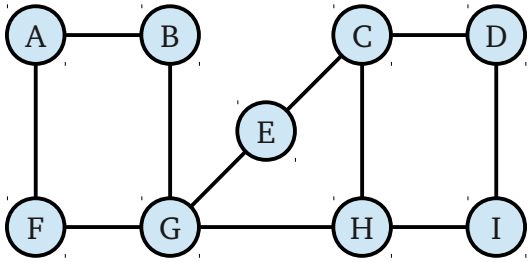



If all that you care about is ensuring that the graph is connected, you might decide to lay out the highways like this:




But is this really a *good* way of laying out highways? After all, highways often need to close down, either for repairs or some (natural or unnatural) disaster that blocks them. Given that this is the case, the above highway system actually is not very good. For example, if the highway between *E* and *C* were to unexpectedly close, it would break connectivity and isolate all of the cities in the east and west regions of the country. That would be disastrous.

On the other hand, you could consider laying out the highway like this:




Given this way of laying out the roads, every city is reachable from every other city (as before). However, this graph is more resilient to damage. Specifically, if any one highway were to shut down, it's still possible to travel from any city to any other city. However, it's possible that *two* highways might shut down and break all east-west transit routes. Specifically, if the highway between *E* and *C* closed at the same time that the highway between *G* and *H* closed down, there would be no way to get across the country.

How do we characterize graph connectivity in circumstances where edges between nodes might suddenly break down? What do well-connected and poorly-connected graphs look like? This section and the one that follows it answers that question.

In the course of this section, we will consider what happens to graph connectivity when you start removing edges from the graph. We could alternatively discuss what happens when you start removing *nodes* from the graph, which gives rise to many interesting and important concepts. However, for the sake of brevity, we'll defer that discussion to the exercises at the end of the chapter.

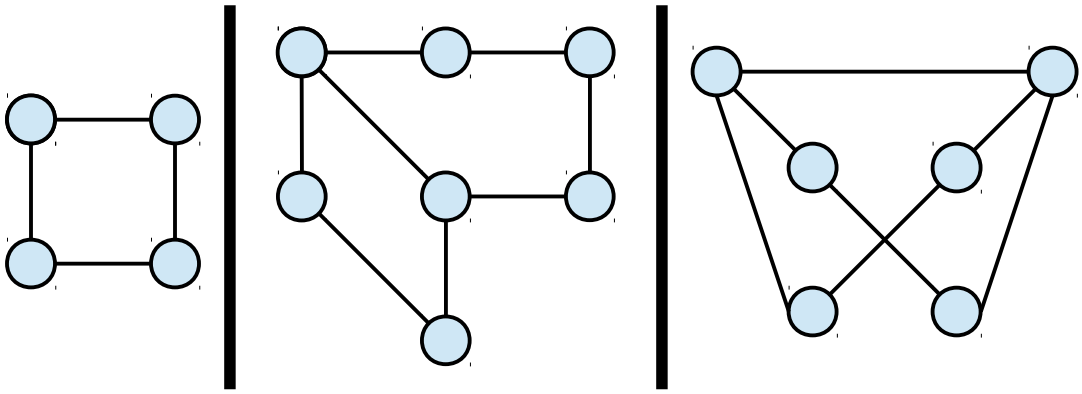
Most of our discussion in this section will focus on particular types of graphs whose connectivity is resilient even as edges start being removed. To begin with, we will give a definition for a class of graphs that are connected, but have some known degree of redundancy.

> An undirected graph $G$ is called ***k-edge-connected*** iff $G$ is connected, and there is no set of $k - 1$ edges that can be removed from $G$ that disconnects it.

Intuitively, you can think of $k$-edge-connected graphs as graphs with the following property. Suppose that you have a collection of computers (nodes) linked together in a network by cables (edges). Initially, each computer can communicate with each other computer either directly (because they are linked together), or indirectly (by routing messages through other computers). A saboteur finds these computers, then chooses and cuts her choice of $k - 1$ of these cables, breaking the links. If the graph of these computers is $k$-edge-connected, then you don't need to worry about the cut cables. Every computer will still be able to reach every other computer. That said, if the saboteur were to cut one more cable, it's possible (though not guaranteed) that the network might end up disconnected.

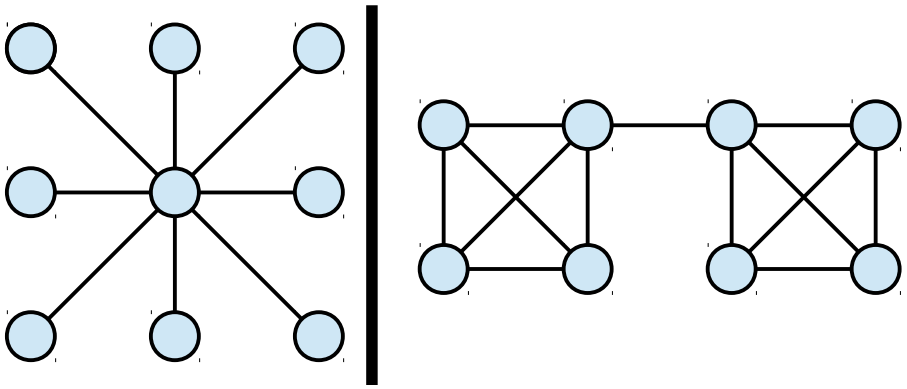
So what do the $k$-edge-connected graphs look like? It turns out that for $k = 1$ and $k = 2$, the $k$-edge-connected graphs have beautiful theoretical properties and are surprisingly common in computer science (especially when $k = 1$). While $k$-edge-connected graphs are quite interesting for $k \geq 3$, the rest of this section will only focus on the case where $k = 1$ or $k = 2$.

We will first focus on the 2-edge-connected graphs. By definition, these are the graphs that are connected, but cannot be disconnected by removing any single edge. For example, the following graphs are 2-edge-connected:



As a quick exercise, I'd suggest confirming for yourself that you can't disconnect any of these graphs by cutting just one edge.

For contrast, these graphs are not 2-edge-connected:



Both of these non-2-edge-connected graphs contains at least one edge that, if removed, will disconnect the graph. These edges have a name:

> A ***bridge*** in a connected, undirected graph *G* is an edge in *G* that, if removed, disconnects *G*.

Given this definition, we can restate the definition of a 2-edge-connected graph as follows:

> ***Theorem:*** An undirected graph *G* is 2-edge-connected iff it is connected and has no bridges.

This theorem is effectively a restatement of the definition of a 2-edge-connected graph. I'm going to omit the proof and (in a move that will shock most mathematicians) *not* leave it as an exercise to the reader.

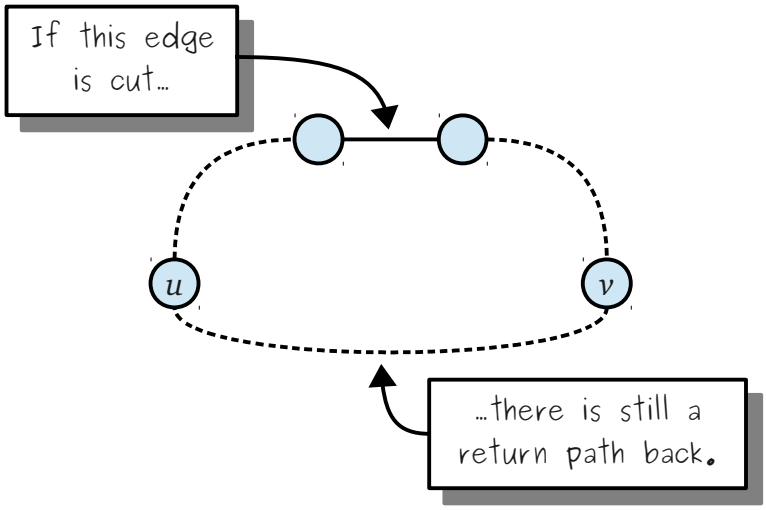
The above definition gives us a way to check if a graph is 2-edge-connected: we can check whether the graph is connected, and from there check each edge to see if it forms a bridge. While this procedure will correctly check whether a graph is 2-edge-connected, it doesn't shed much light on exactly why a 2-edge-connected graph is 2-edge-connected. What is it about the underlying structure of 2-edge-connected graphs that allows any edge to be deleted without breaking connectivity? To answer this question, we will embark on a bold and epic mathematical quest to understand the nature of 2-edge-connectivity and 2-edge-connected graphs.

To start off our journey, it probably makes sense to look for simple graphs that are 2-edge-connected. If we can understand why these simple graphs are 2-edge-connected, we can try to scale up our reasoning to larger and more complicated graphs. If you're ever confronted with a mathematical or algorithmic problem, it often helps to adopt this approach – start with some simple cases, and see if you can figure out a more general pattern.

Let's take a minute to think about why a graph would be 2-edge-connected. For this to be true, the graph has to be connected, and it must stay connected even if we delete any single edge. Our definition of connectivity says that a graph is connected iff there is a path in the graph between any pair of nodes. Consequently, we can restate what it means to be 2-edge-connected as follows:

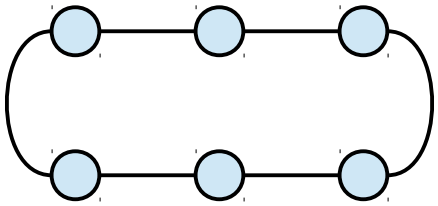
1. There is a path between any pair of nodes in the graph, and
2. After deleting any single edge from the graph, there is still a path between any pair of nodes in the graph.

One way of thinking about these two statements is as follows. Pick any pair of nodes $u$ and $v$ in a graph $G$. If the graph is 2-edge-connected, then there must be a path between $u$ and $v$. Now, delete any edge you'd like from the graph. Since $u$ and $v$ are still connected, one of two things must have happened. First, it's possible that the edge we deleted wasn't on the path we picked, which means that $u$ and $v$ still have to be connected. Second, it's possible that the edge we deleted was indeed on the path from $u$ to $v$. But since the graph is still connected, we know that there must be some alternate route we can take that goes from $u$ to $v$. You can see this schematically below:




Now, for a key insight. Suppose that we start off at $u$ and take our initial path to $v$. We then take the secondary path from $v$ back to $u$. This gives us a cycle that goes from $u$ back to itself. It's not necessarily a *simple* cycle, though, since we might end up retracing some of the same nodes and edges. But nonetheless, this quick thought experiment shows that there is some kind of connection between 2-edge-connected graphs and cycles. Let's play around with this idea and see what we can find out exactly what it is.

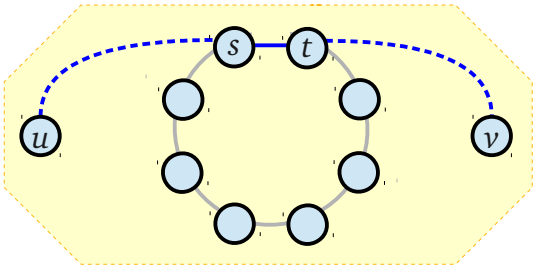
One observation that we might have is the following. Consider the following very simple graph, which consists purely of a single cycle:



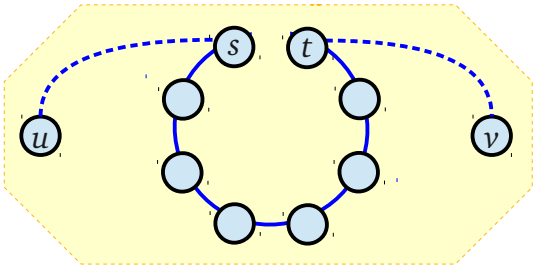
We can check that this graph is 2-edge-connected; deleting any single edge won't disconnect the graph. One way we can reason about this is the following: pick any pair of nodes in this graph and choose one of the two paths between those nodes. Then, delete any edge out of the graph. Now, if you didn't delete an edge that's on the path you picked, then clearly the pair of nodes you've picked are still connected to one another. However, if you did pick an edge on the path, you can find a different path from the first node to the second by going around the cycle in a different direction.

It turns out that this reasoning is actually slightly more general. Let's suppose that you have an arbitrary graph $G$ that contains a simple cycle somewhere within it. Now, suppose that $G$ is connected, so there's a path between any pair of nodes in $G$. What happens if we delete any one of the edges on the indicated cycle? Can that possible disconnect the graph?

The answer is no, for the following reason. Pick any start and end nodes $u$ and $v$, and choose any path between them. As with before, if the edge that we deleted off of the cycle isn't on the path from $u$ to $v$, then $u$ and $v$ are still connected by the previous path. On the other hand, if the edge we deleted from the cycle was on the path from $u$ to $v$, then we can always find another path as follows. Let's suppose that the edge we deleted ran between nodes $s$ and $t$. This means that the original path looked like this:




In this case, we can adjust the broken path from $u$ to $v$ so that it no longer tries to follow the nonexistent edge from $s$ to $t$. We do so as follows. First, follow the path from $u$ up until it first hits $s$. Then, since $s$ and $t$ are on a cycle, route the path around the cycle from $s$ until you arrive at $t$. Finally, proceed along the path from $t$ to $v$ as before. This is demonstrated here:




Note that this process isn't guaranteed to produce a *simple* path from $u$ to $v$. This path might take us partway around the cycle, then backtrack one we hit the broken edge. But nevertheless, this does give a valid path from $u$ to $v$.

The above intuition gives a justification for the following theorem:

> ***Theorem:*** Let $G = (V, E)$ be any graph containing a simple cycle $C$. Let $u, v \in V$ be nodes in $G$. If $u \leftrightarrow v$, then after deleting any single edge in $C$ from graph $G$, it is still the case that $u \leftrightarrow v$.

Intuitively, if two nodes in a graph are connected and we delete any edge from a cycle, then it must be the case that those two nodes are still connected. We can formally prove this result by using a line of reasoning similar to the reasoning from before.

***Proof:*** Consider any graph $G = (V, E)$ with a simple cycle $C = (x_1, x_2, \ldots, x_n, x_1)$. Consider any $u, v \in V$ such that $u \leftrightarrow v$. This means that there must be some simple path $(u, y_1, y_2, \ldots, y_m, v)$ from $u$ to $v$.*

Now, suppose that we remove the edge $\{x_i, x_{i+1}\}$ from $G$.† We need to show that $u \leftrightarrow v$ in this modified graph. We consider two cases. First, it might be the case that the edge $\{x_i, x_{i+1}\}$ does not appear on the path $(u, y_1, \ldots, y_m, v)$. In that case, the path $(u, y_1, \ldots, y_m, v)$ is a valid path from $u$ to $v$ in the new graph, so $u \leftrightarrow v$ still holds.

Second, it might be the case that the edge $\{x_i, x_{i+1}\}$ appears somewhere in our original path $(u, y_1, \ldots, y_m, v)$. Since the graph is undirected, the edge might appear as $\{x_i, x_{i+1}\}$ or as $\{x_{i+1}, x_i\}$ when it occurs in the path. Assume without loss of generality that it appears as $\{x_i, x_{i+1}\}$ (otherwise, we can just reverse the ordering of the nodes in the original cycle so as to relabel the edges). This means that we can split the original path into three smaller paths – a path from $u$ to $x_i$, then the edge $\{x_i, x_{i+1}\}$, and finally a path from $x_{i+1}$ to $v$. Thus $u \leftrightarrow x_i$ and $x_{i+1} \leftrightarrow v$. Now, since the edge $\{x_i, x_{i+1}\}$ lies on the cycle $C$, after deleting the edge from the cycle, there is still a path from $x_i$ to $x_{i+1}$. Specifically, we can follow the edges of the cycle in reverse from $x_i$ until we reach $x_{i+1}$. In other words, in this new graph, we must have that $x_i \leftrightarrow x_{i+1}$.

Since in this new graph $u \leftrightarrow x_i$, $x_i \leftrightarrow x_{i+1}$, and $x_{i+1} \leftrightarrow v$, we thus have that $u \leftrightarrow v$ in the new graph, as required. ■

We will return to this theorem many times in the course of the chapter. It is a fundamental property of cycles and connectivity that deleting an edge from a cycle cannot disconnect a graph that was previously connected.

An important corollary of this theorem is the following result about connected graphs:

***Corollary:*** If $G$ is a connected, undirected graph containing a simple cycle $C$, then $G$ is still connected if any single edge is removed from $C$.

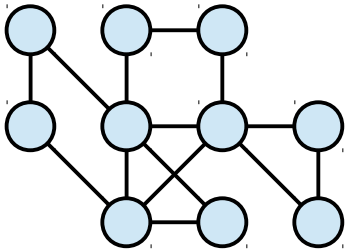
***Proof:*** Consider any undirected, connected graph $G = (V, E)$ containing a simple cycle $C$. Consider any edge $e \in C$. We will prove that if $e$ is removed from $G$, then $G$ is still connected. To see this, consider any pair of nodes $u, v \in V$. Since $G$ is connected, $u \leftrightarrow v$. By the previous theorem, if we remove $e$ from the graph, since $e$ lies on the simple cycle $C$, we know that $u \leftrightarrow v$ still holds in the graph formed by deleting $e$ from $G$. Thus $u$ and $v$ are still connected. Since our choice of $u$ and $v$ were arbitrary, this shows that any pair of nodes in $G$ are still connected after $e$ is removed, as required. ■

Let's take a minute to think about how the above theorem connects to 2-edge-connected graphs. We have just shown that if we have a pair of nodes that are connected in a graph and then delete *any* edge that lies on a cycle, those two nodes must still be connected in the new graph. Now, what would happen if *every* edge in the graph was part of some simple cycle? This is not to say that there is one giant simple cycle that contains

* We have not formally proven that $u \leftrightarrow v$ iff there is a simple path from $u$ to $v$. It's a good exercise to try to prove this result. As a hint, try using the well-ordering principle by considering the shortest path from $u$ to $v$ and proving that it must be a simple path.

† It might be the case that this edge is the last edge on the cycle, which goes from $x_n$ to $x_1$. In proofs such as this one, it is typical to allow for a slight abuse of notation by letting $x_{i+1}$ mean "the next node on the cycle," which might not necessarily have the next index.

every edge; rather, it just says that every edge lies on some simple cycle. For example, in the graph below, every edge lies on at least one simple cycle, and some edges lie on more:



Now, consider any graph like this and pick a pair of connected nodes $u$ and $v$. By the above theorem, we can't disconnect $u$ and $v$ by removing any single edge that lies on a cycle. But if we know that every edge in the graph lies on some simple cycle, then we can guarantee that there is no possible way that deleting any single edge from the graph could disconnect $u$ and $v$. No matter which edge we pick, we can always find a path from $u$ to $v$.

This motivates the following theorem:

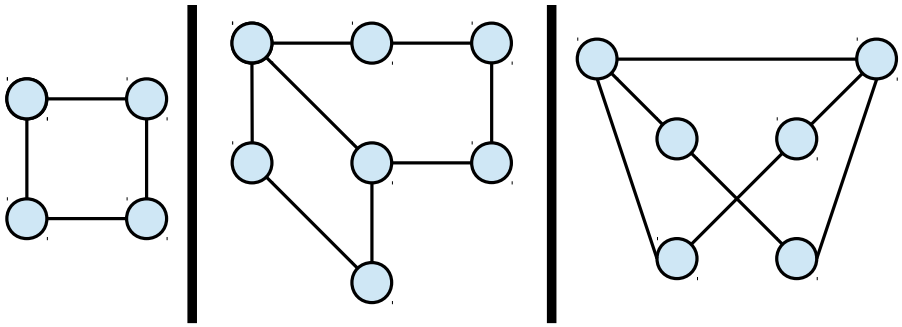
***Theorem:*** Let $G$ be an undirected graph. If $G$ is connected and every edge of $G$ belongs to at least one simple cycle, then $G$ is 2-edge-connected.

***Proof:*** Consider any undirected, connected graph $G = (V, E)$ where each edge of $G$ lies on at least one simple cycle. To show that $G$ is 2-edge-connected, we need to show that $G$ is connected and that if any single edge from $G$ is removed, then $G$ is still connected. By our initial assumption, we already know that $G$ is connected, so all we need to do is show that removing a single edge from $G$ does not disconnect $G$.

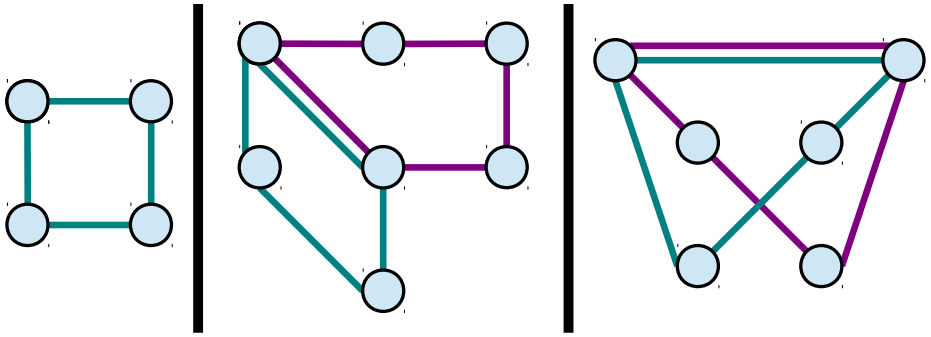
Consider any edge $e$. By assumption, $e$ lies on some simple cycle $C$. Consequently, by our previous corollary, since $G$ is connected and $e$ lies on a simple cycle, $G$ is still connected after removing $e$ from $G$. Since our choice of $e$ was arbitrary, this means that removing any single edge from $G$ does not disconnect it, from which it follows that $G$ is 2-edge-connected. ■

We are now on our way to getting a better understanding of the 2-edge-connected graphs. We have just shown that if we have a connected graph where each edge lies on a simple cycle, then it must be the case that the graph is 2-edge-connected.

To motivate the next observation, let's consider a few 2-edge-connected graphs, like these ones here:



We just proved that any connected graph where each edge lies on a simple cycle must be 2-edge-connected. If you'll notice, all of the above graphs are 2-edge-connected, but also have the additional property that each edge lies on a simple cycle. You can see this here:



Is this a coincidence? Or must it always be the case that if a graph is 2-edge-connected, every edge lies on a simple cycle?

It turns out that this is not a coincidence. In fact, we can prove the following theorem:

> ***Theorem:*** If $G$ is 2-edge-connected, then every edge in $G$ lies on a simple cycle.

The reason for this result is actually quite simple. Consider any 2-edge-connected graph, and any edge within that graph. Suppose that we delete this edge from the graph. Since $G$ is 2-edge-connected, we can't have disconnected $G$ by deleting this single edge. This means that, in particular, the endpoints of this edge must still be connected, meaning that there must be some simple path between them. It becomes clear that our initial edge must be on a cycle; namely, the cycle formed by following the simple path from one endpoint to the other, then following the initial edge.

We can formalize this here:

> ***Theorem:*** If $G$ is 2-edge-connected, then every edge in $G$ lies on a simple cycle.

> ***Proof:*** Consider any 2-edge-connected graph $G = (V, E)$ and any edge $\{u, v\} \in E$. Remove $\{u, v\}$ from $G$. Since $G$ is 2-edge-connected, $G$ must still be connected after removing this edge. Thus in this new graph $u \leftrightarrow v$. Consequently, there must be a simple path from $u$ to $v$ in this new graph. Since the updated graph does not contain the edge $\{u, v\}$, this simple path cannot contain the edge $\{u, v\}$. Moreover, since the path is a simple path, it must not contain $u$ or $v$ as interior nodes. Thus in the original graph, the cycle formed by following the simple path from $u$ to $v$, then crossing the edge from $v$ to $u$ is a simple cycle. Since our choice of edge was arbitrary, this shows that any edge in $G$ must be on a simple cycle. ■

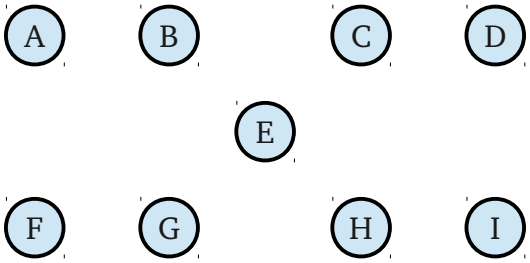
This proof and proof before it give us an exact characterization of the 2-edge-connected graphs: they are precisely the connected graphs in which each edge lies on a cycle. One way of interpreting this result is as follows. Suppose that you have a set of islands connected by bridges and want to see if the islands are still connected even if a bridge fails. You can check this by first confirming that all islands are connected, and then sending an investigator to each bridge. If the investigator can find a way of getting from one side of the

bridge to the other without actually crossing that bridge, you can confirm that the bridges can survive a single disconnection.

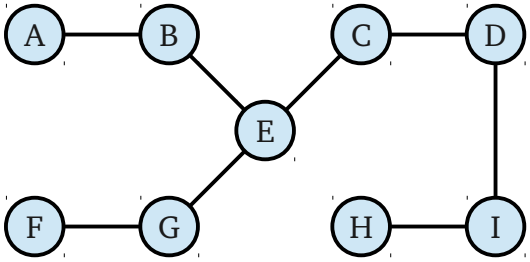
### 4.2.3 Trees ★

Our discussion of 2-edge-connected graphs focused on graphs that were connected with some measure of redundancy. It is always possible to remove an edge from a 2-edge-connected graph without disconnecting that graph. In this section, we turn to the opposite extreme by focusing on the most fragile graphs possible – graphs that are connected, but which have absolutely no redundancy at all.

At first, it might not seem obvious why we would want to study the most fragile possible graphs. However, these graphs have many applications in computer science and operations research. For example, suppose that you have a collection of cities and you want to construct a transportation network to connect all of them. If you have very little money, you would probably want to construct the cheapest network that you can. Such a network probably wouldn't have any redundancy in it, since if there were any redundancies you could always save money by leaving out the redundant roads. Taken to the extreme, you would find that the best road layouts to consider from an economic perspective are road layouts with absolutely no redundancy. For example, given these cities:




You might consider connecting them as follows:

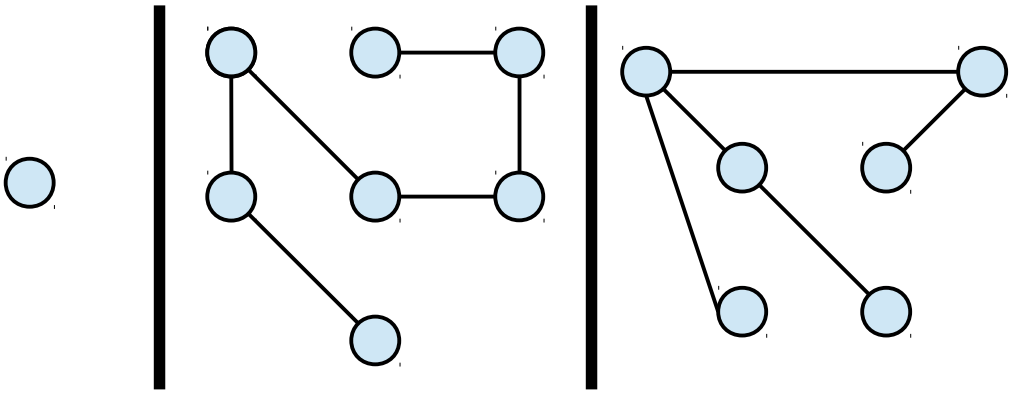



This layout has no margin for error. If a single road closes, you are guaranteed that some number of cities will be disconnected from the grid.

We'll begin this section with a definition:

> An undirected graph *G* is called ***minimally connected*** iff *G* is connected, but the removal of any edge from *G* leaves *G* disconnected.

For example, all of the following graphs are minimally connected, since they are connected but disconnect when any edge is removed:



Notice that the graph of just a single node is considered to be minimally connected because it is indeed connected (every node can reach every other node) and it is also true that removing any edge from the graph disconnects it. This second claim is true vacuously: there are no edges in the graph, so the claim "if any edge is the graph is removed, it disconnects the graph" is automatically true. We can thus think of a single-node graph as a degenerate case of a minimally-connected graph.

In the previous section, we discussed the 2-edge-connected graphs and saw how our initial definition of 2-edge-connectivity (namely, that no single edge disconnection can disconnect the graph) proved equivalent to a different definition (namely, that every edge in the graph lay on a simple cycle). It turns out that the minimally connected graphs have many other characterizations, and there are several other equivalent properties that we can use to describe the minimally-connected graphs. Let's explore a few of these properties.

To begin with, look at each of the above minimally-connected graphs. If you'll notice, not a single one of them contains a cycle. This is not a coincidence. Recall that in the previous section, we proved a theorem that says that if a connected graph contains a cycle, that graph is still connected after we delete that cycle. As a result, it's not possible for a graph to be minimally connected but to contain a cycle, since if there's a cycle in the graph we can cut any edge out of it that we like without disconnecting the graph.

First, let's introduce a quick definition:

A graph is called ***acyclic*** iff it contains no simple cycles.

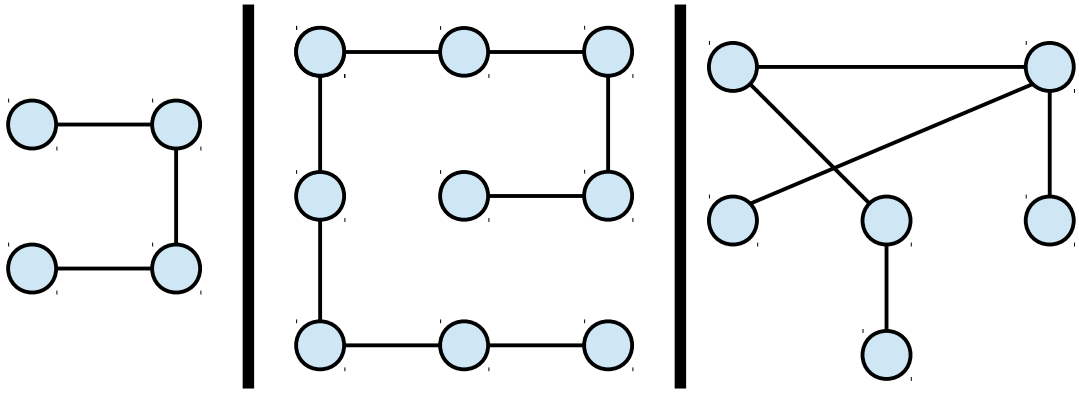
Given this definition and the above observations, we have the following theorem:

***Theorem:*** If an undirected graph $G$ is minimally-connected, then it is connected and acyclic.

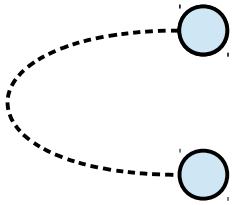
***Proof:*** By contradiction; assume that $G$ is minimally-connected, but that it is not connected or that it is not acyclic. It cannot be the case that $G$ is not connected, since by definition any minimally-connected graph must be connected. So we must have that $G$ is not acyclic, meaning that it contains a simple cycle; call it $C$. By our previous corollary, since $G$ is connected and $C$ is a simple cycle, we can delete any edge $e \in C$ from $G$ without disconnecting $G$. This contradicts the fact that $G$ is minimally-connected. We have reached a contradiction, so our assumption must have been wrong. Thus if $G$ is minimally-connected, then it must be connected and acyclic. ■

We have just proven that a minimally-connected graph must be connected and contain no simple cycles. Is it necessarily true that the converse holds, namely, that any connected graph with no simple cycles is minimally-connected? The answer is yes, and if we wanted to, we could prove this right now. However, we will defer this proof until slightly later on in this section, for reasons that shall become clearer in a few pages. Instead, let's focus on connected graphs that have no cycles. What other properties do these graphs have?

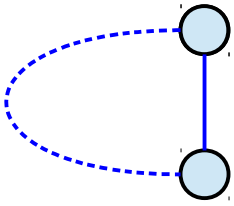
Our definition of minimally-connected graphs concerned what happens when edges are *deleted* from the graph. What happens if we try *adding* edges into the graph? Let's take a look at this and see what happens. Consider any of the following graphs, which are all connected and acyclic:



As a quick exercise, pick any pair of nodes that you'd like, then add an edge connecting them. If you'll notice, no matter which graph you choose or how you pick the nodes, the addition of this new edge creates a simple cycle in the graph. This is no coincidence and is a necessary consequence of the fact that the graph is connected and acyclic. To see why this is, let's think about it schematically. Pick any pair of nodes in a connected, acyclic graph that don't already have an edge between them. Since the graph is connected, there must be a simple path between these nodes:



Since the graph is connected, we know that there must be some simple path connecting these two nodes, as shown here:



Consequently, if we then add in an edge that directly connects the two nodes, we are guaranteed to form a cycle – simply follow an existing simple path from the first node to the second node, then traverse the edge from the second node back to the first node.

This property of connected, acyclic graphs shows that this graphs are, in a sense, as large as they can get while still being acyclic. Adding any missing edge into the graph is guaranteed to give us a cycle. This motivates the following definition:

An undirected graph $G$ is called ***maximally acyclic*** iff it is acyclic, but the addition of any edge introduces a simple cycle.

Given the above line of reasoning, we can prove the following theorem:

***Theorem:*** If an undirected graph $G$ is connected and acyclic, then it is maximally acyclic.

***Proof:*** Consider any undirected, connected, acyclic graph $G = (V, E)$. Now, consider any pair of nodes $\{u, v\}$ such that $\{u, v\} \notin E$. We will prove that adding the edge $\{u, v\}$ introduces a simple cycle. To see this, note that since $G$ is connected, there must be a simple path $(u, x_1, x_2, \ldots, x_n, v)$ from $u$ to $v$ in $G$. Since this path is a simple path, none of the nodes $x_1, x_2, \ldots, x_n$ can be equal to either $u$ or $v$. Now, consider the graph formed by adding $\{u, v\}$ to $G$. We can then complete the previous simple path into a simple cycle by following this new edge from $v$ to $u$, giving the simple cycle $(u, x_1, x_2, \ldots, x_n, v, u)$. Since our choice of edge was arbitrary, this proves that adding any edge to $G$ introduces a simple cycle. Since $G$ is acyclic, this proves that it is maximally acyclic. ■

We now have established the following chain of reasoning:

$$G \text{ is minimally connected} \rightarrow G \text{ is connected and acyclic} \rightarrow G \text{ is maximally acyclic}$$

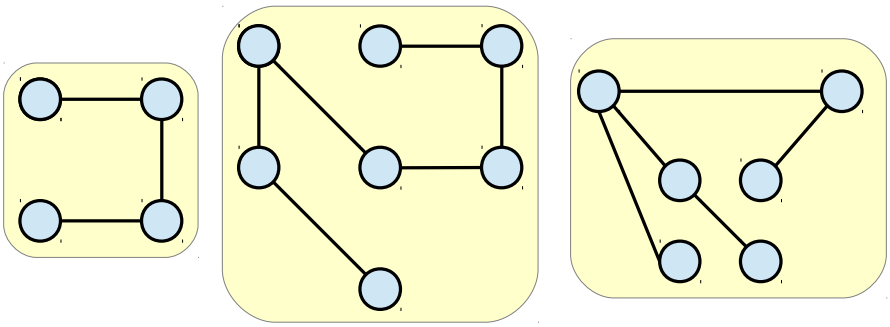
So far, all of the implications have gone in one direction. It's not clear whether it's necessarily true that if $G$ is connected and acyclic, then $G$ must be minimally connected. Similarly, it's not clear whether any graph that is maximally acyclic must also be minimally connected. It turns out, however, that all of the above definitions are completely equivalent to one another.

If we wanted to prove this, one option would be to prove the converse of each of the statements that we have proven so far. However, there is another option that is substantially simpler. What happens if we prove that any maximally acyclic graph must also be minimally connected? In that case, we have a cycle of implication – each of the above properties implies the next. This means, in particular, that we can prove that each of the implications also works in reverse. For example, suppose that we want to prove that $G$ is connected and acyclic iff $G$ is maximally acyclic. We have already proven one direction (namely, that if $G$ is connected and acyclic, then it is maximally acyclic). We can then show the other direction (namely, that if $G$ is maximally acyclic, then it is connected and acyclic) as follows: since $G$ is maximally acyclic, it is minimally connected. Since it's minimally connected, it's therefore acyclic and connected, as required.

More generally, a very powerful proof technique for proving that many properties are all equivalent to one another is to show that each of the properties implies some other property in a way that links all of the properties together in a ring.

Now that we have our work cut out for us, we need to do one more proof – namely, that if $G$ is maximally acyclic, then it must be minimally connected. This proof requires two steps: first, we need to prove that if $G$ is maximally acyclic, then it has to be connected; otherwise, it's not possible for $G$ to be minimally connected! Next, we need to show that if $G$ is maximally acyclic, then it has to be *minimally* connected.

Given that outline, let's get started. First, we need to show that if $G$ is maximally acyclic, then it has to be connected. Why would this be the case? Well, suppose that we have a graph that is maximally acyclic but not connected. In that case, it must consist of several different connected components, as shown here:



Let's think about what would happen if we were to introduce an edge between two nodes contained in different connected components. Since our graph is (allegedly) maximally acyclic, this has to introduce a simple cycle somewhere in the graph. Because the graph was initially acyclic, this means that any simple cycle that we introduced would have to use this new edge somewhere. But this is problematic. Remember that we added an edge that bridged two different connected components. This means that, aside from the edge we just added, there cannot be any edges between nodes in the first connected component and nodes in the second connected component.

Given this observation, we can start to see that something bad is going to happen if we try to route a simple cycle across this new edge. Pick any node on the cycle that starts in the first connected component. After we cross the new edge into the second connected component, we somehow have to get back into the first connected component in order to complete the cycle. Unfortunately, we know that there is only one edge that crosses the connected components – the edge we've added. This means that the cycle has to retrace that edge, meaning that it's no longer a simple cycle.

Given this intuition, we can formalize this into as proof as follows:

***Lemma:*** If $G$ is maximally acyclic, then $G$ is connected.

***Proof:*** By contradiction. Suppose that $G = (V, E)$ is a maximally acyclic graph that is not connected. Since $G$ is not connected, it must consist of several connected components. Choose any two of these connected components and call them $CC_1$ and $CC_2$.

Now, consider any nodes $u \in CC_1$ and $v \in CC_2$. Since $u$ and $v$ are in separate connected components, note that $u \nleftrightarrow v$ and the edge $\{u, v\} \notin E$. So consider what happens when we add the edge $\{u, v\}$ to the graph. Since $G$ is maximally acyclic, this must introduce a simple cycle; call it $C$. Since $G$ is acyclic, this new cycle must use the edge $\{u, v\}$. Additionally, note that since $\{u, v\}$ is an edge in the new graph, we have that $u \leftrightarrow v$ in this new graph.

By our earlier theorem, since in this new graph $u \leftrightarrow v$ and $C$ is a simple cycle, if we delete any single edge from $C$, it will still be the case that $u \leftrightarrow v$ still holds. In particular, this means that if we delete $\{u, v\}$ from the new graph (which yields the original graph $G$), we should have that $u \leftrightarrow v$. But this is impossible, since we know that $u \nleftrightarrow v$ in the original graph.

We have reached a contradiction, so our assumption must have been wrong. Thus if $G$ is maximally acyclic, it must be connected. ■

It's really impressive how much mileage we're getting out of that theorem about cutting edges from cycles!

Given this lemma, we just need to prove one last result to show that any maximally acyclic graph is minimally connected. Specifically, we need to show that if a graph is maximally acyclic, then cutting any edge will disconnect the graph. It turns out that to prove this, we actually don't need to use the fact that the graph is *maximally* acyclic; just being acyclic is sufficient. The proof is given below:

***Theorem:*** If $G$ is maximally acyclic, then it is minimally connected.

***Proof:*** Let $G = (V, E)$ be any maximally acyclic graph. By the previous lemma, $G$ is connected. We need to show that if we remove any edge $e \in E$ from $G$, then $G$ becomes disconnected. To do this, we proceed by contradiction. Suppose that there is an edge $\{u, v\} \in E$ such that if $\{u, v\}$ is removed from $G$, $G$ remains connected. In that case, we must have that after removing $\{u, v\}$ from $G$, there is a simple path between $u$ and $v$. This means that in the original graph $G$, there is a simple cycle – namely, take the simple path from $u$ to $v$, then follow the edge $\{u, v\}$ from $v$ back to $u$. But this is impossible, since $G$ is maximally acyclic and thus acyclic. We have reached a contradiction, so our assumption must have been incorrect. Thus $G$ is minimally connected. ■

Combining these three theorems together, we have the following overall result:

***Theorem:*** Let $G$ be an undirected graph. The following are all equivalent:

1. $G$ is minimally connected.
2. $G$ is connected and acyclic.
3. $G$ is maximally acyclic.

The fact that graphs with any one of these properties also have the other two properties suggests that graphs with these properties are special. In fact, these graphs are incredibly important in computer science, and are so important that we give them their own name: *trees*.

A ***tree*** is an undirected graph that is minimally connected. Equivalently, a tree is an undirected graph that is connected and acyclic, or an undirected graph that is maximally acyclic.

Trees have many uses in computer science. Trees underpin many important data structures, such as the binary search tree or trie, and can be used to model recursive function calls in programs. The motivating problem from this section – how to lay out roads in a way that guarantees connectivity at the lowest cost – can be modeled as searching for the lowest-cost tree connecting all cities. Trees also play a role in certain types of algorithmic analyses, and many problems involving trees are known to be computationally simple, while the same problems on general graphs are often much harder to compute. We'll address this last bit in later chapters.

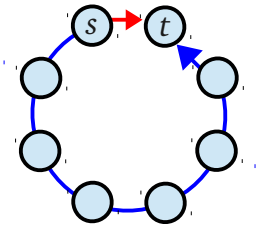
The three properties we proved about trees – minimal connectivity, connectivity/acyclicity, and maximal acyclicity – give us a good intuition about many properties of trees. However, there is one property of trees that is in many cases even more useful than these three. Take a look at any of the trees we've seen so far and pick any pair of points. How many simple paths can you find between those points? No matter what

tree you pick, or which pair of nodes you find, the answer is always one. This is an important theorem about trees:

> ***Theorem:*** Let $G = (V, E)$ be an undirected graph. Then any pair of nodes $u, v \in V$ have exactly one simple path between them iff $G$ is a tree.

In order to prove this result, we will have to prove both directions of implication. First, we will need to show that if there is exactly one simple path between any two nodes in $G$, then $G$ is a tree. Second, we will need to show that if $G$ is a tree, there is exactly one simple path between any two nodes in $G$.

For the first part, let's assume that we have a graph $G = (V, E)$ such that every pair of nodes in $G$ are connected by a unique simple path. To show that $G$ is a tree, we can prove either that $G$ is connected and acyclic, or that $G$ is minimally connected, or that $G$ is maximally acyclic. For simplicity, we'll prove that $G$ is connected and acyclic. We can see this as follows. First, we know that $G$ must be connected, because there's a path between every pair of nodes in $G$. Second, we can see that $G$ must be acyclic for the following reason. Suppose that $G$ contained a simple cycle. Then any two nodes on that cycle would have to have two different simple paths between them, formed by going in opposite directions around the cycle. Schematically:




This contradicts the fact that there is exactly one simple path between any pair of nodes.

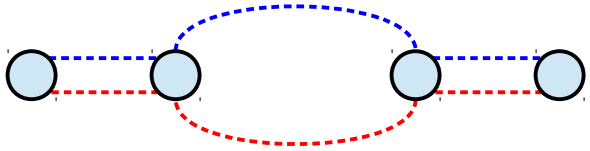
We can formalize this proof as follows:

> ***Lemma:*** Let $G = (V, E)$ be an undirected graph. If for each pair of nodes $u, v \in V$, there is a unique simple path between $u$ and $v$, then $G$ is a tree.

> ***Proof:*** Let $G = (V, E)$ be an undirected graph where each pair of nodes $u, v \in V$ are connected by a unique simple path. We will prove that $G$ is connected and acyclic.
>
> To see that $G$ is connected, note that for any nodes $u, v \in V$, there is a simple path between $u$ and $v$. Thus $u \leftrightarrow v$.
>
> To see that $G$ is acyclic, assume for the sake of contradiction that $G$ is not acyclic. Thus there exists a simple cycle $C = (x_1, x_2, \ldots, x_n, x_1)$ in $G$. We can then split this simple cycle into two simple paths between $x_1$ to $x_2$: $(x_1, x_2)$, and $(x_2, x_3, \ldots, x_n, x_1)$. But this contradicts the fact that there is exactly one simple path between $x_1$ and $x_2$. We have reached a contradiction, so our assumption must have been wrong. Thus $G$ is acyclic.
>
> Since $G$ is connected and acyclic, $G$ is a tree. ■

For the next part of the proof, we need to show that if a graph $G$ is a tree, then there is exactly one simple path between any pair of nodes in $G$. To do this, let's think about what would have to happen if we *didn't* have exactly one simple path between some pair of nodes. If we don't have exactly one simple path, then either we have no paths at all, or there are multiple paths. We can immediately rule out the case where there are zero paths, because in that case it means that the graph is disconnected, which is impossible if $G$ is a tree. So this means that we have to have multiple paths between some pair of nodes.

If we draw out what this looks like, we get the following:



From this picture, it's clear that there has to be a cycle somewhere in the graph, since if we follow any two different simple paths between the nodes, they will diverge at some point, then converge back together later on. It's possible that they diverge at the beginning and then converge at the end, or they may only diverge for a short time in the middle. However, somewhere within this process, they must form a simple cycle. Since trees are acyclic, this immediately tells us that if $G$ is a tree, $G$ can't have more than one simple path between any pair of nodes.

In order to formalize this reasoning in a proof, we somehow need to pin down how to form a simple cycle given two simple paths between the same pair of nodes. One idea that we might get from the above picture is the following – let's look at the first place where the two paths diverge. We know that the paths diverge at some point, since they aren't identically the same path, so clearly there must be some first point where the divergence occurs. At that point, we can see that the two paths will go their own separate ways until ultimately they meet at some node. We can guarantee that they eventually will meet up, since both of the paths end at the destination node, and so we can talk about the earliest point at which they meet. If we then take the segments of the paths spanning from the first spot in which they diverge to the first point after this that they meet, we will have formed a simple cycle.

This is written up as a formal proof here.

***Lemma 2:*** If $G$ is a tree, then there is a unique simple path between any pair of nodes in $G$.

***Proof:*** By contradiction. Suppose that $G = (V, E)$ is a tree, but that there is a pair of nodes $u, v \in V$ such that there are is not exactly one simple path between $u$ and $v$. This means that either there are no paths between $u$ and $v$, or that there are two or more paths between them. It cannot be the case that there are no simple paths between $u$ and $v$, since $G$ is connected, and so there must be at least two distinct simple paths between $u$ and $v$. So choose any two distinct simple paths between $u$ and $v$. Let these simple paths be $(x_1, x_2, \ldots, x_n)$, and $(y_1, y_2, \ldots, y_m)$, where $u = x_1 = y_1$ and $v = x_n = y_m$.

Next, consider the largest value of $i$ such that $x_1 = y_1$, $x_2 = y_2$, …, $x_i = y_i$. Since these paths are not the same path, there must be some value of $i$ for which this is possible, and since the paths have finite length there must be some largest of these values. By our choice of $i$, we know that $x_{i+1} \neq y_{i+1}$, since otherwise our choice of $i$ is not the largest such $i$.

Since both of these paths end at $v$, there must be some earliest node in the subpaths $(x_{i+1}, x_{i+2}, \ldots, x_n)$ and $(y_{i+1}, y_{i+2}, \ldots, y_m)$ that is in common between these two paths. Let this node be labeled $x_s$ and $y_t$.

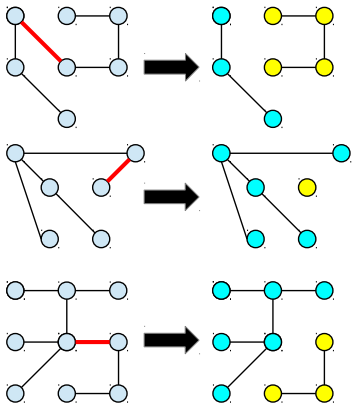
Finally, consider the simple paths $(x_i, x_{i+1}, \ldots, x_s)$ and $(y_i, y_{i+1}, \ldots, y_t)$. By definition, we know that $x_i = y_i$ and that $x_s = y_t$. However, these paths can have no other nodes in common, since we chose $x_s$ and $y_t$ to be the earliest nodes in common to these two paths. Consequently, this means that $(x_i = y_i, x_{i+1}, \ldots, x_s = y_t, y_{t-1}, \ldots, y_i = x_i)$ is a simple cycle. But this is impossible, because $G$ is a tree and is therefore acyclic.

We have reached a contradiction, so our assumption must have been wrong. Thus there must be exactly one simple path between each pair of nodes in $G$. ■

The combination of these two lemmas proves the theorem that $G$ is a tree iff there is exactly one simple path between any pair of nodes. This property, effectively a restatement of the fact that $G$ is acyclic, is very useful in proving additional results about trees.

### 4.2.3.1 Properties of Trees

An important property of trees is that they are minimally connected; removing any edge will disconnect the graph. One question we might ask about trees in this case is the following – suppose that we do cut one of the edges and end up with a disconnected graph. What will the connected components of this graph look like? We can try this out on a few graphs to get a feel for what exactly it is that we're doing:



There's an interesting observation we can make at this point – each of the connected components of the graph, treated in isolation, are trees. This hints at an interesting recursive property of trees: as we start cutting the edges of a tree over and over again, we end up with many smaller graphs, each of which is itself a tree!

To formalize this argument, we will need to prove two points: first, that there will be exactly two connected components; second, that the edges in each of those connected components forms a tree. Let's take this first one as an example – how would we prove that we get exactly two connected components? Initially this might seem obvious, but it's not immediately clear exactly how we can demonstrate this mathematically. So let's play around with some ideas. Intuitively, when we cut the graph into two pieces by deleting an edge, we're going to get two connected components, one on each side of the edge. Let's say that this edge is $\{u, v\}$. We know that $u$ and $v$ will belong to connected components in the resulting graph (since *every* node belongs to some connected component), so if we can show that every node either belongs to the connected component containing $u$ or the connected component containing $v$, then we're done, because that then accounts for all the connected components in the graph.

We now have a clear goal in mind – show that every node is either in $u$'s connected component or in $v$'s connected component – so how would we proceed from here? Well, let's start thinking about what we know about graphs. One important property we proved about trees is that there is exactly one simple path connecting each pair of nodes. Let's think about how we might use that fact here.

Here's a fun experiment to try: take a look at any of the graphs from the previous page and pick one of the two endpoints of the edge that was deleted. Now, pick any node you'd like from the connected component containing it. What is the relation of the unique path from that node to your chosen endpoint and the edge you disconnected? Similarly, pick any node you'd like from the opposite connected component. How does its simple path to your chosen node interact with the edge that was deleted?

Let's suppose that the edge we deleted was $\{u, v\}$. Let's purely focus on the node $u$. If you pick an arbitrary node $x$ in the tree and trace out the simple path from that node to $u$, then one of two things will happen. First, it's possible that the path doesn't contain the edge $\{u, v\}$. In that case, when we delete the edge $\{u, v\}$ from the graph, the node $x$ and the node $u$ will still be connected to one another, since the old path is still valid. Second, it's possible that the path does use the edge $\{u, v\}$. Since we're only considering simple paths, this means that the path had to be following this edge from $v$ to $u$ and not from $u$ to $v$, since if we cross the edge from $u$ to $v$ we can't get back to $v$ without retracing that edge. Thus if we cut the edge $\{u, v\}$ from the graph, we can guarantee that the node $x$ can still reach $v$ (just trace the path up to where you would normally cross $\{u, v\}$), but it can't reach node $u$, since we just cut an edge along the unique path from $x$ to $u$. This gives us an excellent way of defining the connected components of the graph we get when disconnecting an edge $\{u, v\}$. One connected component contains all the nodes whose simple path to $u$ doesn't use $\{u, v\}$, and one connected component contains all the nodes whose simple path to $u$ does use $\{u, v\}$.

Using this reasoning, we can formally prove that cutting any edge in a tree yields two connected components:

***Lemma:*** Let $G = (V, E)$ be a tree with at least one edge. If any edge from $G$ is removed, the resulting graph has exactly two connected components.

***Proof:*** Let $G = (V, E)$ be any tree with at least one edge. Suppose that we remove the edge $\{u, v\}$ from $G$. We claim that the new graph has exactly two connected components. To prove this, we will construct two sets $C_u$ and $C_v$ such that every node in $V$ belongs to exactly one of these sets. We will then prove that they are connected components.

We will define the two sets as follows:

$$C_u = \{ x \in V \mid \text{the unique simple path from } x \text{ to } u \text{ does not cross } \{u, v\} \}$$
$$C_v = \{ x \in V \mid \text{the unique simple path from } x \text{ to } u \text{ crosses } \{u, v\} \}$$

Note that for all $x \in V$, then $x$ belongs to exactly one of these two sets. We now prove that these sets are connected components of the new graph.

First, we prove that any pair of nodes $x, y \in C_u$ satisfy $x \leftrightarrow y$ in the new graph. To see this, consider any $x, y \in C_u$. Note that with the edge $\{u, v\}$ deleted, that $x \leftrightarrow u$ and $y \leftrightarrow u$, since the unique paths from $x$ to $u$ and from $y$ to $u$ in the original graph still exist in the new graph, because we only deleted the edge $\{u, v\}$ from the graph and that edge was not on either of those paths. Consequently, $x \leftrightarrow y$.

Second, we prove that any pair of nodes $x, y \in C_v$ satisfy $x \leftrightarrow y$. To see this, we will prove that for any node $z \in C_v$, that $z \leftrightarrow v$. To see this, consider the unique path from $z$ to $u$. Since this path used the edge $\{u, v\}$, the edge must have been followed from $v$ to $u$. Otherwise, if the edge were followed from $u$ to $v$, the path could not be a simple path, because ultimately that path ends at $u$, which would be a duplicate node. This means that the simple path from $z$ to $u$ must consist of a simple path from $z$ to $v$, followed by the edge $\{u, v\}$. Since the only edge we removed was $\{u, v\}$, this means that the simple path from $z$ to $v$ is still valid in the new graph. Consequently, $z \leftrightarrow v$ in the new graph. Since our choice of $z$ was arbitrary, this means that in the new graph, $x \leftrightarrow v$ and $y \leftrightarrow v$ for any $x, y \in C_v$. Thus $x \leftrightarrow y$ for any $x, y \in C_v$.

Finally, we will show that for any $x \in C_u$ and $y \in V - C_u$, that $x \not\leftrightarrow y$. Note that since every node in $V$ either belongs to $C_u$ or $C_v$, $V - C_u = C_v$. This means that we need to show that for any $x \in C_u$ and $y \in C_v$, that $x \not\leftrightarrow y$. To do this, we proceed by contradiction. Suppose that there exists $x \in C_u$ and $y \in C_v$ such that in the new graph, $x \leftrightarrow y$. As we proved earlier, we know that $x \leftrightarrow u$ and that $y \leftrightarrow z$, so this means that in the new graph, $u \leftrightarrow v$. This means that there is a simple path $P$ from $u$ to $v$ in the graph formed by removing $\{u, v\}$ from $G$. This means that there is a simple cycle in $G$ formed by starting at $u$, following $P$ to $v$, then taking the edge $\{v, u\}$ from $v$ back to $u$. But this is impossible, since $G$ is a tree and therefore acyclic. We have reached a contradiction, so our assumption was wrong and $x \not\leftrightarrow y$.

We have just shown that all nodes in $C_u$ are connected to one another, all nodes in $C_v$ are connected to one another, and that no pair of nodes in $C_u$ or $C_v$ are connected to one another. Thus $C_u$ and $C_v$ are connected components in the new graph. Since all nodes in $V$ belong to one of these two sets, there can be no other connected components in the graph. Thus the new graph has exactly two connected components. ■

This proof is lengthy, but hopefully each individual piece is reasonably straightforward. We first define two sets that we claim will form the connected components, then prove that each is a connected component by (first) showing that all the nodes in each set are connected to one another, and (second) that there are no connections between any nodes in different connected components.

Given this lemma as a starting point, we need only a little more effort to conclude that those two connected components also happen to be trees. Since they're connected components, we know that each piece is connected, and since the original graph was acyclic, we know that each connected component must be acyclic as well. We can formalize this here:

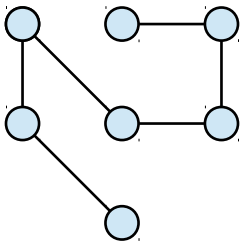
***Theorem:*** Let $G$ be a tree with at least one edge. If any edge is removed from $G$, the resulting graph consists of two connected components that are each trees over their respective nodes.

***Proof:*** Let $G = (V, E)$ be a tree with at least one edge, and let $\{u, v\} \in E$ be an arbitrary edge of $G$. By our lemma, if we remove $\{u, v\}$ from $G$, we are left with two connected components; call them $C_u$ and $C_v$, with $u \in C_u$ and $v \in C_v$. Since $C_u$ and $C_v$ are connected components, they are connected. Consequently, if we can show that $C_u$ and $C_v$ are acyclic, then we can conclude that $C_u$ and $C_v$ are trees.

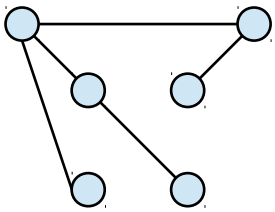
To show that $C_u$ and $C_v$ are acyclic, assume for the sake of contradiction that at least one of them is not; without loss of generality, let it be $C_u$. This means that there is a simple cycle contained purely within $C_u$. But since all of the edges in $C_u$ are also present in $G$, this means that there is a simple cycle in $G$, contradicting the fact that $G$ is a tree. We have reached a contradiction, so our assumption must have been wrong. Thus $C_u$ and $C_v$ must be acyclic, and therefore are trees. ■

The main utility of this theorem is that trees are recursively structured – we can always split apart a large tree into two smaller trees. This makes it possible for us to prove many results about trees inductively by taking a tree, splitting it into two subtrees, and combining the results about the subtrees together.

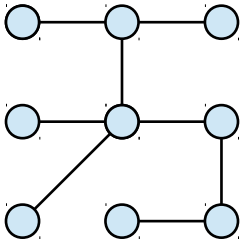
As an example of such a proof, let's investigate the relation between the number of nodes and edges in a tree. If we look at a couple of examples, we'll spot a trend:



7 Nodes, 6 Edges



6 Nodes, 5 Edges



9 Nodes, 8 Edges

1 Node, 0 Edges

It seems that the number of nodes in a tree is always one greater than the number of edges. Let's see if we can try to prove this.

To prove this result, we will use a proof by induction on the number of nodes in the tree. As our base case, we'll consider a tree with exactly one node. Since trees must be acyclic, the only tree with a single node must consist solely of an isolated node with no edges. In this case, we immediately get that the number of nodes (1) is one greater than the number of edges (0).

For the inductive step, let's assume that for some natural number $n$, that all trees with $1 \le n' < n$ nodes have $n' - 1$ edges. Then consider a tree with $n$ nodes. If we pick any edge and cut it, we split the tree into two subtrees, one with $k$ and one with $n - k$ nodes. By our inductive hypothesis, we get that the $k$-node tree has $k - 1$ edges, and the $(n - k)$-node tree has $n - k - 1$ edges. If we add this together, we get $n - 2$ edges. Factoring in the one edge that we deleted, we get that the total number of edges is thus $n - 1$. And we're done!

We can formalize this here:

***Theorem:*** Let $G = (V, E)$ be a tree. Then $|E| = |V| - 1$.

***Proof:*** By strong induction. Let $P(n)$ be "Any tree with $n$ nodes has $n - 1$ edges." We will prove that $P(n)$ is true for all $n \in \mathbb{N}^+$ by strong induction on $n$.

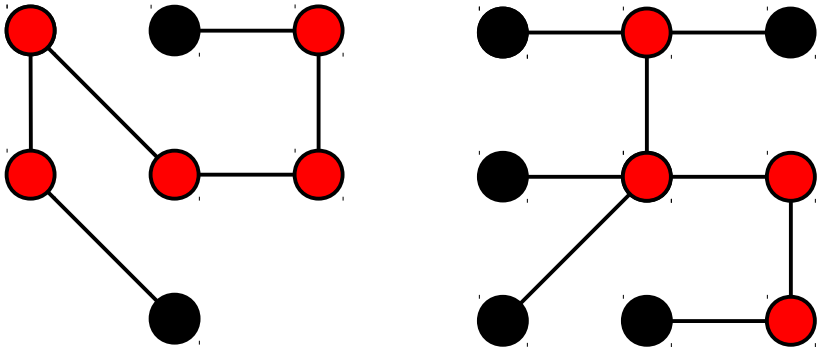
Assume that for some $n \in \mathbb{N}^+$ , that for any $n' \in \mathbb{N}^+$ with $n' < n$, that $P(n')$ holds and any tree with $n'$ nodes has $n' - 1$ edges. We will prove $P(n)$, that any tree with $n$ nodes has $n - 1$ edges.

If $n = 1$, then we need to prove that any tree with 1 node has 0 edges. Any tree with one node cannot have any edges, since any edge would have to be from the one node to itself and would thus cause a cycle. Thus any tree with 1 node has 0 edges, so $P(n)$ holds.

Otherwise, $n > 1$, so $n \geq 2$. So consider any tree $T$ with $n$ nodes; since this tree is connected and has at least two nodes, it must have at least one edge connecting some pair of those nodes. If we pick any edge from $T$ and remove it, we split $T$ into two subtrees $T_1$ and $T_2$. Let $k$ be the number of nodes in $T_1$; then $T_2$ has $n - k$ nodes in it. We have that $k \geq 1$ and that $n - k \geq 1$, since each subtree must have at least one node in it. This means that $1 \leq k \leq n - 1$, so $1 \leq n - k \leq n - 1$. Thus by our inductive hypothesis, we know that $T_1$ must have $k - 1$ edges and $T_2$ must have $n - k - 1$ edges. Collectively, these two trees thus have $n - k - 1 + (k - 1) = n - 2$ edges. Adding in the initial edge we deleted, we have a total of $n - 1$ edges in $T$. Thus $P(n)$ holds.

In either case, $P(n)$ holds, completing the induction. ■

As another example of the sort of inductive proof we can do on trees, let's take one more look at the structure of trees. Consider the following trees:

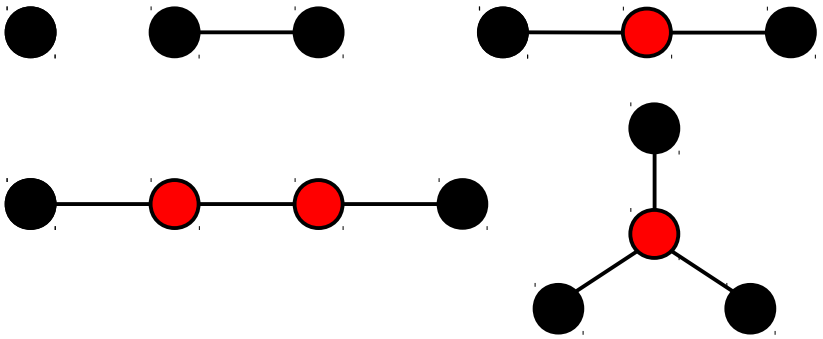


The nodes in these trees can be broadly categorized into two different groups. Some of these nodes (highlighted in red) help form the "backbone" of the tree and keep the rest of the nodes connected. Other nodes (highlighted in black) just hang off of the tree. Many important proofs or algorithms on trees make a distinction between these types of nodes, so we'll introduce some new terminology in order to make it easier to discuss the pieces of a tree:

A ***leaf node*** in a tree is a node with at most one edge connected to it. An ***internal node*** is a node in a tree that is not a leaf.

All of the trees we've seen so far have had at least one leaf node in them. Even the "trivial tree" with just one node has a leaf in it. Is it a coincidence that all these trees have leaves? Or is this an essential property of trees?

It turns out that we can guarantee that every tree has to have leaves. We can actually prove a fairly strong result: every tree has at least one leaf node, and any tree with at least two nodes has at least two leaves. You can see this by example if we start listing off all of the trees with 1, 2, 3, and 4 nodes:



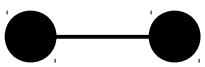
Now, how exactly might we go about proving this? Like most important results in mathematics, there are many approaches that could be made to work. The approach we'll use in this section is a beautiful application over induction on trees. Our intuition will be to do an induction over the number of nodes in the tree. It's pretty clear that the unique tree with just one node has a leaf node. The interesting case is the inductive step. Let's suppose that for all trees of size at most $n$, we know that the tree either has a single leaf node and has size one, or has at least two leaf nodes. How might we use this fact to establish that every $n$-node tree must have at least two leaves?

As before, we'll begin by picking any edge in the tree and cutting it, leaving us with two smaller subtrees. When we do this, each subtree will either have exactly one node in it, or it will have at least two nodes. In the event that a subtree has exactly one node, then that node is a leaf in the corresponding subtree. If it has two or more nodes, then there will be at least two leaves. Consequently, there are four possible options for what might happen when we split the tree into two subtrees $T_1$ and $T_2$:

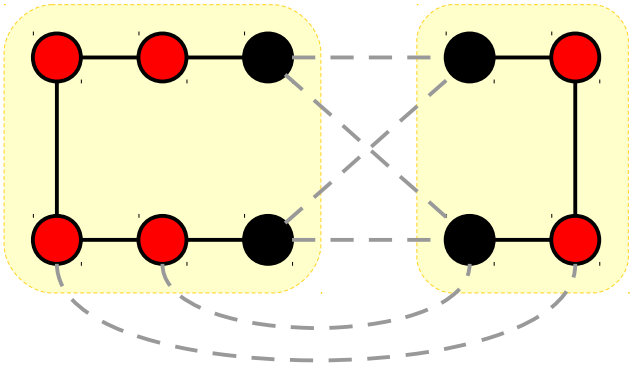
- $T_1$ and $T_2$ each have exactly one node.
- $T_1$ has exactly one node and $T_2$ has more than one node.
- $T_1$ has more than one node and $T_2$ has exactly one node.
- $T_1$ and $T_2$ each have more than one node.

Of these four cases, the middle two are symmetric; we can consider this as the more general case "one subtree has exactly one node and one subtree has at least two nodes."

In each of these three cases, we need to show that the original tree of $n$ nodes (let's assume $n \geq 2$ here, since otherwise we'd just use the base case) has at least two leaves. We can see this in each of the tree cases. First, if both subtrees have exactly one node, then the original tree must be this tree here:

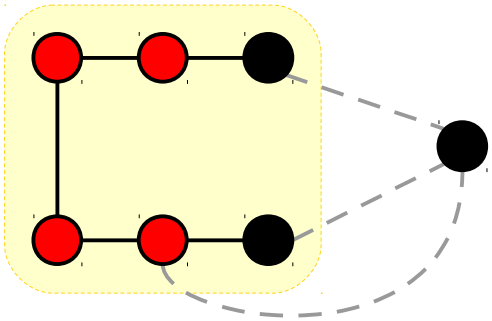


So we're done. Next, if both subtrees have at least two leaves, think about what happens when we add in the edge that we initially deleted from our tree. This edge will be incident to one node from each of the subtrees. Since each of the subtrees has at least two leaves, we can guarantee that in adding in this edge, at least one of the leaves from each of the subtrees hasn't had any edges added to it, meaning that those two nodes must be leaves in the new tree. You can see this below:



Here, we have two separate trees, one on the left and on on the right. If we connect the two trees by adding any of the dashed edges between the black leaf nodes, then at least one black node from each side will still be a leaf in the resulting. If we connect any red internal node to one of the black leaf nodes, then three of the black leaf nodes will still be leaves. Finally, if we connect any two red internal nodes together, all four of the black leaf nodes will still be leaves.

All that's left to do is think about the case where one of the subtrees has exactly one node in it. In that case, we can use some of the above reasoning to guarantee that when we add in the initial edge that we disconnected, at least one of the leaves from the large subtree will remain a leaf node in the new tree. What about the other subtree? Well, since the one-node subtree has (unsurprisingly) just one node in it, that node can't have any edges connected to it. When we then add in the original edge that we deleted, this node ends up having exactly one edge connected to it. Consequently, it also must be a leaf in the new tree. Combined with the leaf we got from the larger subtree, we end up with at least two leaves, just is as required. This is illustrated below.



Now, if we connect either of the leaves on the left-hand side to the isolated leaf on the right, then the leaf on the right stays a leaf in the new tree, while the unchosen leaf from the old tree remains a leaf. Otherwise, if we connect the isolated right-hand leaf to an internal node of the old tree, all the leaves of the old tree, plus the right-hand leaf, will be leaves in the new tree.

All that's left to do now is to formalize this reasoning with a proof:

***Theorem:*** Let $T = (V, E)$ be a tree. If $|V| = 1$, then $V$ has exactly one leaf node. Otherwise, $V$ has at least two leaf nodes.

***Proof:*** By induction. Let $P(n)$ be "any tree with $n$ nodes has exactly one leaf if $n = 1$ and at least two leaves if $n \geq 2$." We prove $P(n)$ is true for all $n \in \mathbb{N}^+$ by induction on $n$.

Assume that for some $n \in \mathbb{N}^+$, that for all $n' \in \mathbb{N}^+$ with $n' < n$, that $P(n')$ holds and any tree with $n'$ nodes either has one node and exactly one leaf, or has at least two nodes and at least two leaves. We will prove that $P(n)$ holds in this case.

First, if $n = 1$, then the only tree with $n$ nodes is one with a single isolated node. This node has no edges connected to it, so it is a leaf. Thus $P(n)$ holds.

Otherwise, assume that $n \geq 2$ and consider any tree $T$ with $n$ nodes. Choose any edge of $T$ and remove it; this splits $T$ into two subtrees $T_1$ and $T_2$. We now consider three cases about the relative sizes of $T_1$ and $T_2$.

*Case 1:* $T_1$ and $T_2$ each have one node. This means that the tree $T$ has exactly two nodes, so it has exactly one edge. Thus each of the nodes in $T$ are leaves, since they are incident to just one edge each, and so $T$ has at least two leaves.

*Case 2:* Both $T_1$ and $T_2$ have at least two nodes. By the inductive hypothesis, this means that $T_1$ and $T_2$ each have two leaf nodes, meaning that there are at least four nodes in the graph that have at most one edge touching them. When we add back to $T$ the initial edge that we deleted, this new edge can be incident to at most two of these nodes. Consequently, the other two nodes must still have at most one edge incident to them, and so they are still leaves in the overall graph $T$. Thus $T$ has at least two leaves.

*Case 3:* One of $T_1$ and $T_2$ has exactly one node, and the other has at least two. Without loss of generality, assume that $T_1$ has one node $u$, and that $T_2$ has at least two. By the inductive hypothesis, $T_2$ has at least two leaves. Also by the inductive hypothesis, $T_1$'s sole node $u$ must be a leaf, and moreover it must have no edges incident to it, because any one-node graph must have no edges. When we add back to $T$ the initial edge that we deleted, this edge will be incident to $u$ and incident to at most one of the at least two leaves from $T_2$. This means that there are now at least two leaves in $T$: first, the node $u$, which now has exactly one edge incident to it, and second, one of the leaves from $T_2$ that is not incident to the new edge. Thus $T$ has at least two leaves.

Thus in all three cases $T$ has at least two leaves, so $P(n)$ holds, completing the induction. ■

This proof makes it possible for us to inductively prove results about trees in a different way. Because each tree has at least one leaf, we can proceed using weak induction on the number of nodes in the tree as follows:

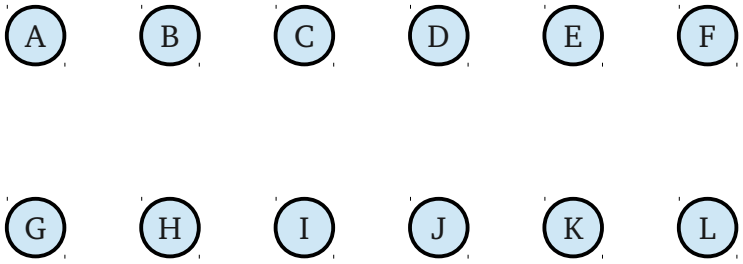
- Prove the base case directly for the one-node tree.
- Prove the inductive step by starting with an ($n$+1)-node tree, removing a leaf, and then applying the inductive hypothesis to the $n$-node tree that remains.

### 4.2.4 Directed Connectivity

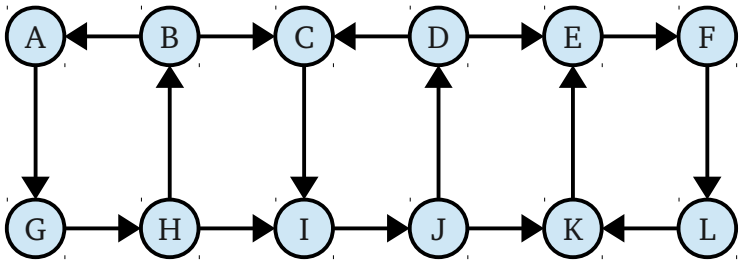
This section on graph connectivity has primarily focused on undirected graphs and undirected connectivity. What happens if we focus instead on directed graphs? How does this affect our definition of connectivity?

To motivate this discussion, suppose that you are laying out streets in a city and want to make the streets one-way. You want to ensure that it's possible to get from any location in the city to any other location in the city without going the wrong way on any street. How should you lay out the streets?

This is not as simple as it might look. For example, suppose that the locations in the city are laid out as follows:

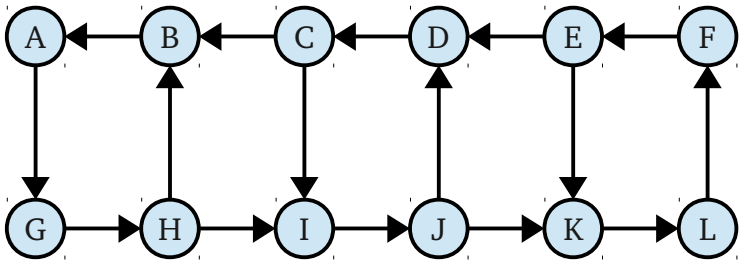



Now, consider the following way of laying out roads:




Notice that it's not always possible to get from any intersection to any other intersection. For example, you cannot get to location A starting from location C. This road network has the property that if you ignore directionality, the city is connected (in an undirected connectivity sense), but it is *not* connected if you are forced to follow the one-way streets.

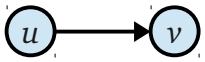
A better layout would be this one, where every location is reachable from every other location:



When we discussed undirected connectivity, we said that two nodes were *connected* iff there was a path between them. We denoted this $u \leftrightarrow v$. In order to discuss directed connectivity, we will introduce a few similar terms. First, we need a way of indicating that it's possible to start off at one node in a graph and arrive at some other node. This is given below with a definition that is almost identical to the analogous definition for undirected graphs:

> Let $G = (V, E)$ be an undirected graph. A node $v \in V$ is said to be ***reachable from*** a node $u \in V$ iff there is a path in $G$ from $u$ to $v$. We denote this $u \rightarrow v$.

Note that since we are considering directed graphs, it is not necessarily the case that if $u \rightarrow v$, then $v \rightarrow u$. This is a major asymmetry between directed and undirected graphs. For example, in the very simple directed graph below, it is true that $u \rightarrow v$, but it is *not* true that $v \rightarrow u$:




The behavior of reachability is very similar to the behavior of connectivity. Earlier, we proved several properties of undirected connectivity, such as the fact that $x \leftrightarrow x$ is always true, that $x \leftrightarrow y$ and $y \leftrightarrow z$ implies $x \leftrightarrow z$, etc. We can prove similar properties for reachability as well:

> ***Theorem:*** Let $G = (V, E)$ be a directed graph. Then:
>
> If $v \in V$, then $v \rightarrow v$.
> If $x, y, z \in V$ and $x \rightarrow y$ and $y \rightarrow z$, then $x \rightarrow z$.

The proof of this theorem is so similar to the proof about undirected connectivity that in the interest of space and simplicity we'll omit it here. It's good to try writing this proof out yourself, though, to double-check that you're comfortable with this style of proof.

It does sometimes happen that in a directed graph, there are a pair of nodes $u$ and $v$ where $u \rightarrow v$ and $v \rightarrow u$. This means that it's possible to start off at $u$, head over to $v$, and then come back to $u$. The paths there and back may be (and often are) completely different from one another, but we know that they exist. If this occurs, we say that the nodes are *strongly connected* to one another:

> If $u$ and $v$ are nodes in a directed graph, we say that $u$ and $v$ are ***strongly connected*** iff $u \rightarrow v$ and $v \rightarrow u$. If $u$ and $v$ are strongly connected, we denote this as $u \leftrightarrow v$.

The fact that we use the $\leftrightarrow$ symbol to mean both "connected in an undirected graph" and "strongly connected" might seem a bit odd here, but conceptually they mean the same thing. After all, in an undirected graph, if there's a path from $u$ to $v$, there is guaranteed to be a path back from $v$ to $u$.

Many properties of undirected connectivity also hold for directed connectivity, as evidenced by this theorem:

***Theorem:*** Let $G = (V, E)$ be a directed graph. Then:

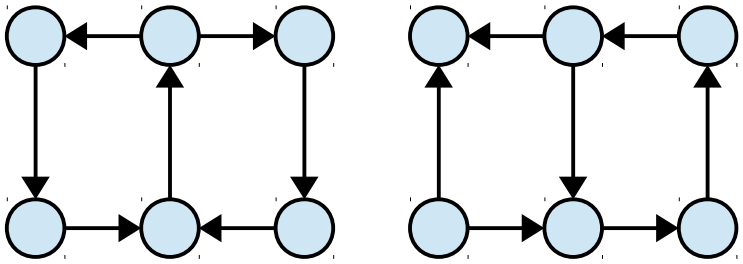
If $v \in V$, then $v \leftrightarrow v$.
If $u, v \in V$ and $u \leftrightarrow v$, then $v \leftrightarrow u$.
If $x, y, z \in V$ and $x \leftrightarrow y$ and $y \leftrightarrow z$, then $x \leftrightarrow z$.

As above, this proof is so similar to the proof for directed connectivity that we will omit it. You can prove this theorem using the properties of $\rightarrow$ that we listed above, and doing so is a reasonably straightforward exercise.

Given our definition of strong connectivity, we can define what it means for all nodes in a directed graph to be reachable from one another:

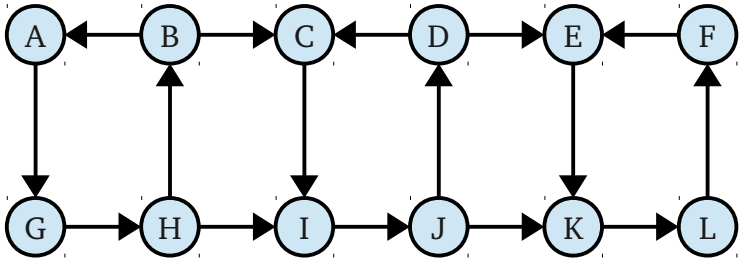
A directed graph $G = (V, E)$ is called ***strongly connected*** iff for any $u, v \in V$, that $u \leftrightarrow v$.

For example, the graph on the left is strongly connected, while the graph on the right is not:

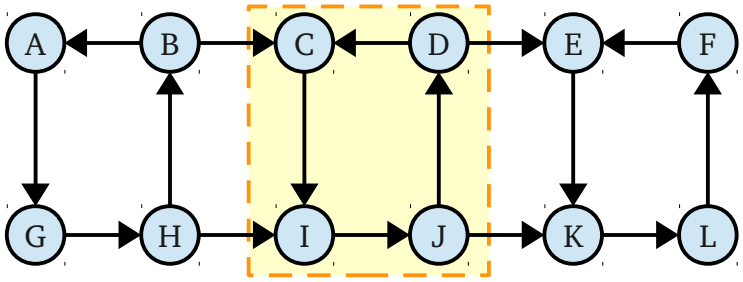


So far, our discussion of strong connectivity has paralleled our discussion of undirected connectivity. One of the interesting properties of undirected connectivity that we explored were connected components, regions of an undirected graph that are connected to one another. The connected components of a graph, intuitively, formed "pieces" of the graph. Can we extend this definition to strongly connected components?

To motivate this, suppose that you live in a city whose roads have been very poorly planned. All of the streets are one-way streets, but not every location is reachable from every other location. As a result, if you go driving, there is no guarantee that you'll be able to get back to where you started from. For example, let's suppose that the roads are laid out like this:




Let's suppose that you start at location C. Suppose that you want to go out for a drive, but don't want to go anywhere from which you can't get back to where you started. Where could you safely go? With a bit of thought, you can see that these locations are the ones indicated here:




The reason for this is as follows. Any of the locations A, B, G, or H can't be reached from where you're starting, so there's no possible way that you could go there. Any of the locations E, F, K, or L can be reached from C, but if you go to them you'll not be able to return from where you started. The remaining locations are reachable from your starting point, and have a return back back to the start.

If you think about the nodes that you can visit, they're precisely the nodes that are strongly connected to your initial starting point. All of the other nodes in the graph, for some reason or another, aren't strongly connected to C. Consequently, you either can't visit them, or can't visit them and return to your starting point. In other words, you can think of the set of nodes that you can reach from C as the set

$$S = \{ v \in V \mid v \leftrightarrow \text{C} \}$$

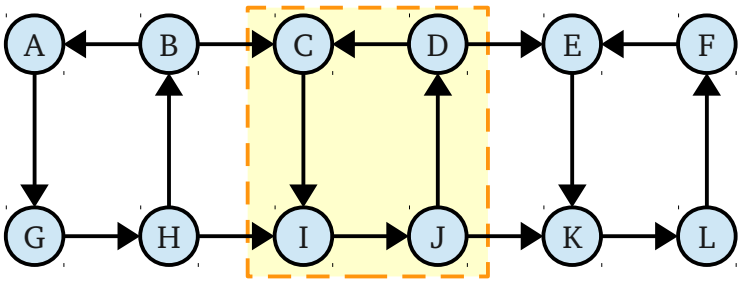
Does this set definition seem familiar from anywhere? If you'll recall, when we were discussing connected components, we ended up proving that every node in a graph belongs to a connected component by constructing sets that look almost identically the same as the above set, though with $\leftrightarrow$ referring to undirected connectivity rather than strong connectivity. In a sense, the above set forms a "strongly connected" component, a group of nodes mutually reachable from one another. In fact, that's precisely what it is.

When discussing connected components, our goal was to break a graph down into a set of smaller "pieces." To define what a "piece" was, we reasoned that it would be a maximal set of nodes that were all connected to one another. What if we repeat this analysis for directed graphs? Our goal this time will be to split the graph into smaller "pieces" based on connectivity, except this time we will use *strong* connectivity, since this is a directed graph, rather than undirected connectivity. This gives rise to the following definition:

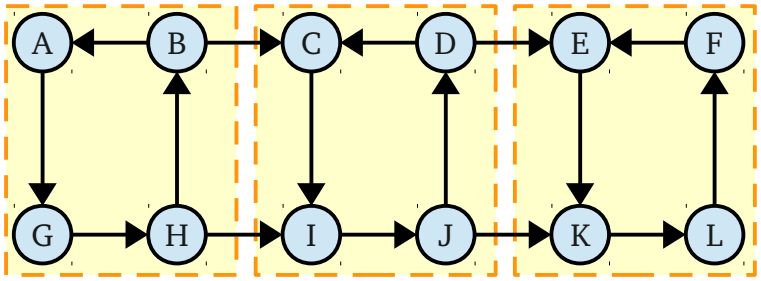
> Let $G = (V, E)$ be a directed graph. A ***strongly connected component*** (or ***SCC***) of $G$ is a nonempty set of nodes $C$ (that is, $C \subseteq V$), such that
>
> (1) For any $u, v \in C$, we have $u \leftrightarrow v$.
> (2) For any $u \in C$ and $v \in V - C$, we have $u \nleftrightarrow v$.

This definition is identically the same as the definition of a normal (undirected) connected component, except that we have replaced connectivity with strong connectivity.

Right now all we have is a definition of a strongly connected component without much of a feel for what they are. To get a better intuition for strongly connected components, let's return to our initial example of driving around a badly-planned city. We recognized that, starting at C, we could not necessarily drive anywhere and then drive back, either because we couldn't drive to the destination at all, or because if we did drive there, we'd get stuck and unable to drive back. The set of nodes reachable from our starting point formed a strongly connected component, which we saw here:




However, this isn't the only strongly connected component in the graph, and in fact there are two more of them, which are shown here:




Notice how all the nodes in a strongly connected component are mutually reachable from one another. Also notice that in the above graph, every node belongs to exactly one strongly connected component. It turns out that the following theorem is true, and the proof is pretty much the same one we had for normal undirected connected components:

> ***Theorem:*** Every node in a directed graph *G* belongs to exactly one strongly connected component.

This theorem implies that no two strongly connected components can overlap one another. This means that we can think about the structure of how the strongly connected components of a graph relate to one another. This is a fascinating topic, and we'll cover it later in this chapter.

# 4.3 DAGs

The idea of a graph is very general and encompasses many different types of structures – it can encode both the connections in the human brain and preferences for different types of food. Often, it is useful to focus on specific types of graphs that are useful for encoding particular connections between objects. Much of this chapter will focus on specific sorts of graphs and their applications. This particular section focuses on one type of graph called a *directed acyclic graph* that arises in many contexts in computer science.

Let's suppose that you are interested in making pancakes. Here is a simple pancake recipe taken from all-recipes.com:*

* Taken from http://allrecipes.com/recipe/good-old-fashioned-pancakes/, accessed on 21 July, 2012. Recipe highly recommended.

| | |
|---|---|
| 1 1/2 cups all-purpose flour | 1 tablespoon white sugar |
| 3 1/2 teaspoons baking powder | 1 1/4 cups milk |
| 1 teaspoon salt | 1 egg |
| 3 tablespoons butter, melted | |

In a large bowl, sift together the flour, baking powder, salt and sugar. Make a well in the center and pour in the milk, egg and melted butter; mix until smooth.

Heat a lightly oiled griddle or frying pan over medium high heat. Pour or scoop the batter onto the griddle, using approximately 1/4 cup for each pancake. Brown on both sides and serve hot.

Now, suppose that you and your friends are interested in making a batch of these pancakes. To do so, you need to distribute the tasks involved in the recipe. You'd like the work to be distributed in a way where the job will get finished as fast as possible. How would you distribute these tasks?

Let's think of what tasks are involved in this recipe:

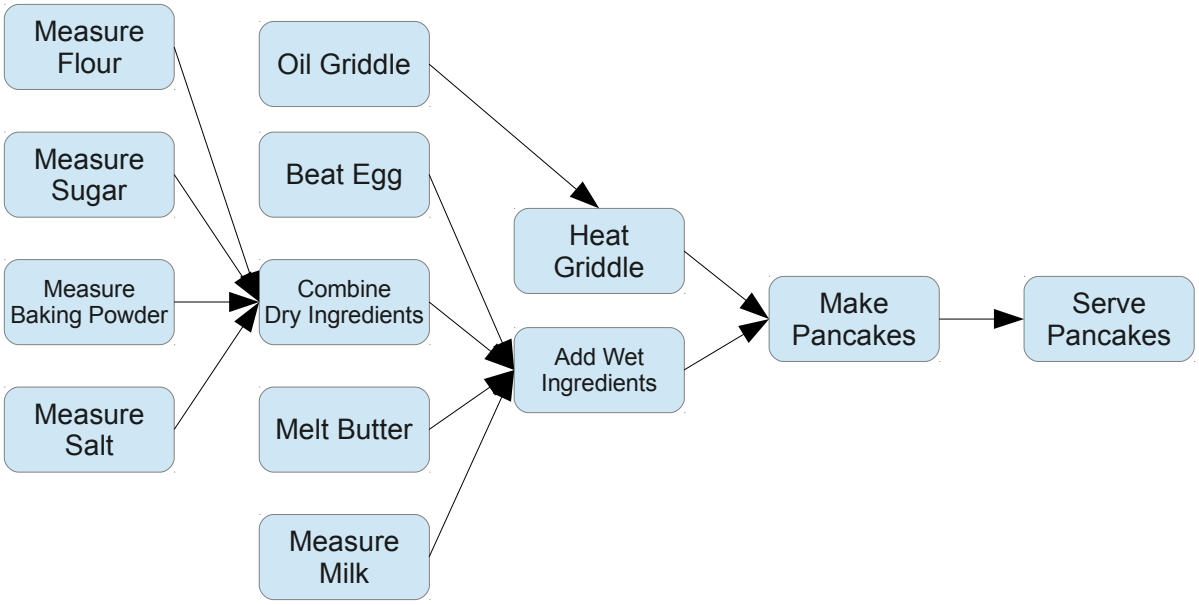
- Measure the flour, baking powder, salt, sugar, and milk.
- Melt the butter.
- Beat the egg.
- Combine dry ingredients.
- Mix in wet ingredients.
- Oil the griddle.
- Head the griddle.
- Make pancakes.
- Serve pancakes.

For you and your friends to make these pancakes as quickly as possible, you would probably want to work on a lot of these tasks in parallel. For example, while one person measures the flour, another could beat the egg, measure milk, etc. However, not all tasks can be run in parallel. For example, you can't mix in the wet ingredients at the same time that you're measuring the dry ingredients, since those ingredients need to be combined together first. Similarly, you can't serve the pancakes at the same time that you're melting the butter, since there aren't any pancakes yet to serve!

The issue is that some of the above tasks depend on one another. Given that this is the case, you want to find a way to get as much done in parallel as possible, subject to the restriction that no task is performed until all of its prerequisite tasks have been taken care of first.

So how exactly do we do this? As with much of mathematics and computer science, we will do this in two steps. First, let's see what mathematical structures we might use to model the problem. Once we have that model, we can then analyze it to find the most efficient way of parallelizing the tasks.

As you might have guessed, we will formalize the cooking constraints as a graph. Specifically, we'll use a directed graph where each node represents a task, and there is an edge from one task to another if the second task directly depends on the first. Here is what this graph would look like for our cooking tasks:



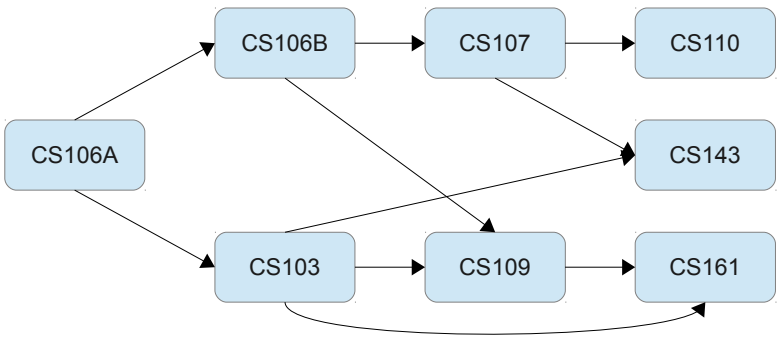

Given this graph, it's a lot easier to see how we might parallelize the tasks. If you have a total of five people working on pancakes, you could distribute the tasks as follows: initially, have everyone work on measuring one of the dry ingredients. Next, have someone combine everything, one person work on each liquid ingredient, and one person oil the griddle. As a next step, have someone heat the griddle while someone else combines everything. Then, make the pancakes, and then serve them. This process is about as efficient as it gets – pretty much everyone is working on something up until there aren't enough tasks left to go around!

The important thing to note about the above graph is that it is indeed possible to perform all the tasks in some order. Although there are dependencies in the graph, there is one way of ordering the tasks so that there are no conflicts. Consequently, we eventually do get to eat our pancakes.

Let's now switch gears and consider a totally different example. Suppose that you have a set of classes that you need to take (say, because you're interested in doing a CS major or CS minor) and want to find out what order you should take your classes in. You have before you the list of classes you want to take, along with their prerequisites. For example, in the Stanford CS department, you might have this set of classes:

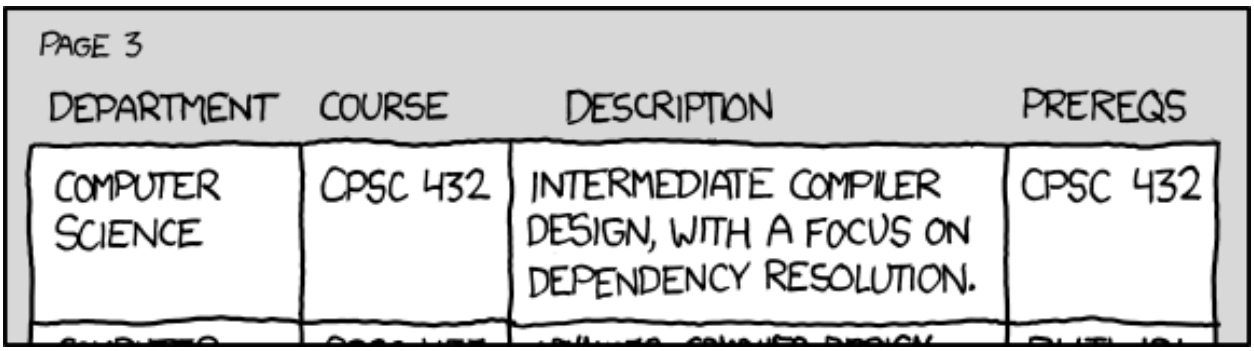
| Number | Title | Description | Prerequisites |
|---|---|---|---|
| CS103 | Mathematical Foundations of Computing | Discrete math, proof techniques, computability, and complexity. | CS106A |
| CS106A | Programming Methodology | Introduction to computer programming and software engineering in Java. | None |
| CS106B | Programming Abstractions | Data structures, recursion, algorithmic analysis, and graph algorithms. | CS106A |
| CS107 | Computer Organization and Systems | Data representations, assembly language, memory organization. | CS106B |
| CS109 | Introduction to Probability for Computer Scientists | Combinatorics, probability, independence, expectation, and machine learning. | CS103, CS106B |
| CS110 | Principles of Computer Systems | Software engineering techniques for computer systems. | CS107 |
| CS143 | Compilers | Techniques in compiler construction and code optimization. | CS103, CS107 |
| CS161 | Design and Analysis of Algorithms | Big-O, recurrences, greedy algorithms, randomization, and dynamic programming. | CS103, CS109 |

How would you order these classes so that you can take all of them without violating any of the prerequisites? Just looking over this table, it's a bit tricky to see a valid ordering (unless you already happen to know one). However, if we represent these dependencies as a graph, it becomes a lot easier to find a legal structure. As before, we'll build a graph where each node represents a class, and an edge from one class to another means that the second class has the first as a prerequisite. If we do this, we get the following graph:



From here, we can start seeing how to order the courses. For example, one ordering would be to do CS106A first, then CS106B and CS103 in parallel, then CS107 and CS109, followed by CS110, CS143, and CS161. Of course, that's a pretty crazy course schedule, so you could also take the more reasonable CS106A, then CS106B, then CS103, then CS107, then CS109, then CS161, and finally CS143.

As with the baking question, this graph has the nice property that even though there are dependencies between courses, the courses can indeed be ordered such that they can all be completed in order. Not all course schedules have this property; here's one possible class schedule where it's impossible to take all classes with their prerequisites, courtesy of Randall Munroe:*




## 4.3.1 Topological Orderings

Both of the previous examples – baking tasks and course scheduling – were easily modeled as dependency graphs, where nodes and edges correspond to restrictions on how various objects can be ordered. The essential properties of these graphs were the following:

- The graphs were directed, so that dependency orders could be enforced, and
- There was some order in which the tasks could be completed without breaking any dependencies.

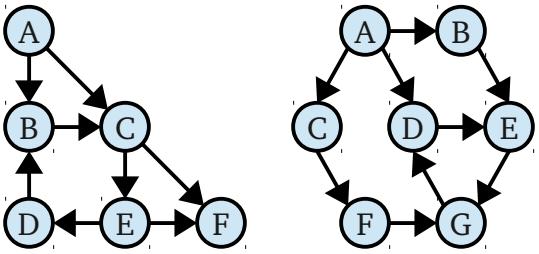
We already have a formalization for the first of these requirements (directed graphs). Can we formalize this second constraint? Using the new language of graph theory, it turns out that it is possible to do so.

> Let $G = (V, E)$ be a directed graph. A ***topological ordering*** of $G$ is an ordering $v_1, v_2, \ldots, v_n$ of the nodes in $G$ such that if $i \leq j$, then there is no edge from $v_j$ to $v_i$.

Don't let the term “topological ordering” throw you off; this definition is not as complex as it may seem. Stated simply, a topological ordering is a way of listing off all the nodes in a graph so that no node is listed before any of the nodes that it depends on. For example, in the class listing example, a topological ordering corresponds to a sequence of classes that you can take while respecting prerequisites, while in the case of cooking a topological ordering is a way of performing all the steps in the recipe without performing any one task before all the others it depends on.

Not all graphs have topological orderings. For example, neither of these graphs can be topologically ordered:

* http://xkcd.com/754



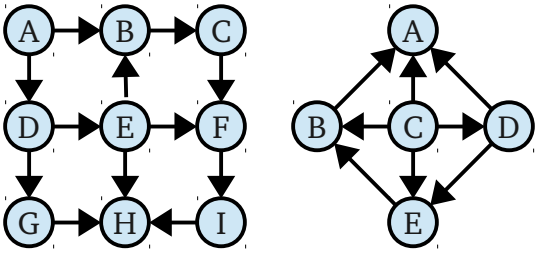

Intuitively, the graph on the left can't be topologically ordered because no matter how you try to order the nodes B, C, D, and E, one of the nodes will have to come before one of the nodes it depends on. Similarly, the graph on the right has the same problem with the nodes D, E, and F.

What sorts of graphs do have a topological ordering? Graphs with these properties arise frequently in computer science, and we give them a special name: the *directed acyclic graph*.

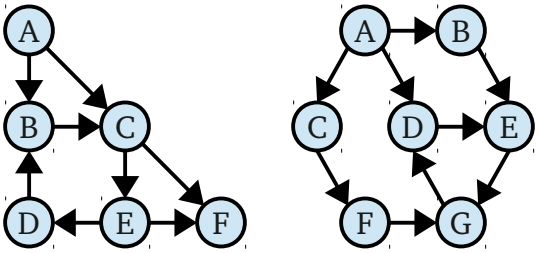
> A ***directed acyclic graph***, or ***DAG*** for short, is a directed graph containing no cycles.

Let's take a minute to look at exactly what this means. We've talked about directed graphs before, but what about "acyclic" graphs? If you'll recall, a cycle in a graph is a series of nodes that trace out a path from some node back to itself. An acyclic graph is graph that contains no cycles. Intuitively, this means that if you trace out any path in the graph starting from any node, there is no possible way to return to where you have started.

Both of the dependency graphs from before are DAGs – you can verify this by trying to find a path from any node back to itself – as are the following graphs:




However, these graphs from before, which have no topological orderings, are also not DAGs:




Right now, we have two completely different definitions. On the one hand, we have topological orderings, which talk about a way that we can order the nodes in a graph. On the other hand, we have DAGs, which talk about the structural properties of graphs. When defining DAGs, we claimed that DAGs are precisely

the graphs that admit a topological ordering. Why exactly is this? How do we connect orderings of nodes to cycles?

To prove that this is true, we will prove two theorems that, taken together, establish that a graph is a DAG iff it has a topological ordering.

> ***Theorem:*** Let $G$ be a directed graph. If $G$ has a topological ordering, then $G$ contains no cycles.
>
> ***Theorem:*** Let $G$ be a directed graph. If $G$ contains no cycles, then $G$ has a topological ordering.

The rest of this section is dedicated to proving each of these results.

We will first prove that if a graph has a topological ordering, then it must be a DAG. This proof is actually not too difficult. The idea is as follows: rather than proving this directly, we'll use the contrapositive, namely, that if a graph contains a cycle, then it cannot have a topological ordering. Intuitively, you can think about this by considering a set of classes whose dependencies form a cycle. That is, class A depends on class B, which depends on class C, which eventually in turn depends again on class A. No matter how you try to order the classes, whichever class you try to take first must have some prerequisite that you haven't taken yet.

We can formalize this using the following proof.

> ***Theorem:*** Let $G$ be a directed graph. If $G$ has a topological ordering, then $G$ contains no cycles.

> ***Proof:*** By contrapositive; we will show that if $G$ contains a cycle, then $G$ does not have a topological ordering.
>
> Consider any graph $G$ containing a cycle $C$; let that cycle be $(v_1, v_2, \ldots, v_k, v_i)$. Now, consider any way of ordering the nodes of $G$. Consider the very first node $v_i$ of the cycle that appears in this ordering, and let it be at position $r$ in the ordering. Let $v_j$ be the node that comes before $v_i$ in the cycle, and let its position in the ordering be $s$. Since $v_i$ is the earliest node of the cycle to appear in the ordering, we know that $r \leq s$. However, since $v_j$ immediately precedes $v_i$ in the cycle, there must be an edge from $v_j$ to $v_i$. Consequently, this ordering of the nodes cannot be a topological ordering. Since our choice of ordering was arbitrary, this means that any ordering of the nodes of $G$ is not a topological ordering. Thus $G$ has no topological orderings, as required. ■

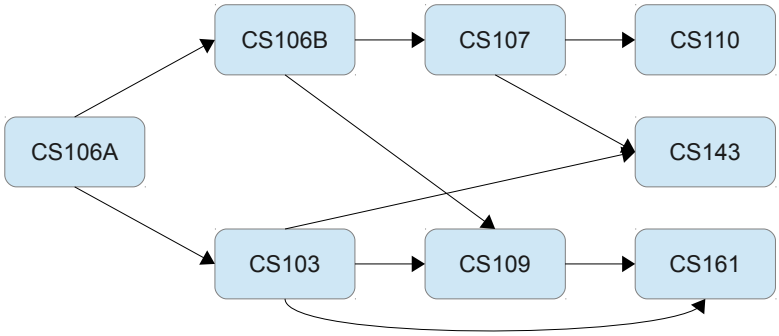
Notice how this proof is put together. We start off by identifying the cycle that we want to consider, and then think about any arbitrary ordering. We know that every node in the cycle must appear somewhere in this ordering, so there has to be a node in the cycle that appears in the ordering before any other node in the cycle (this is the node $v_i$). This means that every node in the cycle must appear after $v_i$ in the ordering. In particular, this means that the node that precedes the $v_i$ in the cycle (which we'll denote $v_j$) must come after $v_i$ in the ordering. Therefore, the ordering can't be a topological ordering; there is an edge from $v_j$ to $v_i$, even though $v_i$ comes before $v_j$.

An important detail of this proof is that we chose the earliest node in the ordering that appeared in the cycle. This made it possible to guarantee that the node that comes before it in the cycle must appear later on in the

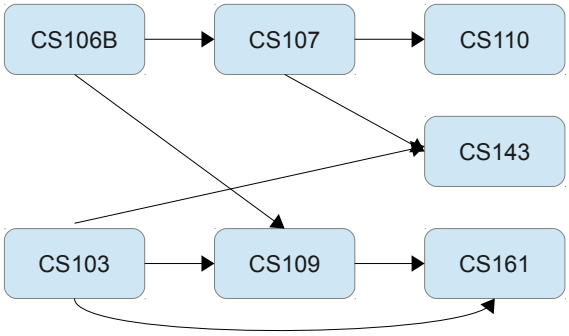
ordering than it. This is an excellent application of the well-ordering principle. If we picked an arbitrary node in the cycle, there is no guarantee that anything immediately connected to it would cause a problem for the topological ordering.

The second proof – that any DAG has a topological ordering – is more involved than our previous proof. In order to prove this result, we will have to explore the properties of DAGs in more depth.

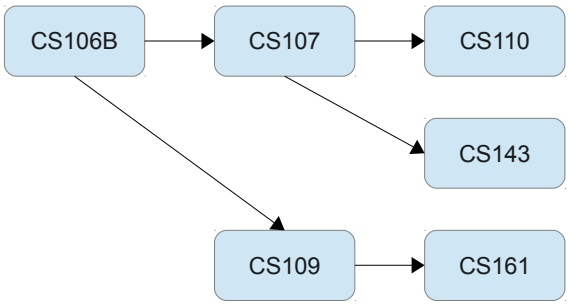
Intuitively, why is it always possible for us to topologically order the nodes of a DAG? To see why, let's consider an example. Consider the course prerequisites DAG from earlier in this chapter:



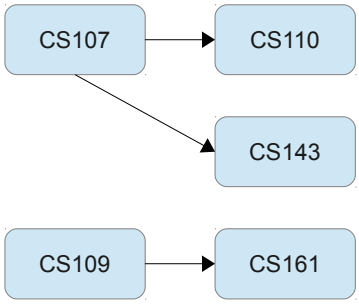

If we want to topologically order the nodes of this graph, we would have to start off with CS106A, since everything in the graph depends on it. Once we've taken that course, we're left with




Now, we can take either CS106B or CS103. Let's suppose that we want to take CS103 next. This leaves




Now, we have to take CS106B. Doing so leaves




So we can now take CS107 or CS109. Iterating this process over and over again will eventually give us a topological ordering of the graph.

Notice that at each point in this process, we are always able to find a node in the DAG that has no incoming edges. We can then take that node out of the DAG, place it next in the topological ordering, and then repeat this process over and over again. Eventually, we'll be left with a topological ordering.

To formalize this, let us introduce a few important definitions:

> Given a DAG *G*, a ***source*** in *G* is a node with no incoming edges. A ***sink*** is a node with no outgoing edges.

For example, in the graph of CS courses, CS106A is a source, while CS110, CS161, and CS143 are sinks.

Our previous discussion about topological orderings of DAGs can now be restated as follows: if the graph is a DAG, we can always find a source node in it. We can then remove it, place it at the front of our topological ordering, and repeat the process on what remains. Assuming that this is possible, it gives us a proof that any DAG has a topological ordering. Moreover, it gives us a constructive algorithm we can use to find a topological ordering – repeatedly find a source node and remove it from the graph.

To justify this algorithm, we need to introduce two key lemmas:

> ***Lemma 1:*** If *G* is a DAG with at least one node, *G* has at least one source.
>
> ***Lemma 2:*** If *G* is a DAG with at least one source node, the graph *G'* formed by removing that source node from *G* is also a DAG.

These lemmas, taken together, can be used to prove that any DAG has a topological ordering (we'll go over that proof at the end of this section). In the meantime, let's work through the proofs of both of these lemmas.*

First, how would we show that every DAG with at least one node has at least one source? In a sense, this seems obvious – any DAG that we can try must have this property. But to formally prove this, we'll need a stronger argument.

Let's see what we have to work with. The definition of a DAG is pretty minimal – all we know is that it's a directed graph with no cycles. If we want to show that this means that it has at least one source node, we'll have to somehow base that on the fact that it has no cycles. How might we use this fact? Well, one option would be to try a proof by contrapositive. Could we show that if a (nonempty) graph has no sources, then it cannot be a DAG? If it's not a DAG, that means that it must have a cycle in it. So now we have something to work with: let's try to prove that if a nonempty graph has no sources, then it cannot be a DAG.

Now, what can we say about a graph with no sources? Since the graph has no sources, we know that every node in the graph must have some incoming edge. In other words, starting at any node in the graph, we could walk *backwards* across the edges of the graph without ever getting stuck, because no matter what node we end up at, we are always able to walk backward one more step.

It's here where we can spot something fishy. Suppose that there are $n$ nodes in our graph. Start at any one of them, and take $n + 1$ steps backwards. In the course of doing so, you will have visited $n + 1$ nodes. But all these nodes can't be different, since there are only $n$ total nodes in the graph! Consequently, we had to have visited the same node twice. If that's the case, the path that we've traced out must have contained a cycle, because at some point we entered a node, followed some nodes, then reentered that node again. (Technically speaking we traced that cycle out in reverse, but it's still a cycle).

We can formalize this intuition here:

---

* Did you know that the plural of *lemma* is *lemmata*? You are now part of the .01% of people that know this scintillating fact!

***Lemma 1:*** Any DAG with at least one node contains a source.

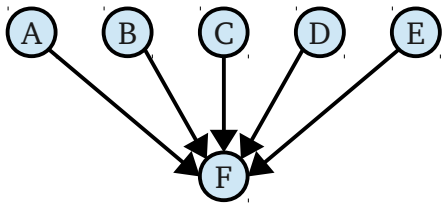
***Proof:*** Let $G$ be a nonempty graph. We will prove that if $G$ is a DAG, then $G$ contains a source by contrapositive; we instead show that if $G$ has no sources, then $G$ is not a DAG. Since $G$ contains no sources, there are no nodes in $G$ that have no incoming edges. Equivalently, every node in $G$ has at least one incoming edge.

Let $n$ be the total number of nodes in $G$. Now, starting from any node in $G$, begin following some edge entering $G$ in reverse. We will always be able to do this, since every node has at least one incoming edge. Repeat this process $n + 1$ times to get a series of nodes $v_{n+1}$, $v_n$, …, $v_1$ traced out this way. In other words, the sequence $v_1$, $v_2$, …, $v_{n+1}$ is a path in $G$.

Now, since there are $n + 1$ nodes in this path and only $n$ different nodes in the graph, the sequence $v_1$, $v_2$, …, $v_{n+1}$ must contain at least one duplicate node. Let $v_i$ be the first occurrence of this duplicated node and $v_j$ be the second, with $i \leq j$. Then the path $v_i$, $v_{i+1}$, …, $v_j$ starts and ends at the same node, so it is a cycle in $G$. Since $G$ contains a cycle, it is not a DAG, which is what we wanted to show. ■

The third paragraph of the above proof is the key step. It works by arguing that since we have visited $n + 1$ nodes in our path, but there are only $n$ total nodes in the graph, that we must have visited some node twice, from which we can derive the cycle. This is an example of a technique called the *pigeonhole principle*, a powerful technique in discrete mathematics. We will explore this technique in more depth in a later chapter.

With this lemma in hand, we now know that any nonempty DAG must have at least one source node. It turns out that DAGs can have *many* source nodes; as an example, here's a DAG where all but one of the nodes are sources:




The next step in our proof is to show that if we peel off any source node from a DAG, what we're left with is a DAG. This proof is much easier than the previous one. The intuition is that if we start off with a graph that contains no cycles and then start removing nodes from it, we can't ever introduce a cycle. This is formalized below:

***Lemma 2:*** Let $G$ be a DAG with source node $v$. Then the graph $G'$ formed by removing $v$ and all its outgoing edges is a DAG.

***Proof:*** By contradiction; assume that there is a DAG $G$ with source node $v$, but that $G'$ contains a cycle. Let that cycle $C$ be $x_1, x_2, \ldots, x_k, x_1$. Since $G'$ is $G$ with the node $v$ removed, we know that $v \notin C$. Consequently, all of the nodes $x_i$ and their corresponding edges must also be in $G$, meaning that $C$ is also a cycle in $G$. But this is impossible, since $G$ is a DAG. We have reached a contradiction, so our assumption must have been wrong and $G'$ must also be a DAG. ■

Now, we can combine both of these lemmas together to prove that any DAG has a topological ordering. To do so, we'll show that if you iteratively remove a source node and place it next in the ordering, then eventually you will build a topological ordering for the DAG. Since this process is iterative and keeps shrinking the DAG, our proof will be inductive on the number of nodes in the DAG.

***Theorem:*** Any DAG $G$ has a topological ordering.

***Proof:*** By induction; let $P(n)$ be "any DAG with $n$ nodes has a topological ordering." We will prove $P(n)$ is true for all $n \in \mathbb{N}$ by induction.

As our base case, we prove $P(0)$, that any DAG with 0 nodes has a topological ordering. This is true because a DAG with no nodes has a trivial topological ordering – just list all 0 of its nodes in order.

For the inductive step, assume that $P(n)$ holds for some $n \in \mathbb{N}$, meaning that any DAG with $n$ nodes has a topological ordering. We then prove $P(n + 1)$, that any DAG with $n + 1$ nodes has a topological ordering. Consider any DAG $G$ with $n + 1$ nodes; since this DAG is nonempty, it must have at least one source node. Pick any such source and call it $v_0$. We can then remove $s$ from $G$ to form a new DAG $G'$ with $n$ nodes. By our inductive hypothesis, this DAG has a topological ordering $v_1, v_2, \ldots, v_n$.

Now, order the nodes of $G$ as $v_0, v_1, v_2, \ldots, v_n$. We claim that this is a topological ordering of $G$. First, we can see that every node of $G$ appears exactly once in this ordering – if the node is $v_0$, it's at the front of the ordering, and every other node appears once as some $v_i$. Next, we show that for any nodes $v_i$ and $v_j$, if $i \leq j$, then there is no edge from $v_j$ to $v_i$. To see this, we consider two cases:

*Case 1:* $i = 0$. Since $v_0$ is a source, it has no incoming edges. Thus there cannot be any edges from $v_j$ to $v_0$ for any $j$.

*Case 2:* $i > 0$. Since $0 < i \leq j$, we know that $1 \leq i \leq j$. Consequently, both $v_i$ and $v_j$ are nodes in $G'$. Since $v_1, v_2, \ldots, v_n$ is a topological ordering of $G'$, this means that there is no edge from $v_j$ to $v_i$.

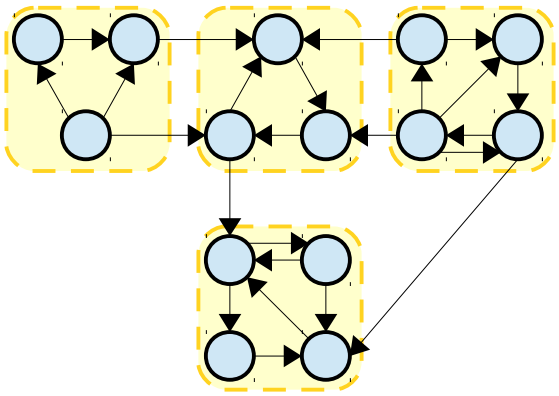
Thus in either case we have that there are no edges from $v_j$ to $v_i$ for any choice of $i$ and $j$. Consequently, $v_0, v_1, \ldots, v_n$ is a topological ordering of $G$. Since our choice of $G$ was arbitrary, this means that any DAG with $n + 1$ nodes has a topological ordering, completing the proof. ■

As a fitting coda to this section, notice that we have just proved that there is a topological ordering of a set of tasks (that is, a way of accomplishing all the tasks in some order) precisely when no tasks have circular dependencies. This might seem obvious, but it's always reassuring when we can use the vast power of mathematics to confirm our intuitions!

### 4.3.2 Condensations ★

As hinted at in the section about strongly connected components (SCCs), the ways in which strongly connected components relate is actually quite interesting. This section explores exactly what structures arise when we consider the connections between SCCs.

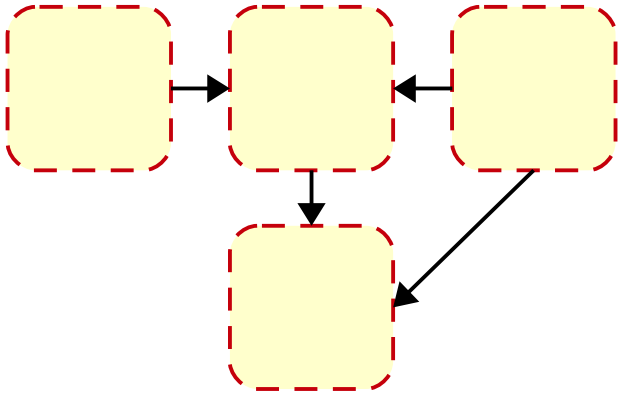
Consider, for example, the following graph, whose SCCs have been outlined:



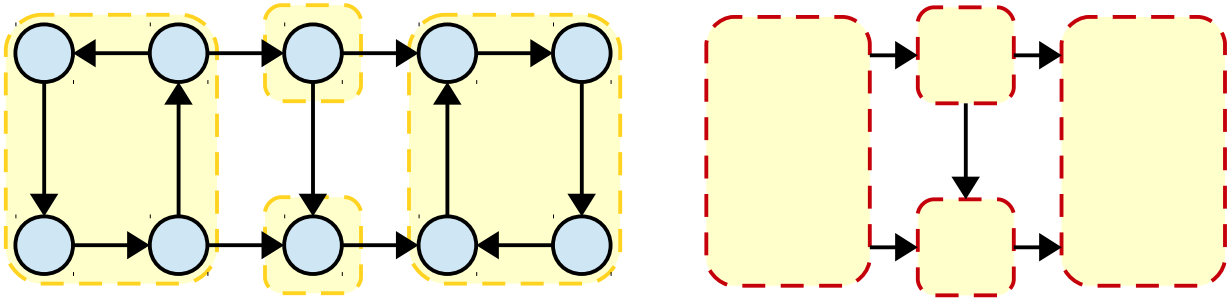
If you'll notice, there are some nodes in the leftmost SCC with edges to the top-middle SCC. There is also a node in the right SCC with an edge to the top-middle SCC, as well as a node with an edge to the bottom-middle SCC. The top-middle SCC in turn has several nodes with edges to the bottom-middle SCC. In this sense, we can think of various SCCs being linked directionally – one SCC might have edges from it that arrive in another SCC.

Given this idea as a starting point, we can consider a somewhat unusual idea. What if we were to construct, from the above graph, a new graph showing the original graph's SCCs and the connections between them? To do this, we would proceed as follows. Each node of this new graph will represent an SCC of the original graph. We will then add directed edges between two SCCs if there is a node in the first SCC with an edge to a node in the second SCC. Intuitively, you can think of this as follows. Draw a border around all of the strongly-connected components in the graph. After doing this, there might be some edges in the graph that start inside an SCC, but then cross the border into another SCC. These edges are then converted into new edges that run from the first SCC into the second.

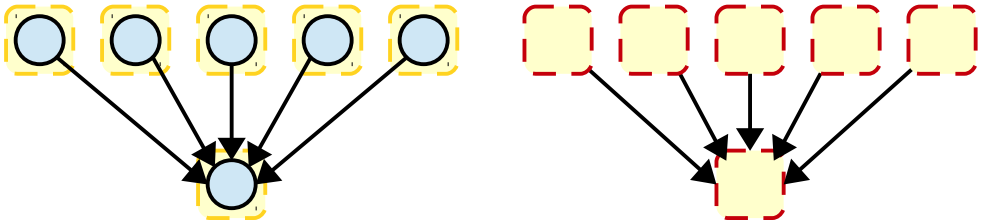
As an example, given the above graph, we would construct the graph of its SCCs as follows:



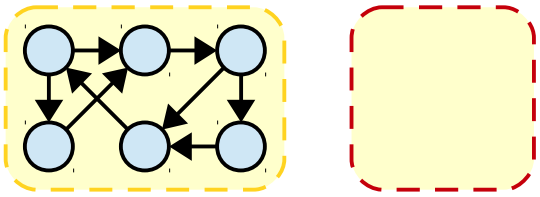
Here is another example of a graph, its SCCs, and the resulting graph:



Some directed graphs have very uninteresting SCCs consisting of just a single node each. In that case, the resulting graph is the same as the original. You can see this here:



On the other hand, some graphs have just one SCC, in which case the graph of the SCCs is just a single node:



Our goal in this section will be to analyze the properties of the graphs formed this way and to reason about their properties. Doing so ends up giving up some interesting insights into the structure of arbitrary directed graphs. But first, we need to formalize how we're constructing the above graph.

It turns out that formalizing this construction is not challenging, but can become notationally quite messy. In order to describe the graph of SCCs, we need to define what the nodes and edges of that graph are. So let's take an arbitrary directed graph $G = (V, E)$ as our starting point. From here, we will define two sets $V'$ and $E'$ that will be the nodes and edges of the new graph that we will form.

Of the two sets $V'$ and $E'$, the set $V'$ is easiest to define. Since we want a graph of all the SCCs in the original graph, we can just define $V'$ to be the SCCs of the original graph. More specifically, we'll say that

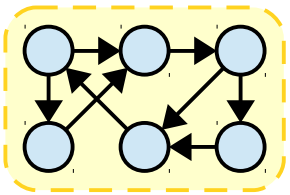
$$V' = \{ S \mid S \text{ is an SCC of } G \}$$

It may seem strange that the nodes in our new graph are *sets* of nodes from the original graph, but that's precisely what we're doing. Remember that the mathematical definition of a graph just says that we need a set of objects to act as nodes and a set of objects to act as edges. The objects that acts as nodes can be anything, including sets of other objects that themselves act as nodes in another graph. When drawing out the nodes of the new graph that we'll form, I will usually not draw each node as if it's an SCC of the original graph, but I will try to give some indication that the nodes themselves are sets.

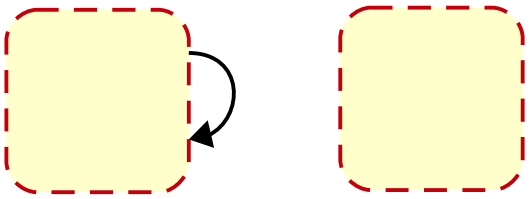
Now, we need to come up with a set of edges. This ends up being a bit trickier. Our intuition for the edges in the new graph was to take each edge whose nodes crossed from one SCC to another, then to convert that edge to an edge between the corresponding SCCs. Formally speaking, we're saying that there is an edge from SCC $S_1$ to SCC $S_2$ iff some node in $S_1$ has an edge to some node in $S_2$. If we were to write this out in set-builder notation, we might attempt to do so as follows:

$$E' = \{ (S_1, S_2) \mid \text{There exists } u \in S_1 \text{ and } v \in S_2 \text{ such that } (u, v) \in E \}$$

(Here, $E$ is the original edge set from the graph.) Unfortunately, this definition doesn't quite give us what we want. In particular, let's see what this definition would have us do on the following graph:



This graph is strongly connected, so there's just one SCC, which encompasses all the nodes. However, notice that there are plenty of edges between nodes in this SCC. If we look at the literal definition of $E'$, we should therefore have that in the SCC graph of the graph above, there is an edge from the SCC to itself. Under the current definition, this means that we'd get the graph on the left as our SCC graph, rather than the graph on the right (which is more what we intended):



As a result, we'll add one more constraint to our set of edges. Not only must there be an edge crossing from the first SCC to the second, but the two SCCs themselves must not be the same. This looks as follows:

$$E' = \{ (S_1, S_2) \mid S_1 \neq S_2, \text{ and there exists } u \in S_1 \text{ and } v \in S_2 \text{ such that } (u, v) \in E \}$$

Great! We now have a formal definition of the nodes and edges of our SCC graph. These two definitions, taken together, give us the following:

Let $G = (V, E)$ be a directed graph. The ***condensation of G***, denoted $G^{SCC}$, is a graph of the strongly-connected components of $G$ defined as follows:

Let $V' = \{ S \mid S \text{ is an SCC of } G \}$
Let $E' = \{ (S_1, S_2) \mid S_1 \neq S_2$, and there exists $u \in S_1$, $v \in S_2$ such that $(u, v) \in E \}$

Then $G^{SCC} = (V', E')$.

Note that we'll use the notation $G^{SCC}$ to describe the graph that we get from the SCCs of $G$, which as mentioned above is called the *condensation* of $G$.

And now for a pretty neat observation about condensations. Take a look at all of the condensations that we have seen so far. Interestingly, not a single one of them contains a cycle. This is particularly interesting, because it means that the structure of all of these graphs, most of which contain cycles, can be reduced down to DAGs, where each node is a strongly connected component of the original graph. If it turns out that this is true in general, and that the condensation of *any* graph $G$ is guaranteed to be a DAG, it shows that DAGs are a fundamentally important class of graph.

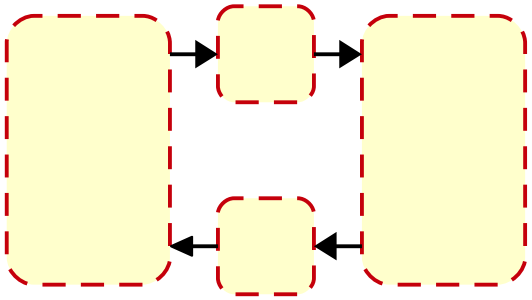
It turns out that, indeed, the above result is no coincidence, and in fact holds for all graphs. Formally, we have the following theorem:

***Theorem:*** For any directed graph $G$, the graph $G^{SCC}$ is a DAG.

Proving this theorem will require us to first gain an understanding about why this result is true, and second to build up the mathematical machinery necessary to prove it.

Let's start off with an intuition. Why must all condensations be DAGs? This has to do with the definition of an SCC. Recall that an SCC is, informally, a cluster of nodes that are all strongly connected to one another that cannot be grown any further. This means that if we pick any SCC from a graph and choose any node $v$ outside that SCC, then none of the nodes in the SCC are strongly connected to $v$. This means that either there is no path from any of the nodes in the SCC to $v$, or that there is no path from $v$ to any of the nodes in the SCC, or possibly both.

So now let's think about what it would mean if there was a condensation that wasn't a DAG. This graph would have to have a simple cycle in it, meaning that we could start off at one SCC, trace a path out to another SCC, and then trace a path back to the original SCC. For example:



Let's think about what this means. Consider the first edge of the path, which goes from one SCC to the next. This means that in the original graph, some node in the first SCC has an edge connecting it to some

node of the second SCC. Let's call these nodes $u$ and $v$, with $u$ in the first SCC and $v$ in the second. Since all of the nodes in the first SCC are reachable from one another, this means that it's possible to get from any node in the first SCC to $u$, and thus from any node in the first SCC to $v$. Since all the nodes in the second SCC are strongly connected, it's possible to get from $v$ to any node in the second SCC. Consequently, it's possible for any node in the first SCC to reach any node in the second SCC.

More generally, consider any path in the condensation. Any node in the first SCC in the path can reach any node in the second SCC in the path. By the same token, any node in the second SCC in the path can reach any node in the third, etc. In fact, any node in the first SCC on the path can reach any node in *any* of the SCCs on the path.

So what happens if the condensation isn't a DAG? Well, in that case, we'd have a simple cycle of SCCs that goes $(S_1, S_2, \ldots, S_n, S_1)$, where all the $S_i$'s are SCCs of the original graph. We can then split this simple cycle into two simple paths: $(S_1, S_2, \ldots, S_n)$, which goes from $S_1$ to $S_n$, and $(S_n, S_1)$ from $S_n$ to $S_1$. By our previous line of reasoning, this means that every node in $S_1$ can reach every node in $S_n$, and that every node in $S_n$ can reach $S_1$. But $S_1$ and $S_n$ are are supposed to be SCCs, which means that either the nodes in $S_1$ can't reach the nodes in $S_n$, or the nodes in $S_n$ can't reach $S_1$. This gives us a contradiction, meaning that we must have that the condensation is a DAG.

Let's review the structure of this proof before we formalize each individual step. First, we argued that if we have a path in the condensation, then any node in the first SCC on the path can reach any node in the last SCC on the path. This means that a path in the condensation gives us a way of finding one-directional connectivity in the original graph. Second, we argued that if there is a simple cycle in the condensation, it means that two different SCCs must be mutually reachable from each other, which causes a contradiction, because two disjoint SCCs can't be mutually reachable from one another.

Given this setup, let's prove each of these parts individually. The first part (about how a path in the condensation gives reachability information) we'll prove as a lemma for the second, which is the main theorem.

Here is the lemma we need to prove:

> ***Lemma:*** Let $G$ be a directed graph and $G^{SCC}$ its condensation. Then if there is a path $(S_1, S_2, \ldots, S_n)$ in $G^{SCC}$, then for any $u \in S_1$ and $v \in S_n$, we have $u \to v$.

To prove this lemma, we'll proceed by induction on the length of the path. Earlier, we sketched a line of reasoning showing that any adjacent SCCs in a path must have all nodes in the second SCC reachable from any of the nodes in the first SCC. We can then iterate this logic over each edge in the path to get the desired result.

***Proof:*** Consider any directed graph $G = (V, E)$ and let $G^{SCC}$ be its condensation. Then, Let $P(n)$ be "for any path $(S_1, S_2, \ldots, S_n)$ of length $n$ in $G^{SCC}$, if $u \in S_1$ and $v \in S_n$, then $u \to v$." We prove $P(n)$ is true for all $n \in \mathbb{N}^+$ by induction on $n$.

As our base case, we prove $P(1)$, for any path $(S_1)$ of length 1 in $G^{SCC}$, that if $u \in S_1$ and $v \in S_2$, that $u \to v$. Since $u \in S_1$ and $v \in S_1$ and $S_1$ is an SCC, we have that $u \leftrightarrow v$, which implies that $u \to v$ as required.

For our inductive step, assume that for some $n \in \mathbb{N}^+$ that $P(n)$ holds and for any path $(S_1, S_2, \ldots, S_n)$ in $G^{SCC}$, that if $u \in S_1$ and $v \in S_n$, that $u \to v$. We will prove that $P(n + 1)$ holds, meaning that for any path $(S_1, S_2, \ldots, S_n, S_{n+1})$ in $G^{SCC}$, that if $u \in S_1$ and $v \in S_{n+1}$, that $u \to v$.

Consider any path $(S_1, S_2, \ldots, S_n, S_{n+1})$ in $G^{SCC}$. This means that $(S_1, S_2, \ldots, S_n)$ is also a path in $G^{SCC}$. By our inductive hypothesis, this means that for any $u \in S_1$ and $x \in S_n$, $u \to x$. Since our initial path ends with $(S_n, S_{n+1})$, this edge exists in $G^{SCC}$, meaning that there exists some $y \in S_n$ and $z \in S_{n+1}$ such that $(y, z) \in E$, thus $y \to z$. Consequently, we know that for any $u \in S_1$, since $y \in S_n$, we have $u \to y$ and $y \to z$, so for any $u \in S_1$, we have that $u \to z$. Finally, since $S_{n+1}$ is an SCC and $z \in S_{n+1}$, we know that for any $v \in S_{n+1}$ that $z \to v$. Thus for any $u \in S_1$ and for any $v \in S_2$, we have that $u \to z$ and $z \to v$, so $u \to v$ as required. Thus $P(n + 1)$ holds, completing the induction. ■

Given this lemma, we can prove the following theorem:

***Theorem:*** For any directed graph $G$, the graph $G^{SCC}$ is a DAG.

***Proof:*** By contradiction; assume that there is a directed graph $G$ for which $G^{SCC}$ is not a DAG. Thus $G^{SCC}$ must contain a simple cycle. Note that, by construction, $G^{SCC}$ does not contain any edges from a node to itself. Thus this simple cycle must have the form $(S_1, S_2, \ldots, S_n, S_1)$, where $n \geq 2$.

We can split this simple cycle into two paths $(S_1, S_2, \ldots, S_n)$ from $S_1$ to $S_n$ and $(S_n, S_1)$ from $S_n$ to $S_1$. By our lemma, this means that for any $u \in S_1$ and $v \in S_n$, that $u \to v$ and $v \to u$. Consequently, $u \leftrightarrow v$. But since $S_1 \neq S_n$, this is a contradiction, because $u$ and $v$ are strongly connected but in different SCCs. We have reached a contradiction, so our assumption must have been incorrect. Thus $G^{SCC}$ must be a DAG. ■

One way of interpreting this theorem is as follows. Given any directed graph $G$, we can construct $G$ by starting with a collection of strongly connected graphs, ordering them in some way, and then adding edges between the nodes of those strongly connected graphs in a way that never adds an edge between a higher-numbered graph and a lower-numbered graph.

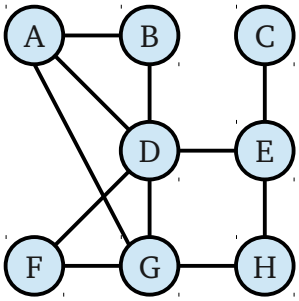
## 4.4 Matchings ★

Let's consider the following problem. Suppose that you have a group of people that you'd like to split into teams of two. Not everyone is willing to work with everyone else, though. (Although in theory we should

all just get along, in this case practice lags a bit behind the theory.) If you are given all of the pairs of people who might be willing to work together, how might you pair people up in the best possible way?

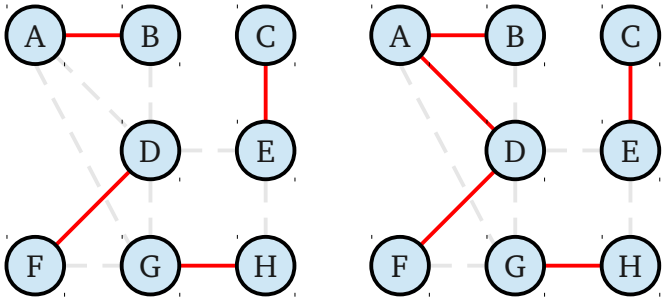
To answer this question, let's begin by building up a mathematical model of the problem. Once we have this model, we can start reasoning about it, and ultimately can arrive at a clean and elegant solution.

The first question we have to answer is how we could represent this problem mathematically. We are given two pieces of information: first, a list of all the people we need to pair up, and second, a list of all pairs of people that might be willing to work together. With a bit of thought, we can see that this arrangement could be represented using a graph. We'll represent each person with a node, and connect by edges pairs of people that might be willing to work with one another. For example, here is one possible graph:




In this graph, person A is comfortable working with people B, D, and G. Person H will work with either person E or G, but person C only wants to work with person E.

Our goal is to break this group of people up into pairs of people who are willing to work together. We've decided to represent each possible pair as an edge, and so if we're trying to choose how to pair people up, we must be looking for a way of choosing which of these edges we want to use. However, not all sets of edges can work. For example, consider the following two possible sets of edges:




On the left, we have a set of edges that represents a valid way of pairing up people within this group. The right side, however, does not give a valid pairing. The reason for this is as follows. Take a look at person D. Notice that we have chosen two edges incident to her. As a result, it seems like we should pair her up with A, as well as with F. This is a problem, since we'd like to make sure that everyone is assigned to just one pair.

Given the three above cases, it's clear that certain sets of edges are “better” than others for this problem. Before we start trying to solve the problem, let's begin by defining some basic terminology that we can use to reason about what sets of edges are valid, and of those sets which are better than others.

First, what criteria does a set of edges have to have in order to be legal at all? Our goal is to assign people into pairs. As a result, we have to make sure that we don't pick two edges that share any endpoints; if we did, then someone would be assigned into two different pairs, which we do not want to allow. Consequently,

we want to pick a set of edges such that no two of those edges share any endpoints. This gives rise to the following definition:

> A ***matching*** in an undirected graph $G = (V, E)$ is a set $M \subseteq E$ such that no two edges in $M$ share any endpoints.

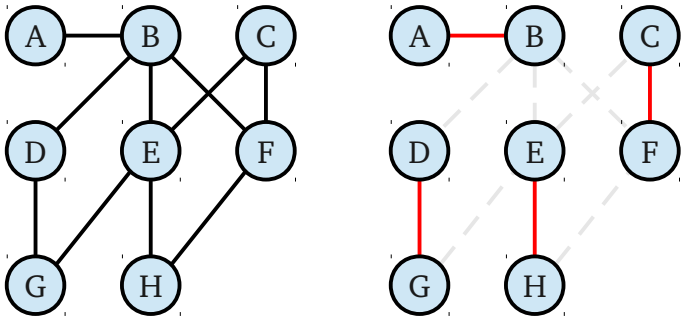
By this definition, the left and center choices of edges are matchings, while the third is not. Our goal in solving this problem will be to pick the "best" matching out of the graph, for some definition of "best."

Now that we have a definition of a matching, let's see if we can think about how to categorize matchings as "better" or "worse." One metric we could use to determine how good a matching is is the number of edges in the matching. A matching with a large number of edges pairs up a large number of people, while a matching with a small number of edges pairs up only a few people. By this criterion, the left matching above is a better matching than the center matching, since it matches more people.

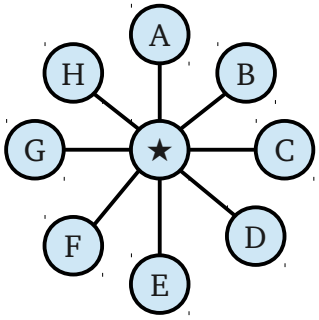
In the absolute best case, we would like a matching that pairs up everyone with someone else. That way, no one is left alone. For this reason, matchings of this sort are called *perfect matchings*:

> A ***perfect matching*** in an undirected graph $G = (V, E)$ is a matching $M$ in $G$ such that every node of $G$ is contained in some edge of $M$.

For example, here is a graph and a perfect matching in that graph:




When pairing people up, it would be ideal if we could find a perfect matching. However, it turns out that this is not always possible. For example, consider the following graph:




Here, it's not possible to find a perfect matching. No matter what edge we pick first as part of our matching, no other edges can be added, because that edge will have to share the starred node as an endpoint.

Given that we can't come up with a perfect matching in all graphs, we'll need to introduce a new piece of terminology that will let us describe the “best” matching in a given graph, even if it isn't an ideal matching. For this, we have the following definition:

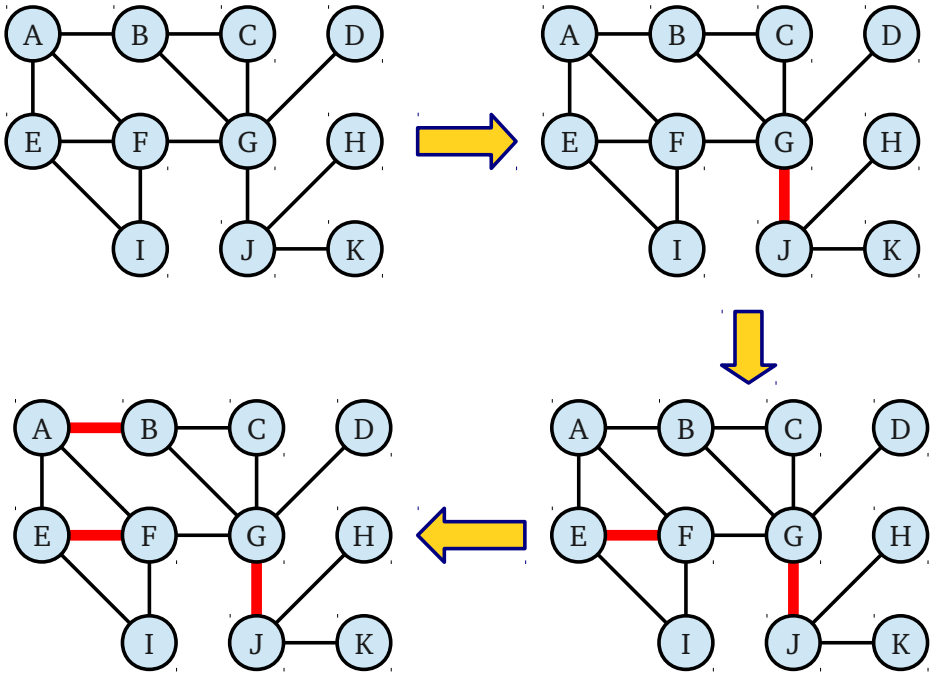
> A ***maximum matching*** in graph $G$ is a matching $M^*$ where for any matching $M$ in $G$, $|M^*| \geq |M|$.

In other words, a maximum matching is a matching that is at least as large as any other matching in the graph. There might be multiple different maximum matchings that all have the same size, but given a maximum matching, there is no other matching that is bigger than it.

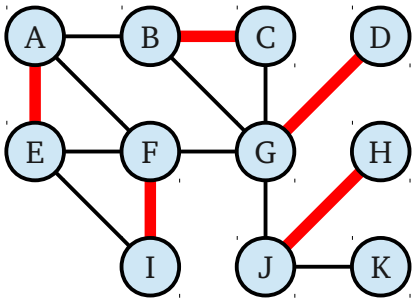
You might be wondering why we're denoting a maximum matching $M^*$, using a star. Often, when we describe the “best” object of some type, we denote it with a star. Here, $M^*$ indicates that this is a maximum matching, rather than some ordinary matching $M$. Later on, we will introduce some other maximum or minimum graph structures, which we'll denote with a *.

At this point, we're well on our way toward solving our original problem. We've formulated the idea of a matching, and have now come up with a definition that captures our goal: we want to find a maximum matching. While it helps that we have a mathematical specification of what we want, from a practical perspective we still haven't accomplished anything, since we don't have a sense of how exactly one might find a maximum matching. To do this, let's play around with matchings and see if we can come up with a decent algorithm.

To begin with, let's try doing something reasonably straightforward. What happens if we pick an edge from the graph at random, then pick another edge that doesn't share an endpoint with that edge, then pick another edge that doesn't share an endpoint with either of those edges, etc.? In other words, we'll try to grow up a maximum matching by iteratively adding more and more edges to our matching. For example, here is how we might try to build up a maximum matching one edge at a time:




Notice that, at the very end, we can't add any more edges into the matching. Does this mean that it's a maximum matching? Unfortunately, the answer is no. Here is a matching in the same graph with even more edges:




This initial attempt at an algorithm reveals something important about maximum matchings. If we have a set of edges that form a matching, and to which we can't add any more edges in the graph, we are *not* guaranteed that such a matching is a maximum matching. It might be a pretty good matching, but we can't be guaranteed that it's the best of all possible matchings.

To highlight the distinction between this kind of matching, where it's impossible to make any local improvements, and a true maximum matching, we introduce a new definition:

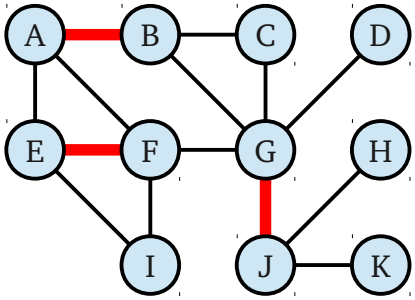
> A ***maximal matching*** is a matching $M$ in a graph $G = (V, E)$ is a matching such that for any edge $e \in E$, either $e \in M$, or $e$ shares an endpoint with some edge in $M$.

The terminology here is a bit subtle. A maxim**um** matching is a matching that is as large as is possible in the graph. We can't do any better than a maxim**um** matching. A maxim**al** matching is a matching that can't be grown by adding in any edges from the graph, but which might not necessarily be as large as is possible in the graph.

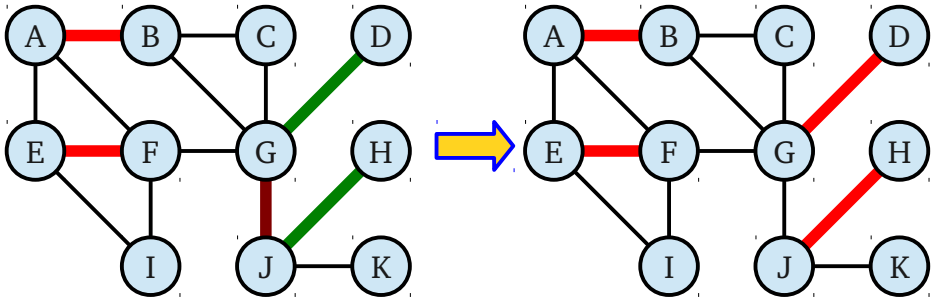
The distinction between maximum and maximal matchings tells us that it might to be tricky to find a maximum matching in a graph. Our naïve approach of iteratively adding in more and more edges is not guaranteed to find a maximum matching, so we will have to change our approach.

Although our initial guess didn't pan out, it doesn't mean that this approach is doomed to failure. The key intuition behind this algorithm – build up a matching by continuously increasing the number of edges in it until we can't do so any more – is a good one. We just need to think a bit harder about how we might grow a fledgling matching into a maximum matching.

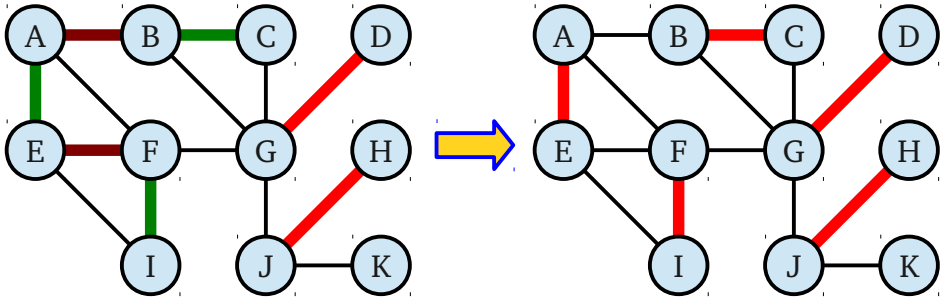
The reason that we can get stuck when growing a maximum matching by adding in edges is that in some cases, the only way to increase the size of a matching is to *remove* some edges as well. For example, here's the maximal matching that we came up with earlier:




We can make this matching larger by *removing* some edges, then adding in some new edges:




Notice that we have increased the total number of edges in the matching by one. This matching is still maxim**al** but not maxim**um**, but at least we've made some progress! Let's see if we can keep improving it. As before, we can grow the matching by removing some edges, and adding some new ones in:

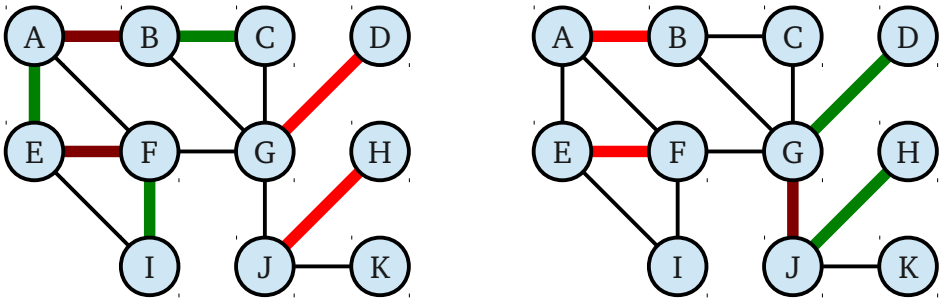



Once more we've increased the number of edges in the matching by one. This time, though, we actually have a maximum matching, and we're done.

The above discussion shows that for this particular example, it was possible to iteratively step up the number of edges in the matching one at a time until we ultimately arrived at a maximum matching. But how on

earth were we supposed to figure out what edges to add and remove at each point? And are we sure that it's always possible to step up the number of edges by one at each iteration?

To answer these questions, let's look carefully at what edges we removed and what edges we added. Below I've reprinted all of the intermediate matchings that we built up in the course of searching for a maximum matching, along with the edges that we removed (in red) and the edges that we added (in green):



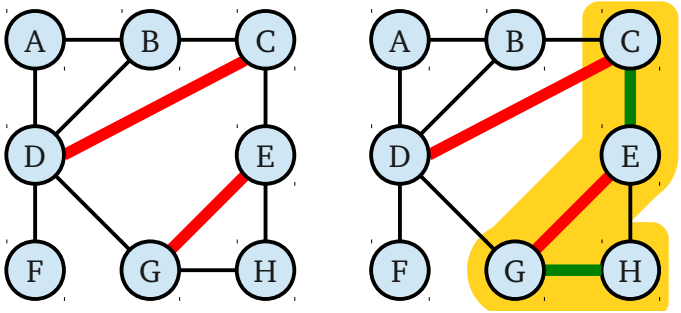

If you'll notice, there is something strange about these edges. Notice that in each case, the toggled edges form a path between two nodes. Moreover, those paths alternate between green (added) and red (removed) edges. Now *that's* odd. Is this purely a coincidence? Or is there something deeper going on here?

To answer this question, we will first need some terminology. First, let's give a name to paths like the above, which alternate between new and old edges:

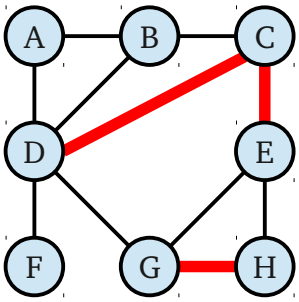
> Let $G = (V, E)$ be an undirected graph and $M$ be a matching in $G$. An ***alternating path*** in $G$ is a simple path $P = (v_1, v_2, \ldots, v_n)$ such that $(v_i, v_{i+1}) \in M$ iff $(v_{i+1}, v_{i+2}) \in E - M$. An ***alternating cycle*** is a simple cycle $C = (v_1, v_2, \ldots, v_n, v_1)$ such that $(v_i, v_{i+1}) \in M$ iff $(v_{i+1}, v_{i+2}) \in E - M$

In other words, an alternating path is a path that toggles back and forth between edges in the matching and edges not in the matching. All of the paths highlighted above are alternating paths.

In the above examples, we increased the size of a matching in the graph by finding some alternating path, adding in all of the edges that were not in the matching, and then removing all of the edges that were part of the original matching. Although in these cases it is possible to do this while leaving the resulting matching valid, in general we cannot take an arbitrary alternating path and update the matching by transforming it this way. For example, consider the following matching, which is maxim**al** but not maxim**um**. I've highlighted an alternating path in that graph:




If we transform the matching by toggling these edges, then we would get this new set of edges, which is not a matching:




If you'll notice, the reason that we can't update the matching by toggling this path is that the endpoints of the path were touching some edges in the matching that weren't part of the path. Consequently, when we toggled the path, we accidentally introduced new edges to the matching that touched these existing edges. Notice, however, that the alternating paths from the previous page, which were actually used to improve the size of the matching, had endpoints that weren't touching any edges in the matching. It seems like it's important to keep track of the endpoints of alternating paths if we're considering toggling them, since we want to make sure that toggling the path doesn't introduce edges that would touch other edges. This observation motivates the following definition:

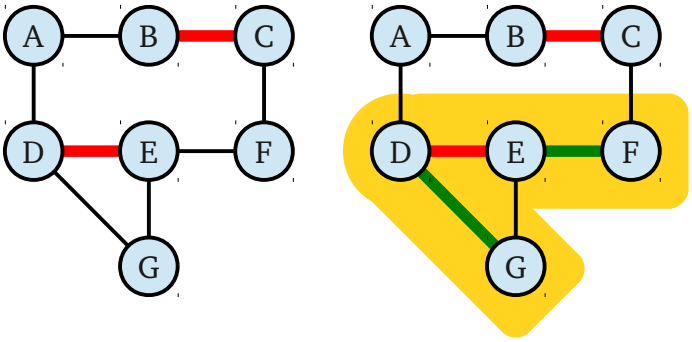
> Let $M$ be a matching in $G = (V, E)$. A node $v \in V$ is called ***matched*** iff there is some edge in $M$ that has it as an endpoint. A node $v \in V$ is called ***unmatched*** iff there is no edge in $M$ that has it as an endpoint.

Comparing the alternating paths that we used earlier to increase the size of a matching and the alternating path above, which ended up breaking the matching, we can see that the former all had endpoints that were unmatched, while the latter had endpoints that were matched.

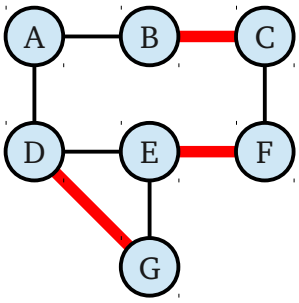
Based on this observation, we have one last definition to introduce:

> Let $M$ be a matching in $G$. An ***augmenting path*** in $G$ is an alternating path whose endpoints are unmatched.

For example, consider the following maxim**al** matching that is not maxim**um**. Notice that it contains an augmenting path, since it's an alternating path whose endpoints are unmatched:




If we toggle this augmenting path, we get an even larger matching:




This matching is a maximum matching, since with three edges, a total of six nodes are covered. If we were to add another edge, we would have to cover eight nodes, but there are only seven total in this graph.

Augmenting paths are the sorts of alternating paths that we want to find in a matching. If they are toggled, they definitely increase the size of the matching by one. However, the story does not end here. We can also show that if a matching is not maximum, then it *has* to contain an augmenting path. This is an important result in graph theory, and we formalize it below:

> ***Theorem (Berge):*** Let $M$ be a matching in $G$. Then $G$ has an augmenting path iff $M$ is not a maximum matching.

This is a powerful result. If we have a matching that we think might be a maximum matching, then we can check if it has no augmenting paths. If it doesn't, then we know that what we have is a maximum matching. Otherwise, we can find an augmenting path, toggle it, and end up with a larger matching. This gives us an algorithm for finding maximum matchings:

- Pick any matching.
- If it has no augmenting paths, report that it is a maximum matching.
- If it has an augmenting path, toggle that path and repeat.

It turns out that there is a clever algorithm for doing just that called the *blossom algorithm*, invented by Jack Edmonds in 1965. This algorithm, in a technical sense that we will define in Chapter 17, is efficient. The inner workings of the algorithm are somewhat tricky, but intuitively the algorithm works by growing a maximum matching up by searching for augmenting paths and then toggling them. As a result, it's possible to efficiently solve the maximum matching problem, and in fact many important algorithms (ranging from simple resource allocation tasks to more complex algorithms for finding paths through multiple cities) use maximum matching algorithms as a subroutine.

Of course, in order to convince ourselves that this algorithm works at all, we need to formally prove Berge's theorem. This proof is quite clever, though some of the details that arise when formalizing it are tricky. If you're interested how to prove Berge's theorem, see the next section. Otherwise, you can feel free to skip it and move on.

### 4.4.1 Proving Berge's Theorem

The proof of Berge's theorem will proceed in two parts. First, we will prove that if a matching has an augmenting path, then it is not maximum. Second, we will prove that if a matching is not maximum, then it has an augmenting path. This first proof is simpler, so we'll begin with it. Our goal will be to show the following:

***Theorem:*** Let $M$ be a matching in $G$. If $G$ has an augmenting path, then $M$ is not a maximum matching.

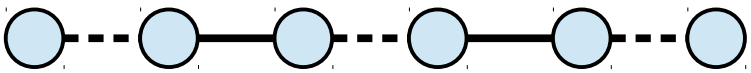
If you'll recall from before, we were able to increase the size of non-maximum matchings by taking an augmenting path and toggling all the edges in it; every edge on the path that was previously in the matching was removed, and every edge on the path that was previously not in the matching was added to the matching. Our proof of the above theorem will work by showing that given *any* augmenting path, performing this operation always yields a new matching with a larger number of edges than the original matching.

To do this, we will need to prove two key lemmas:

***Lemma 1:*** Let $M$ be a matching in $G$. If $P$ is an augmenting path in $G$, then $M \Delta P$ is a matching in $G$.

***Lemma 2:*** Let $M$ be a matching in $G$. If $P$ is an augmenting path in $G$, then $|M \Delta P| > |M|$.

Before we discuss the implications of this lemma, let's quickly review what on earth this $\Delta$ symbol means. If you'll recall from Chapter 1, $\Delta$ is the *symmetric difference* operator. Given two sets, it produces the set of all elements in exactly one of the two sets. To see why this corresponds to our "toggling" operation, let's think about what an augmenting path looks like. From the perspective of the matching, an augmenting path looks like this:



Here, the dotted edges represent edges not in the matching, and the solid edges represent edges within the matching. Now, consider what happens when we take the symmetric difference of the set of all the edges along the path and the set of edges in the matching. Each edge of the graph is either

- Not on this path and not in the matching, in which case it is not in the symmetric difference,
- Not on this path but in the matching, in which case it is in the symmetric difference.
- On the path and in the matching, in which case it is not in the symmetric difference because it is both in the matching and on the path,
- On the path but not in the matching, in which case it is in the symmetric difference.

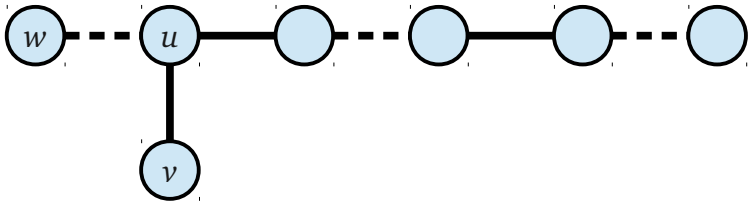
In other words, the only edges in the symmetric difference are the edges that are either (1) originally in the matching but not on the augmenting path, or (2) on the augmenting path but not in the original matching. Consequently, the notation $M \Delta P$ does indeed capture what we mean by "toggling" an augmenting path.

Now that we have a slightly better understanding of the notation, let's see how we might go about proving that these lemmas are true. Let's focus first on Lemma 1, which says that if we take $M \Delta P$, where $P$ is an augmenting path, then we end up with a set that is a legal matching. To do this, we need to show that no pair of edges in the resulting graph have any endpoints in common. An equivalent way of thinking about this is to show that no node in the graph is an endpoint of two different edges from $M \Delta P$. It's this line of reasoning that we'll pursue.

To show that no node has two edges touching it, we'll proceed by contradiction and suppose that there is some node $u$ that is incident to two different edges; let's call them $\{u, v\}$ and $\{u, w\}$. So let's think about

where these edges are from. Since they're contained in $M \Delta P$, we know that each edge is either in $M$ but not $P$ or $P$ but not $M$. This gives rise to three possibilities:

1. Both $\{u, v\}$ and $\{u, w\}$ are in $M$, but not $P$. This is impossible, since it would mean that two edges in the matching $M$ share an endpoint.
2. Both $\{u, v\}$ and $\{u, w\}$ are in $P$ but not $M$. Since $P$ is an alternating path, this means that somewhere in the path is the sequence $(v, u, w)$. But since $P$ is an alternating path, this means that at least one of $\{v, u\}$ and $\{u, w\}$ has to be in $M$, since the edges of $P$ alternate between edges of $M$ and edges not in $M$. This contradicts the initial assumption that the edges are in $P$ but not $M$.
3. One of $\{u, v\}$ and $\{u, w\}$ is in $M$ but not $P$ and the other is in $P$ but not $M$. Let's assume for simplicity that $\{u, v\}$ is in $M$ but not $P$, and that $\{u, w\}$ is in $P$ but not $M$. Graphically, this would mean that we have a setup like this:




Something has to be wrong here. Since $\{u, v\}$ is in $M$, we know that $u$ must be a matched node. Consequently, it can't be the endpoint of the path $P$. This means that there must be a node to the left and to the right of it on the path. Because the path is an alternating path, one of those edges must be in the matching. It's not $\{u, w\}$, so it must be the other edge. But then there are two different edges incident to $u$ in the original matching, which is impossible.

Formalizing this argument pretty much means translating the above argument into more rigorous terms, which we'll do here:

***Lemma 1:*** Let $M$ be a matching in $G$. If $P$ is an augmenting path in $G$, then $M \Delta P$ is a matching in $G$.

***Proof:*** By contradiction; assume that $M$ is a matching in $G$, $P$ is an augmenting path in $G$, but that $M \Delta P$ is not a matching in $G$. This means that there must be some node $u$ such that $u$ is incident to two different edges $\{u, v\}, \{u, w\} \in M \Delta P$.

Since $\{u, v\}, \{u, w\} \in M \Delta P$, we have that each edge either is in $M - P$, or it is in $P - M$. We consider three cases about which edges are in which sets.

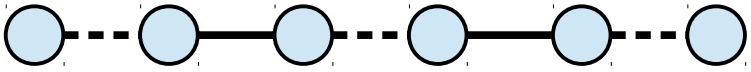
*Case 1:* $\{u, v\}, \{u, w\} \in M - P$. Since $M - P \subseteq M$, this means that $\{u, v\}, \{u, w\} \in M$. But this is impossible, since $M$ is a matching and the two edges share an endpoint.

*Case 2:* $\{u, v\}, \{u, w\} \in P - M$. Since $P$ is a simple path, this means that the sequence $(u, v, w)$ or $(w, v, u)$ must occur in $P$. But since $P$ is an alternating path, this means that either $\{u, v\} \in M$ or $\{u, w\} \in M$, which is impossible because $\{u, v\}, \{u, w\} \in P - M$.

*Case 3:* Exactly one of $\{u, v\}$ and $\{u, w\}$ belongs to $P - M$ and one belongs to $M - P$. Assume without loss of generality that $\{u, v\} \in M - P$ and $\{u, w\} \in P - M$. Since $P$ is an augmenting path, its endpoints are unmatched. Thus $u$ cannot be an endpoint of $P$. Consequently, there must be exactly two edges along path $P$ incident to $u$. Since $P$ is an alternating path, one of these edges (call it $\{u, x\}$) must be in $M$. Since $\{u, v\} \notin P$, this means that $x \neq v$. But then this means that $\{u, x\} \in M$ and $\{u, v\} \in M$, which is impossible since $M$ is a matching.

In all three cases we reach a contradiction, so our assumption must have been wrong. Thus $M \Delta P$ is a matching in $G$. ■

Great! We've established that if $P$ is an augmenting path, then $M \Delta P$ is also a matching in $G$. All that remains to do is to prove Lemma 2, which states that $|M \Delta P| > |M|$, meaning that the size of the matching has increased. Intuitively, this is true because if we take any augmenting path, like this one here:



The number of edges from $M$ is one less than the number of edges not in $M$. Consequently, when we toggle the edges along the path, the number of edges in $M$ increases by one. The actual details of this proof are left to you as one of the chapter exercises.

Given these two lemmas, we have the following result:

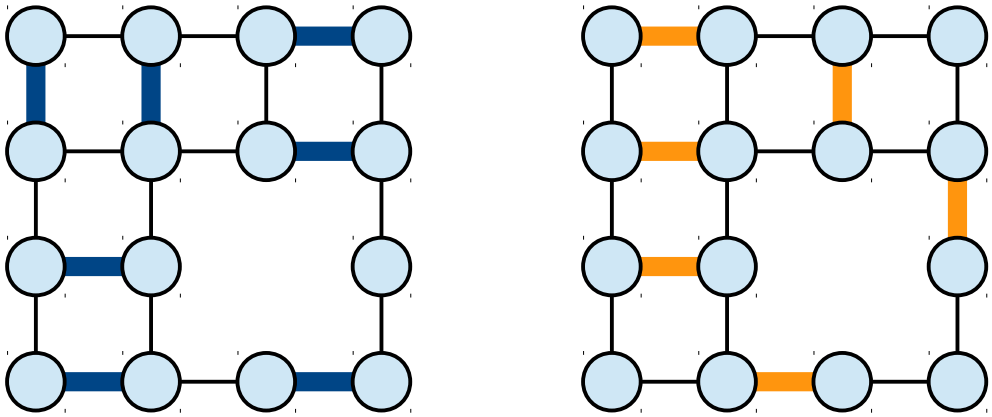
***Theorem:*** Let $M$ be a matching in $G$. If $G$ has an augmenting path, then $M$ is not a maximum matching.

***Proof:*** Let $M$ be a matching in $G$, and let $P$ be an augmenting path. Then by Lemma 1, $M \Delta P$ is a matching in $G$, and by Lemma 2, $|M \Delta P| > |M|$. Thus $M$ is not a maximum matching. ■

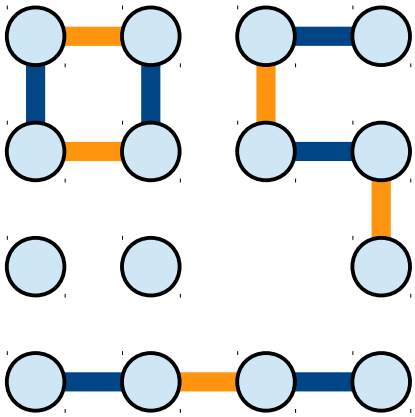
We're now halfway there. We've formally proven that if you have an augmenting path, it's possible to increase the size of a non-maximum matching. But now we have to prove the converse, namely, that if we have a non-maximum matching, we can always find an augmenting path. This is a beautiful proof, but it will require us to adopt a proof technique that we previously have not seen before.

The key idea behind the proof will be the following. Suppose we have an arbitrary matching $M$ that, for whatever reason, we know is not maximum. We know that there must exist at least one maximum matching $M^*$. We don't necessarily know anything about (how many edges it has, which edges it contains, etc.), but mathematically speaking it's fine to reason about this matching $M^*$. Given that our matching isn't maximum, we know that there must be more edges in $M^*$ than in $M$. What would happen if we were to consider the "difference" between $M$ and $M^*$? At first this might seem like a strange idea – how can we reason about the difference between our current matching and a matching we've never seen before? – but this sort of inquiry is actually quite common in graph theory.

To give a concrete example of this, consider the following graph, along with two matchings in it:



The matching $M^*$ on the left is a maximum matching, while the matching $M^*$ on the right is maximal but not maximum. Normally we wouldn't have the luxury of actually having a maximum matching lying around, but to motivate the key idea here let's suppose that we do. Now, let's see what happens if we consider the difference between these two matchings. Specifically, we'll consider the set $M \Delta M^*$. This set corresponds to the edges that are unique to one of the two matchings. In our case, we get the following:



Notice that this set of edges isn't a matching – there are many nodes that have two edges incident to them – but at the same time this isn't a totally random set of edges. In particular, notice that every node has at most two edges incident to it, possibly a yellow edge that comes from $M$, and possibly a blue one that comes from $M^*$. We can't have two edges from the same matching touching a given node, since otherwise what we started with wasn't a matching.

The fact that every node has at most two edges connected to it strongly constrains what structures can arise in the symmetric difference $M \Delta M^*$. As you can see from above, the graph will consist of several connected components, where each connected component is either an isolated vertex with no edges, a simple path, or a simple cycle. Any other structure would require some node to have more than two edges adjacent to it. This is formalized with this lemma:

> ***Lemma 3:*** Let $M$ and $N$ be matchings in $G = (V, E)$. Then each connected component of the graph $(V, M \Delta N)$ is either an isolated vertex, a simple path, or a simple cycle.

The proof of this result is actually quite interesting, and is left as an exercise at the end of the chapter. If it seems like I'm getting lazy and just asking you to do all the work, I swear that that's not what I'm doing. I just want to keep the flow going through this proof so that you can see the beauty of the ultimate result. Honest.

We're getting close to showing how to find an augmenting path in our non-maximum matching $M$. By taking the symmetric difference of the matching and a true maximum matching, we end up with a bunch connected components, each of which is either an isolated node, a simple cycle, or a simple path. We're searching for an augmenting path in $M$, so for now let's focus on the connected components that are simple paths.

Let's suppose that we find a connected component in $M \Delta M^*$ that is a simple path $P$. What can we say about the edges in $P$? Well, every edge in this path is contained in $M \Delta M^*$, so each edge in $P$ belongs to either $M$ or $M^*$, but not both. Since we know that $M$ and $M^*$ are matchings, it's not possible for two consecutive edges $P$ to belong to either $M$ or $M^*$, because then there would be two edges in a matching that share an eYou shouldn't need to construct any type objects for this part of the assignment; the parser should handle that logic. I would suggest seeing if you really want to construct a new ArrayType, or whether you can recycle an old one.ndpoint. In other words, any connected component in $M \Delta M^*$ that is a simple path must also be an alternating path.

This motivates the following lemma:

> ***Lemma 4:*** Let $M$ and $N$ be matchings in $G = (V, E)$ Then any simple path or cycle in $(V, M \Delta N)$ is alternating.

> ***Proof:*** Let $M$ and $N$ be matchings in $G$ and consider the graph $G' = (V, M \Delta N)$. We will show that any simple path or cycle in $G$ must be alternating.
>
> Consider any simple path or cycle $P$ in $G$. To show that $P$ is alternating, suppose for the sake of contradiction that it is not and that there are two adjacent edges $\{u, v\}$ and $\{v, w\}$ in $P$ that are either both contained in $M$ or both contained in $N$. But that means that one of the two matchings contains two edges with the same endpoint, a contradiction. We have reached a contradiction, so our assumption must have been wrong and $P$ must be alternating. ■

Since these paths are *alternating* paths, perhaps one of them might be an *augmenting* path. Remember that an alternating path is an augmenting path if its endpoints are unmatched. Not all the alternating paths we find this way are necessarily going to be augmenting (look at the above example for a concrete instance of this). However, we can make the following observation. Suppose that we find an alternating path this way whose first and last edges are in $M^*$ but not $M$. In this case, look at the very first and last nodes of this path. Are these nodes matched or unmatched in $M$?

It turns out that these endpoint nodes have to be unmatched. To see why this is, we'll proceed by contradiction. Suppose that one of these endpoint nodes is indeed matched in $M$. Let's call this node $u$. Since $u$ is the endpoint of an alternating path that ends with a node from $M^*$, there must be some node $v$ such that $\{u, v\} \in M^*$. Now, since $u$ is allegedly matched in $M$, there must be some edge $\{u, x\}$ in the original matching $M$ that is incident to $u$. Now, is this edge contained in $M \Delta M^*$? If it is, then $u$ really isn't the endpoint of the alternating path, since we could extend that path by following $\{u, x\}$, meaning that the alternating path we were considering wasn't an entire connected component like we said it was. This is shown below:


If the edge *isn't* in $M \Delta M^*$, then it means that $\{u, x\}$ must also be contained in $M^*$. But that is impossible, since $\{u, v\} \in M^*$, meaning that two edges in $M^*$ ($\{u, x\}$ and $\{u, v\}$) share an endpoint. In either case, we have a contradiction.

We can formalize this reasoning below:

***Lemma 5:*** Let $M$ be a matching in $G$ and $M^*$ be a maximum matching in $G$. Suppose that $P$ is a connected component of $M \Delta M^*$ that is a simple path. If the first and last edges of $P$ are contained in $M^*$, then $P$ is an augmenting path.

***Proof:*** Let $M$ be a matching of $G$ and $M^*$ a maximum matching in $G$, and let $P$ be a connected component of $M \Delta M^*$ that is a simple path, such that the first and last edges of $P$ are contained in $M^*$. By Lemma 4, we know that $P$ is alternating. To show that $P$ is an augmenting path, we need to show that its first and last nodes are unmatched in $M$.

We proceed by contradiction and assume that some endpoint $u$ of $P$ is matched by $M$; assume without loss of generality that it is the start of path $P$. Since the first edge of $P$ is contained in $M^*$, this edge must have the form $\{u, v\}$ for some node $v$. Because $u$ is matched in $M$, there must be some edge $\{u, x\} \in M$. We consider two cases:

*Case 1:* $\{u, x\} \in M \Delta M^*$. Because $P$ is a simple path, and $u$ is the start of the path, this must mean that the first edge of the path is $\{u, x\}$. Since we already know the first edge of the path is $\{u, v\}$, this means that $v = x$. Consequently, $\{u, x\} = \{u, v\} \in M^*$. But this means that $\{u, x\} \in M$ and $\{u, x\} \in M^*$, so $\{u, x\} \notin M \Delta M^*$, contradicting our initial assumption.

*Case 2:* $\{u, x\} \notin M \Delta M^*$. Since $\{u, x\} \in M$, this means that $\{u, x\} \in M^*$ as well. Now, if $x \neq v$, then this means that $\{u, x\}$ and $\{u, v\}$ are two distinct edges in $M^*$ that share an endpoint, a contradiction. So it must be that $x = v$, so $\{u, x\}$ and $\{u, v\}$ are the same edge. But this is impossible, since $\{u, v\} = \{u, x\}$ is the first edge of a path in $M \Delta M^*$, contradicting the fact that $\{u, v\} = \{u, x\} \notin M \Delta M^*$.

In either case we reach a contradiction, so our assumption must have been wrong. Thus $P$ must be an augmenting path, as required. ■

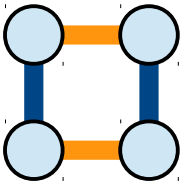
This result guarantees us that if we look at the connected components of $M \Delta M^*$ and find an alternating path whose endpoints are in $M^*$, we have found an augmenting path in the original matching $M$. This is great news for us – we're getting very close to showing that if $M$ is not maximum, then an augmenting path must exist. However, this celebration might prove premature. How are we supposed to show that an alternating path of this sort must exist?

To do this, we will employ a technique called a *counting argument*, a type of reasoning we will explore in more detail in a later chapter. The basic idea behind a counting argument is to use the fact that one set is larger than another to guarantee that some object must exist. In our case, we will use the fact that $M^*$, a maximum matching, is larger than $M$, a non-maximum matching. This means that there are more edges in $M^*$ than in $M$. We will use this fact to show that there has to be an alternating path in $M \Delta M^*$ that starts and ends with edges from $M^*$, simply because the surplus edges from $M^*$ have to go somewhere.

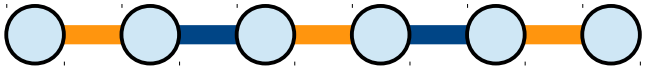
At a high level, our argument will proceed as follows. We'll begin by splitting the edges from $M$ and $M^*$ into two groups – first, the group of edges that they have in common ($M \cap M^*$), and second, the group of edges where they differ ($M \Delta M^*$). The number of edges of $M$ contained within $M \cap M^*$ is the same as the number of edges from $M^*$ contained within $M \cap M^*$, because this intersection represents the elements that they have in common. Consequently, since we know that there are more edges in $M^*$ than in $M$, this means that there must be more edges from $M^*$ in $M \Delta M^*$ than there are edges from $M$ in $M \Delta M^*$.

Given that $M^*$'s edges outnumber $M$'s edges in $M \Delta M^*$, let's take a second look at the structure of $M \Delta M^*$. As mentioned above, this graph consists of several connected components, each of which is either an isolated vertex, an alternating path, or an alternating cycle. Let's look at each of these possibilities, counting up how many edges from $M$ and $M^*$ are represented in each.

- An isolated vertex has no edges at all.
- An alternating cycle must have the same number of edges from $M$ and $M^*$, as seen here:



- An alternating path starting and ending with edges from $M$:



Or starting and ending with an edge from $M$ and an edge from $M^*$ (or vice-versa):

Or starting and ending with edges from $M^*$:

Of these five possibilities, note that in the case of the first four, the number of edges from $M^*$ in the connected component is less than or equal to the number of edges from $M$. Imagine what would happen if all the connected components of $M \Delta M^*$ were of one of these first four types. In that case, if we added up the total number of edges in $M \Delta M^*$, we would find that the number of edges from $M^*$ was no greater than the number of edges from $M$. But that's impossible, since there are supposed to be more edges from $M^*$ here than from $M$. As a consequence, there *has* to be an alternating path of the last type, namely, an augmenting path.

This line of reasoning is called a ***nonconstructive proof*** because it asserts the existence of some object (namely, an augmenting path) without actually showing how to find it. Our reasoning says that such a path has to exist, simply because it's impossible not to. If you come from an algorithmic background, this may be troubling. We've given no indication about how we're actually supposed to produce the augmenting path. We'll return to that question later in the chapter.

To formalize the previous line of reasoning and complete the proof, we need to prove a few final lemmas. First, let's formally demonstrate that there are more edges from $M^*$ in $M \Delta M^*$ than there are edges from $M$ in $M \Delta M^*$:

***Lemma 6:*** Let $M$ be a non-maximum matching and $M^*$ be a maximum matching. Then the number of edges from $M^*$ in $M \Delta M^*$ is greater than the number of edges from $M$ in $M \Delta M^*$. Formally, $|M^* \cap (M \Delta M^*)| > |M \cap (M \Delta M^*)|$

***Proof:*** Since $M^*$ is a maximum matching and $M$ is a non-maximum matching, we know that $|M^*| > |M|$. Note that every edge of $M^*$ and of $M$ either belongs to both $M$ and $M^*$, or it belongs to exactly one of $M$ and and $M^*$. That is, every edge of $M$ and $M^*$ either belongs to $M \cap M^*$ or to $M \Delta M^*$, but not both.

Consequently, we can write $M^* = (M \cap M^*) \cup (M^* \cap (M \Delta M^*))$; that is, it is the union of the edges in $M \cap M^*$, along with just the edges of $M \Delta M^*$ that are also in $M^*$. Similarly, we can write $M = (M \cap M^*) \cup (M \cap (M \Delta M^*))$. Moreover, notice that the sets $M \cap M^*$, $M^* \cap (M \Delta M^*)$, and $M \cap (M \Delta M^*)$ are all disjoint. Therefore, we have that

$$|M^*| = |(M \cap M^*) \cup (M^* \cap (M \Delta M^*))| = |M \cap M^*| + |M^* \cap (M \Delta M^*)|$$
$$|M| = |(M \cap M^*) \cup (M \cap (M \Delta M^*))| = |M \cap M^*| + |M \cap (M \Delta M^*)|$$

Since $|M^*| > |M|$, this means that

$$|M \cap M^*| + |M^* \cap (M \Delta M^*)| > |M \cap M^*| + |M \cap (M \Delta M^*)|$$

so

$$|M^* \cap (M \Delta M^*)| > |M \cap (M \Delta M^*)|$$

as required. ■

Next, let's formalize that the first four connected components all have at most the same number of edges from $M^*$ as from $M$:

***Lemma 7:*** Let $M$ and $M^*$ be a non-maximum matching and maximum matching, respectively, of a graph $G = (V, E)$. Consider any connected component of $(V, M \Delta M^*)$ that is not an alternating path whose first and last edges are in $M^*$. Then this connected component contains no more edges from $M^*$ than from $M$.

***Proof:*** Let $M$ and $M^*$ be a non-maximum matching and maximum matching, respectively, of a graph $G = (V, E)$. By Lemma 3, every connected component of $(V, M \Delta M^*)$ is either an isolated vertex, a simple path, or a simple cycle. So consider any connected component $C$ of $(V, M \Delta M^*)$ that is not a simple path that begins and ends with edges from $M^*$. We consider three cases:

*Case 1:* $C$ is an isolated vertex. Then $C$ has no edges, so the number of edges from $M^*$ and $M$ are both 0 and the claim holds.

*Case 2:* $C$ is a simple cycle; call it $(v_1, v_2, \ldots, v_n, v_1)$. By Lemma 4, this cycle is an alternating cycle. We therefore claim that $n$ is even. The proof is by contradiction; assume that $n$ is odd. Since the edges of $C$ alternate, all edges of the form $(v_{2k+1}, v_{2k+2})$ must be in the same set as $(v_1, v_2)$. But if $n$ is odd, this means that the very last edge $(v_n, v_1)$ must be in the same set as $(v_1, v_2)$. This means that one of the sets $M$ or $M^*$ must have two edges in it that share an endpoint, namely, $v_1$. We have reached a contradiction, so our assumption must have been wrong. Thus if $C$ is a simple cycle, it has even length. Accordingly, since the edges alternate between $M$ and $M^*$, half of these edges belong to $M$ and half belong to $M^*$, so the number of edges from $M$ and $M^*$ are the same.

*Case 3:* $C$ is a simple path. By Lemma 4, this is an alternating path. If $C$ has even length, then half of the edges belong to $M$ and half to $M^*$, so the number of edges from each matching in $C$ is the same. Otherwise, if $C$ has odd length, then we can split $C$ into two smaller paths – an even-length alternating path of all but the last edge (in which the number of edges from $M$ and $M^*$ are equal), plus a single edge. Since the path alternates, this single edge must be from the same set as the first edge of the path. Since (by our initial assumption) the first and last edge of this path are not contained in $M^*$, this means that they must be from $M$. Thus there is one more edge from $M$ than from $M^*$.

In all three cases there are at least as many edges from $M$ as from $M^*$, as required. ■

We're almost done! Given these lemmas, it's possible for us to formally show that if $M$ is not maximum matching, then there is an augmenting path. That proof, which uses all of the previous lemmas, is a proof by contradiction. Suppose that there is no augmenting path in $M \Delta M^*$. In that case, if we add up the total number of edges of all the connected components in $M \Delta M^*$, we will find that the number of edges from $M^*$ can't be bigger than the number of edges from $M$, since (by the previous lemma) each individual connected component has at least as many edges from $M^*$ as from $M$. This contradicts our earlier lemma, which says that the number of edges from $M^*$ in $M \Delta M^*$ has to be bigger than the number of edges from $M$ in $M \Delta M^*$. The only possible option, therefore, is that some connected component of $M \Delta M^*$ has to be an augmenting path.

To formalize this reasoning, we proceed by induction over the number of connected components in $M \Delta M^*$. This is shown here:

***Theorem:*** Let $M$ be a matching in $G$. If $M$ is not a maximum matching, then $G$ has an augmenting path.

***Proof:*** Let $M$ be a non-maximum matching in $G = (V, E)$. Consider any maximum matching $M^*$ and the graph $G' = (V, M \Delta M^*)$. Assume for the sake of contradiction that $G'$ does not contain a connected component that is an alternating path whose first and last edges are in $M^*$. Now consider the total number of edges from $M$ and from $M^*$ in $G'$; that is, the cardinalities of $M \cap (M \Delta M^*)$ and $M^* \cap (M \Delta M^*)$. Each edge of both of those sets must belong to some connected component of $G'$. By Lemma 7, we know that each connected component of $G'$ must have no more edges from $M^*$ than edges from $M$. Summing up the number of edges from $M$ and $M^*$ across all these connected components, we get that $|M \cap (M \Delta M^*)| \geq |M^* \cap (M \Delta M^*)|$. But this contradicts Lemma 6, which states that $|M \cap (M \Delta M^*)| < |M^* \cap (M \Delta M^*)|$. We have reached a contradiction, so our assumption must have been wrong. Thus $G'$ contains at least one connected component that is an alternating path whose first and last edges are in $M^*$. By Lemma 5, such a path must be an augmenting path in $G$, as required. ■

Whew! This is, by far, the trickiest proof we've completed so far. We needed seven lemmas in order to establish the key result.

Despite its complexity, this proof is important because it shows off several important proof techniques. We saw how to compare a non-maximum matching to a maximum matching, even if we haven't seen it, in order to find an augmenting path. In doing so, we also saw our first counting argument, which we used to guarantee the existence of the augmenting path. We will see these techniques employed in many other contexts later on.

## 4.5 Chapter Summary

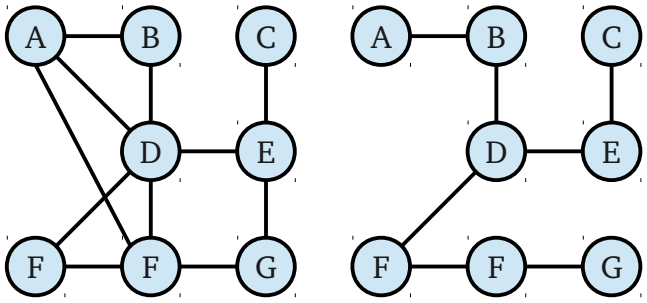
- An *unordered pair* is a collection of two values with no ordering. An *ordered pair* is a collection of two values, a first value and a second value.
- A *graph* is a collection of *nodes* joined by *edges*.
- A graph is *directed* if its edges are ordered pairs. A graph is *undirected* if its edges are unordered pairs.
- A *path* in a graph is a series of nodes where each node has an edge to the next node in the path. A *simple path* is a path with no duplicated nodes.
- A *cycle* in a graph is a path from a node back to itself. A *simple cycle* in a graph is a cycle with no duplicated edges and no duplicated nodes (except for the first and last).
- Two nodes in an undirected graph are *connected* when there is a path between them. An undirected graph as a whole is called *connected* if all pairs of nodes in the graph are connected.
- A *connected component* in an undirected graph is a maximal set of connected nodes in the graph. Every node in a graph belongs to exactly one connected component.
- An undirected graph is *2-edge-connected* when it is connected and remains connected even if any single edge is deleted. 2-edge-connected graphs are precisely the graphs where each edge lies on a simple cycle.
- A *tree* is a minimally-connected graph. Equivalently, it is a maximally acyclic graph, or a connected graph that is a cyclic, or a graph where there is exactly one simple path between any pair of nodes.
- The *degree* of a node in an undirected graph is the number of edges incident to it.
- A *leaf node* in a tree is a node with degree at most 1. All other nodes in a tree are called *internal nodes*. Every tree has at least two leaves, except for trees with just one node.
- Removing any edge from a tree leaves two connected components that themselves are trees. This makes it possible to use proof by induction on trees by inducting on the number of nodes or edges.
- In a directed graph, one node is *reachable* from another if there is a path from the first node to the second. Two nodes are *strongly connected* if they are mutually reachable from one another.
- A *strongly connected component* (SCC) of a graph is a maximal set of strongly connected nodes. Every node in a directed graph belongs to exactly one strongly connected component.
- A *directed acyclic graph* (DAG) is a directed graph with no cycles.
- A *topological ordering* of a directed graph is an ordering of the nodes such that no node in the ordering has an edge to any previous node in the ordering. The graphs that can be topologically ordered are precisely the DAGs.
- A *source* in a DAG is a node with no incoming edges. A *sink* in a DAG is a node with no outgoing edges.
- The *condensation* of a graph is the graph formed by contracting all strongly connected components together into individual nodes. Such a graph always forms a DAG.
- A *matching* in a graph is a set of edges such that no two edges have any endpoints in common.

- A *maximal matching* is a matching that cannot be made larger by adding any single edge. A *maximum matching* is a matching for which no larger matching exists in the same graph.
- An *augmenting path* in a matching is a path between unmatched nodes such that the edges alternate between matched and unmatched.
- *Berge's theorem* states that a matching is maximum iff there are no augmenting paths.

## 4.6 Chapter Exercises

1. We defined a simple path as a path with no repeated nodes. Is this the same as defining a simple path as a path with no repeated *edges*? If so, prove it. If not, give an example of a simple path with repeated edges or of a path with no repeated edges that is not a simple path.
2. When defining an unordered pair, we noted that an unordered pair with just one element $a$ would be represented by the set $\{a, a\}$. Since sets do not allow duplicates, this is equal to the set $\{a\}$. However, from context, we can realize that this set should be interpreted as the set $\{a, a\}$, since we expect to find two elements.

   Can we define an unordered triple as a set of three elements $\{a, b, c\}$, if it's possible for there to be duplicates?
3. Let $G$ be an undirected graph. Prove that if $u$ and $v$ are nodes in $G$ where $u \leftrightarrow v$, then there is a simple path between $u$ and $v$.
4. Let $G$ be an undirected graph. Suppose that there is a simple path from $u$ to $v$ and a simple path from $v$ to $x$. Does this necessarily mean that there is a simple path from $u$ to $x$? If so, prove it. If not, give a counterexample.
5. A graph is called *k-connected* iff it is connected, and there is no set of fewer than $k$ *nodes* that can be removed from the graph without disconnecting the graph.
   1. Prove that any 1-edge-connected graph is also 1-connected.
   2. Prove that there exists a 2-edge-connected graph that is *not* 2-connected.
   3. Prove that any 2-connected graph is 2-edge-connected.
6. Prove that in any tree $T = (V, E)$ that for any node $v \in V$, the edges of $T$ can be assigned a direction such that there is a path from every node $u \in V$ to $v$. If this is done, the node $v$ is called a ***root node*** of the directed tree.
7. If $G = (V, E)$ is a connected, undirected graph, a ***spanning tree*** of $G$ is a tree $T = (V, E')$ with the same nodes as $G$, but whose edges are a subset of the edges of $G$. For example, below is a graph and one of its spanning trees:




Prove that every connected graph has a spanning tree.

8. Prove that a graph has exactly one spanning tree iff it is a tree.
9. Suppose that we augment each of the edges in a graph by assigning a weight to each of them. In that case, a ***minimum spanning tree*** is a spanning tree of the graph where the total weight of the edges in the spanning tree is less than or equal to the total weight of the edges in any spanning tree.
    1. Suppose that there is an edge $e$ in a graph such that for every simple cycle $C$ that contains $e$, $e$ is the heaviest edge on that cycle. Prove that in this case, $e$ cannot be a part of any minimum spanning tree.
    2. Suppose that all edge weights in a graph $G$ are distinct. Prove that in this case, the heaviest non-bridge edge of $G$ cannot be part of any minimum spanning tree.

    Your result from (2) gives a simple and reasonably efficient algorithm for finding minimum spanning trees. Scan across the edges of $G$ in descending order of weight. For each edge visited, if it is not a bridge, remove it. The result will be a minimum spanning tree.
10. Prove that if a graph $G = (V, E)$ is a tree iff $|V| = |E| + 1$ and $G$ is connected.
11. Trees are minimally-connected graphs; removing any one edge will disconnect them.
    1. What 2-edge-connected graphs have the property that removing any two edges is guaranteed to disconnect the graph? That is, what graphs are connected, stay connected after any edge is disconnected, but are disconnected after any two edges are removed?
    2. What 3-edge-connected graphs have the property that removing any three edges is guaranteed to disconnect the graph?
12. Prove that an undirected graph $G = (V, E)$ is 2-edge-connected if, for any pair of nodes, there are two paths $P_1$ and $P_2$ between those nodes that have no edges in common.
13. Prove that if an undirected graph $G = (V, E)$ is 2-edge-connected, then for any pair of nodes, there are two paths $P_1$ and $P_2$ between those nodes that have no edges in common. ★
14. What's so "strong" about strong connectivity? This definition contrasts with another definition called ***weak connectivity***. A directed graph $G$ is called *weakly connected* if the graph formed by replacing each directed edge with an undirected edge is a connected graph.
    1. Prove that if $G$ is strongly connected, then $G$ is weakly connected.
    2. Prove that if $G$ is weakly connected, it is not necessarily strongly connected.
15. Let $G = (V, E)$ be an undirected graph. The ***complement of a graph***, denoted $G^C$, is the graph with the same nodes as $G$, but where $\{u, v\}$ is an edge in $G^C$ iff $\{u, v\}$ is *not* an edge in $G^C$. Prove that for any graph $G$, that at least one of $G$ or $G^C$ must be connected.

16. Let $G = (V, E)$ be a directed graph. The *reverse of* $G$, denoted $G^{rev}$, is the graph $G^{rev} = (V, E')$, where $E'$ is the set $\{ (v, u) \mid (u, v) \in E \}$ with the same nodes as $G$, but with all the edges reversed.

    Prove that $G^{rev}$ is a DAG iff $G$ is a DAG.

17. Using your result from the previous problem, prove that every DAG with at least one node has at least one sink.

18. Prove that the edges in any undirected graph $G$ can be assigned a direction such that $G$ becomes a DAG.

19. What is the maximum number of strongly connected components in a DAG with $n$ nodes? What is the minimum number?

20. Let $G = (V, E)$ be a strongly connected directed graph. Prove that the undirected graph $G'$ formed by replacing each directed edge with an undirected edge is 2-edge-connected.

21. Prove that if an undirected graph $G = (V, E)$ is 2-edge-connected, then there is a way of assigning the edges of $E$ a directionality so that the resulting graph is strongly-connected. This result is called ***Robbins' Theorem***. ★

22. Describe a graph with 1,000 edges whose maximum matching has size 500.

23. Describe a graph with 1,000 edges whose maximum matching has size 1.

24. Although maxim**al** matchings are not necessarily maxim**um** matchings, we can say that the size of any maximal matching isn't too far off from the size of any maximum matching. Prove that if $M$ is a maximum matching and $M^*$ is a maximal matching, that $|M| \leq |M^*| \leq 2|M|$. This shows that the size of a maximal matching is at most half the size of a maximal matching.

25. Let $M^*$ be a maximum matching in $G = (V, E)$ and let $U$ be the set of nodes in $G$ that are uncovered by $M^*$. Prove that there are no edges between any pair of nodes in $U$. (A set of nodes where no pair of nodes has an edge between them is called an ***independent set***).

26. Does your answer from the previous question hold if $M$ is a maximal matching, even if it is not necessarily maximum? If so, prove it. If not, give a counterexample.

27. An ***edge cover*** of a graph $G = (V, E)$ is a set $\rho \subseteq E$ such that every node in $V$ is adjacent to some edge in $\rho$. Prove that if $M^*$ is a maximum matching in $G$, and that if every node in $V$ is adjacent to at least one edge, that there is an edge cover of $G$ that uses at most $|M^*| + |U|$ edges, where $U$ is the set of nodes uncovered by $M^*$.

28. Prove or disprove: If every node in a graph $G$ has two edges incident to it, then $G$ has a perfect matching.

29. Our proof of Berge's theorem had several lemmas whose proof was left as an exercise. In this question, fill in those proofs to complete the proof of Berge's theorem: ★

    1. Prove **Lemma 2**, that if $M$ is a matching in $G$ and $P$ is an augmenting path, that $|M \Delta P| > |M|$.

    2. Prove **Lemma 3**, that if $M$ and $N$ are matchings in $G = (V, E)$, then every connected component of the graph $(V, M \Delta N)$ is either an isolated node, or a simple path, or a simple cycle.

30. The ***degree*** of a node in an undirected graph is the number of edges incident to it. Prove that the sum of the degrees of all nodes in a graph is even. This is sometimes called the ***handshake lemma***,

because if you treat each edge in the graph as a pair of people shaking hands, the total number of hands shaken is always even.

31. Prove that the number of nodes in a graph with odd degree is even.

# Chapter 5 Relations

In the previous chapter, we explored graphs as a way of modeling connections between objects. By studying graphs, we were able to answer the following questions:

- How robust are the connections between objects? Can we break a single connection and fragment the objects, or must we break several edges?
- When can we prioritize objects based on the connections between them?
- If we meander around these connections, will we ever get back where we started?

In answering these questions, we began to categorize graphs based on their properties. We studied connected graphs, 2-edge-connected graphs, trees, DAGs, and strongly-connected graphs, and saw how each of them had their own unique properties.

However, there are many other ways that we might want to categorize the relations between objects. For example, consider a set of objects related by the property "is the same shape as" or "is the same color as." Although these are different ways of relating objects to one another, they have many similar properties. In either case, we can cluster all of the objects into groups based on their similarities to one another. Similarly, consider the relationships of "is tastier than" and "is a subset of." These are totally different relations, but in both cases we can use this relation to build a ranking of the different objects.

This chapter presents a different way of thinking about connections between objects by focusing specifically on the properties of how those objects are connected together. In doing so, we will build up a useful set of terminology that will make it possible for us to reason about connections between objects, even if we have never encountered that particular connection before.

## 5.1 Basic Terminology

### 5.1.1 Tuples and the Cartesian Product

Our goal in this chapter is to explore the ways in which different objects can be related to one another. In order to do this, we will first need to define a few more formalisms that we will use as our starting point in the study of relations.

In the previous chapter on graphs, we first defined graphs informally as a collection of objects (nodes/vertices) and connections (edges/arcs). We then formalized this definition by saying that a graph was an ordered pair of two sets – a set of nodes and a set of edges (which in turn were defined as ordered pairs). Before we begin exploring the properties of relations between objects, we will first introduce a few formalisms that will make it possible for us to more precisely discuss relationships mathematically.

For starters, when we discuss relations, we will be describing properties that hold among groups of objects (usually, two or three objects). Sometimes, these relationships will have some ordering associated with them. For example, if $x$ and $y$ are related by the relation "$x$ is less than $y$," then it's important for us to differentiate the roles of $x$ and $y$ in this relationship. Specifically, we'd need to know that $x$ was less than $y$, and not the other way around. Similarly, if $t$, $u$, and $v$ are related by the relationship "$t$ watched movie $u$ while eating snack $v$," we have to remember which of $t$, $u$, and $v$ is the person, the movie, and the snack.

Of course, not all relations between objects have an ordering enforced on them. For example, in the relation "$x$ and $y$ are the same height," it doesn't matter whether we interchange the roles of $x$ and $y$; if $x$ and $y$ have

the same height, then $y$ and $x$ also have the same height. We'll discuss how to handle this in just a short while. But first, we'll introduce this definition:

> A ***tuple*** is a collection of $n$ (not necessarily distinct) objects in some order. We denote the ordered tuple containing objects $x_1$, $x_2$, …, $x_n$ as $(x_1, x_2, \ldots, x_n)$. To specifically indicate that a tuple contains $n$ elements, we sometimes call it an ***n-tuple***.
>
> Two tuples are equal iff they have the same elements in the same order.

For example, (1, 2, 3) and (1, 1, 1, 1, 1) are both tuples. The tuples (1, 2, 3) and (1, 2, 3) are equal to one another, but (1, 2, 3) and (3, 2, 1) are not because although they have the same elements, they don't appear in the same order. Similarly, (1, 1, 2) ≠ (1, 2), since there are more elements in (1, 1, 2) than in (1, 2).

Given some set $A$, we can think about $n$-tuples formed from the elements of $A$. For example, if we take the set $\mathbb{N}$, then the 4-tuples we can make from elements of $\mathbb{N}$ are

(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 0, 2), …,

(0, 0, 1, 0), (0, 0, 1, 1), (0, 0, 1, 2), …,

…

(0, 1, 0, 0), (0, 1, 0, 1), (0, 1, 0 ,2), …,

…

There are infinitely many such 4-tuples here, though exactly how big this infinity is is a topic for the next chapter. We can think about gathering all of these elements together into a set that contains all of these 4-tuples. In order to do this, we will introduce a new fundamental operation on sets that can be used to construct sets of tuples from individual sets.

> Let $A$ and $B$ be sets. The ***Cartesian product*** of $A$ and $B$, denoted $A \times B$, is the set
>
> $$A \times B = \{ (a, b) \mid a \in A \text{ and } b \in B \}$$

Intuitively, $A \times B$ is the set of all ordered pairs whose first element is in $A$ and whose second element is in $B$. For example, if we take the sets

$$A = \{ \mathbf{1}, \mathbf{2}, \mathbf{3} \} \qquad\qquad B = \{ \mathbf{x}, \mathbf{y} \}$$

Then $A \times B$ is the set

$$A \times B = \{ (\mathbf{1}, \mathbf{x}), (\mathbf{1}, \mathbf{y}), (\mathbf{2}, \mathbf{x}), (\mathbf{2}, \mathbf{y}), (\mathbf{3}, \mathbf{x}), (\mathbf{3}, \mathbf{y}) \}$$

Notice that the ordering of the sets in the Cartesian product matters. For example, given the sets $A$ and $B$ above, then $B \times A$ is the set

$$B \times A = \{ (\mathbf{x}, \mathbf{1}), (\mathbf{x}, \mathbf{2}), (\mathbf{x}, \mathbf{3}), (\mathbf{y}, \mathbf{1}), (\mathbf{y}, \mathbf{2}), (\mathbf{y}, \mathbf{3}) \}$$

Consequently, $A \times B \neq B \times A$.

Given our definition of the Cartesian product, it's legal to consider the product of some set with the empty set. For example, we can take $A \times \varnothing$. But what exactly does this give us? If we look at the definition, we'll see that $A \times \varnothing$ is defined as

$$A \times \varnothing = \{ (a, b) \mid a \in A \text{ and } b \in \varnothing \}$$

This is the set of all pairs whose second element is contained in the empty set. But since nothing is contained in the empty set, there can't be any pairs $(a, b)$ whose second element is in the empty set. Consequently, the above set contains nothing. That is, $A \times \varnothing = \varnothing$.

It is possible to take the Cartesian product of more than two sets at the same time. For example, we can consider the set of all ordered *triples* made from elements of three sets $A$, $B$, and $C$. If we take $A$, $B$, and $C$ as follows:

$$A = \{ 1, 2, 3 \} \qquad B = \{ \text{x}, \text{y} \} \qquad C = \{ \star, \blacksquare \}$$

Then $A \times B \times C$ would be

$$A \times B \times C = \{ (1, \text{x}, \star), (1, \text{y}, \star), (2, \text{x}, \star), (2, \text{y}, \star), (3, \text{x}, \star), (3, \text{y}, \star),$$
$$(1, \text{x}, \blacksquare), (1, \text{y}, \blacksquare), (2, \text{x}, \blacksquare), (2, \text{y}, \blacksquare), (3, \text{x}, \blacksquare), (3, \text{y}, \blacksquare) \}$$

If you're a mathematical stickler, you might exclaim "Hold on a second! The Cartesian product is only defined on pairs of sets, not triples of sets!" If so, you'd be absolutely right. Our definition of the Cartesian product indeed only applies to pairs of sets. To resolve this, let's define how the Cartesian product applies to multiple sets.

First, let's specify that $A \times B \times C$ is interpreted as $A \times (B \times C)$. Under that definition, we would have that

$$A \times B \times C = A \times (B \times C) = \{ (1, (\text{x}, \star)), (1, (\text{y}, \star)), (2, (\text{x}, \star)), (2, (\text{y}, \star)),$$
$$(3, (\text{x}, \star)), (3, (\text{y}, \star)), (1, (\text{x}, \blacksquare)), (1, (\text{y}, \blacksquare)),$$
$$(2, (\text{x}, \blacksquare)), (2, (\text{y}, \blacksquare)), (3, (\text{x}, \blacksquare)), (3, (\text{y}, \blacksquare)) \}$$

This is similar to what we had before, but it's not exactly the same thing. In particular, note that each of the entries of this set is a pair, whose first element is from $A$ and whose second element is a pair of an element from $B$ and an element from $C$. How are we to reconcile this with our above set of triples? The answer lies in how we formally mathematically specify what an $n$-tuple is. It turns out that it's possible to construct $n$-tuples given only ordered pairs. Specifically, we have the following:

> The $n$-tuples are defined inductively. Specifically:
>
> The 2-tuple $(x_1, x_2)$ is the ordered pair $(x_1, x_2)$.
> For $n \geq 2$, the $(n+1)$-tuple $(x_1, x_2, \ldots, x_{n+1})$ is the ordered pair $(x_1, (x_2, \ldots, x_n))$

For example, formally speaking, the 5-tuple $(a, b, c, d, e)$ would be represented as the ordered pair $(a, (b, (c, (d, e))))$.

For notational simplicity, we will always write out $n$-tuples as $n$-tuples, rather than as nested ordered pairs. However, it's important to note that we formally define $n$-tuples in terms of ordered pairs so that if we want to consider many-way Cartesian products (say, like the Cartesian product of twenty or thirty sets), we can do so purely in terms of the binary Cartesian product operator. From this point forward, we'll just write out $A \times B \times C \times D$ instead of $A \times (B \times (C \times D))$.

### 5.1.1.1 Cartesian Powers

One interesting application of the Cartesian product arises if we take the Cartesian product of a set and itself. For example, consider the set $A$ defined as

$$A = \{ 1, 2, 3 \}$$

In that case, the set $A \times A$ is the set

$$A \times A = \{ (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3) \}$$

This is the set of all ordered pairs that we can make from pairs of elements in $A$. This set arises frequently in discrete mathematics, graph theory, and the study of functions and relations. As a result, we give this set a special name and its own terminology:

> The ***Cartesian square*** of $A$, denoted $A^2$, is the set $A \times A$.

The Cartesian square is an important set. Let's think back to the previous chapter on graphs for one application. Recall that a directed graph is defined as a pair $G = (V, E)$, where $V$ is a set of nodes and $E$ is a set of edges. This set $E$ consists of a set of ordered pairs representing directed edges. Although there are many different types of graphs that we can make, there are some serious restrictions on the set $E$. For example, in a graph whose nodes are people, we wouldn't expect (1, 2) to be an edge in the graph. The reason for this is that neither 1 nor 2 are people, and so they aren't nodes in the graph. Consequently, the set $E$ in a graph must be constrained so that each edge's endpoints must be nodes in the graph. This means that each edge in $E$ must be an ordered pair whose components are contained in $V$. In other words, the set $E$ must satisfy $E \subseteq V^2$.

The use of the superscript $^2$ here to indicate the Cartesian square comes from the fact that we are using the times symbol $\times$ to represent the Cartesian product. Just as exponentiation is repeated multiplication, the Cartesian square represents repeated Cartesian products. Given that we can "square" a set using the Cartesian square, can we raise sets to other powers? Intuitively, this should be easy to define. For example, we could define $A^3 = A \times A \times A$, or $A^4 = A \times A \times A \times A$, etc. In fact, we can do just that. We will define this inductively:

> For $n \geq 1$, the $n$th ***Cartesian power*** of a set $A$, denoted $A^n$, is the set formed by taking the Cartesian product of $A$ with itself $n$ times. Formally:
>
> $A^1 = A$
> $A^{n+1} = A \times A^n$

The inductive definition given above simply is a formal way of describing how we would compute Cartesian powers. For example, If we take $A = \{ 0, 1 \}$, then we would compute $A^4$ as follows:

$A^4 =$

$A \times A^3 =$

$A \times A \times A^2 =$

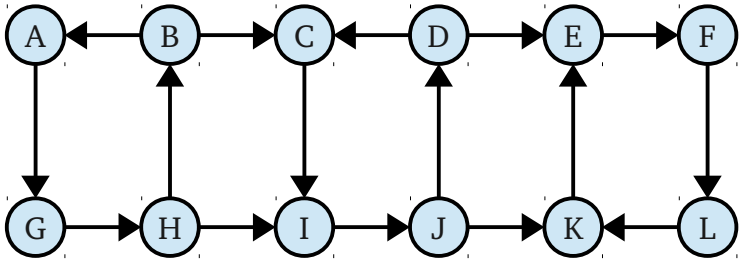
$A \times A \times A \times A = \{ (0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1),$
$(0, 1, 0, 0), (0, 1, 0, 1), (0, 1, 1, 0), (0, 1, 1, 1),$
$(1, 0, 0, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 0, 1, 1),$
$(1, 1, 0, 0), (1, 1, 0, 1), (1, 1, 1, 0), (1, 1, 1, 1) \}$

As we continue exploring more mathematical structures, we will often see operations similar to exponentiation defined in terms of operations similar to multiplication.

## 5.1.2 A Formal Definition of Relations

The object of study for this chapter will be relationships between objects. In order to study this object, we will need to formalize a definition of a relationship.

First, let's give an informal definition of a relation: we'll say that a relation is some property that holds true for certain groups of objects. For example, if the relation is "$x$ is less than $y$," then this relation would hold for 1 and 2, for 2 and 3, for 3 and 4, etc. If the relation is "$x + y$ is less than $z$," then the relation would hold for 1, 2, and 4; for 3, 4, and 8; for 0, 0, and 1; etc. If the relation is "$x$ is reachable from $y$," then in the following graph:




The relation would hold between $I$ and $A$ (because $I$ is reachable from $A$), between $L$ and $G$, between $I$ and $C$, etc.

The approach we have taken here when defining relations has been to define some property, then think of all the groups of objects that satisfy that property. Let's think about this using set-builder notation. We could consider the following set $R$ of all pairs of natural numbers that satisfy the relation "$x$ is less than $y$":

$$R = \{ (x, y) \in \mathbb{N}^2 \mid x < y \}$$

This set stores ordered pairs, since the ordering of the elements definitely matters, and more importantly it stores all the ordered pairs where $x < y$ holds.

As another example, consider the following set $R_G$, which, for a given graph $G = (V, E)$, contains all pairs of nodes that are strongly connected to one another:

$$R_G = \{ (x, y) \in V^2 \mid x \leftrightarrow y \}$$

We could also consider the following set $S$, which holds all triples of numbers where the product of the first two numbers is equal to the third:

$$S = \{ (x, y, z) \in \mathbb{R}^3 \mid xy = z \}$$

Notice that in each case, we are able to start off with some arbitrary property (less than, reachability, etc.) and convert it into a set of tuples. Mathematically speaking, we can check whether a group of objects has the given property by seeing whether or not it is contained in the appropriate set. For example, when talking about the less-than relation, we have that

$$x < y \qquad \text{iff} \qquad (x, y) \in R$$

Similarly, for strong connectivity, the following holds:

$$x \leftrightarrow y \qquad \text{iff} \qquad (x, y) \in R_G$$

We can also invert this process: given a set of tuples, we can define a relation between groups of objects based on that set. For example, consider the following set $T$:

$$T = \{ (0, 0), (0, 1), (0, 2), \ldots,$$
$$(1, 1), (1, 2), (1, 3), \ldots,$$
$$(2, 2), (2, 4), (2, 6), (2, 8), \ldots,$$
$$(3, 3), (3, 6), (3, 9), (3, 12), \ldots \}$$

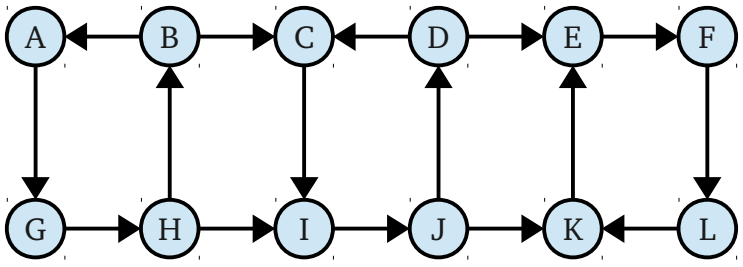
This set contains infinitely many ordered pairs of natural number. We can think about the relationship between objects defined as "$(x, y) \in T$." This relation is well-defined, though it might not immediately be obvious exactly what the relationship we've defined this way means.

This suggests that there is a close connection between relationships between groups of objects and sets of tuples. Specifically, for any relation we'd like, we can always gather up all of the tuples that satisfy that relation into a set containing all instances of the relation. Similarly, given a set of tuples, we can define a relation based on what tuples happen to be contained within the set.

This connection allows us to formalize the definition of a relation. Intuitively, a relation is some property that might hold of a cluster of objects. Formally, we will say that a relation is a set of tuples drawn from some number of sets:

> Let $S_1, \ldots, S_n$ be sets. A ***relation over $S_1, \ldots, S_n$*** is a set $R \subseteq S_1 \times \ldots \times S_n$.

For example, given the following graph $G = (V, E)$:




The relation "edge $v$ is the source of edge $e$" over $V$ and $E$ would be represented by the set

$$\{ (A, (A, G)), (B, (B, A)), (B, (B, C)), (C, (C, I)),$$
$$(D, (D, C)), (D, (D, E)), (E, (E, F)), (F, (F, L)),$$
$$(G, (G, H)), (H, (H, B)), (H, (H, I)), (I, (I, J)),$$
$$(J, (J, D)), (J, (J, K)), (K, (K, E)), (L, (L, K)) \}$$

Similarly, if the following set $R$ is a relation over $\mathbb{N}$ and $\mathbb{Z}$, although it's not immediately clear if there's any pattern to it:

$$R = \{ (1, -14), (137, 42), (271, -3111) \}$$

It may seem silly to allow us to treat *any* set of tuples as a relation, even if there's no clear reason what relates all the elements. However, this definition has many advantages. It allows us to use familiar set operations like union, intersection, difference, and Cartesian products to construct new relations or to modify existing relations, since as long as the result is a set of tuples we are left with a valid relation. It also allows us to consider relations that we know are valid even if we don't know what the "meaning" of that relation is. For example, the above relation $R$ might actually be meaningful, even if we don't know why the elements are related the way they are.

A strange consequence of the above definition is that many relations that we know and love, such as the less-than relation, can be defined as a set of ordered pairs. For example, the relation $<$ over the natural numbers would be

$$< = \{ (0, 1), (0, 2), (0, 3), \ldots, (1, 2), (1, 3), (1, 4), \ldots \}$$

This might seem hard to read, since we've started to treat the symbol $<$ not as a symbol we can place in-between two numbers, but as a mathematical object in of itself. That is, we are focusing on the essential properties of $<$, rather than on how particular mathematical expressions might relate to each other according to $<$.

When we study relations in depth in the remainder of this chapter, we will tend to focus primarily on relations between pairs of objects. These relations are called *binary relations*:

> A ***binary relation*** is a relation $R$ over some sets $A$ and $B$. A ***binary relation over a set A*** is a relation $R$ over the sets $A$ and $A$.

For example, the relation "$x$ divides $y$" is a binary relation over integers (or natural numbers, if you'd like), and the relation "$x$ is reachable from $y$" is a binary relation over nodes in a graph. Similarly, we can study $<$ as a binary relation over $\mathbb{N}$, if we so choose.

We formally defined a relation as a set of ordered tuples, which means that if we want to say something like "$x$ is less than $y$" we would write $(x, y) \in <$. However, in the special case of a binary relation, we typically would write this as $x < y$ rather than $(x, y) \in <$, since the former is significantly cleaner than the latter. More generally, we almost never see relations written out using this set-theoretic notation. In the case of binary relations, we almost always write the name of the relation in-between the two values it relates.

> Let $R$ be a binary relation over a set $A$. Then we write $aRb$ iff $(a, b) \in R$.

If this seems a bit strange, try replacing $R$ with a relation like $=$, $<$, $\leq$, $|$, or $\leftrightarrow$. In these cases, we would prefer to write out $a \leq b$ rather than $(a, b) \in \leq$, or $a \leftrightarrow b$ rather than $(a, b) \in \leftrightarrow$. For simplicity's sake, we'll adopt this convention throughout the rest of the course.

### 5.1.3 Special Binary Relations

In the remainder of this chapter, we will explore certain types of relations that arise frequently in discrete mathematics and computer science, and will analyze their properties. By studying groups of relations in the abstract, we will be able to immediately draw conclusions about concrete relations that we discover later on.

In order to motivate the definitions from later in this chapter, we will need to introduce a basic set of terms we can use to describe various types of relations. Given this vocabulary, we can then introduce broad categories of relations that all have certain traits in common.

To begin with, let's consider the following three relationships:

$$x \leq y$$

$x$ is in the same connected component as $y$

$x$ is the same color as $y$

These relations have wildly different properties from one another. The first of these deals with numbers, the second with nodes in a graph, and the third with objects in general. However, these three relations all do have two essential properties in common.

First, notice that each of these relations always relate an object to itself: $x \leq x$ is true for any number $x$, any node $x$ is always in the same connected component as itself, and any object is always the same color as itself. Not all binary relations have this property. For example, the relation "less-than" ($x < y$) does not relate numbers to themselves; the statement "$x < x$" is always false. Similarly, the relation "$(x, y) = (y, x)$" over ordered pairs sometimes relates ordered pairs to themselves (for example, $(0, 0) = (0, 0)$), but sometimes does not (for example, $(1, 0) \neq (0, 1)$).

If $R$ is a binary relation over a set $A$ that always relates every element of $A$ to itself, we say that $R$ is *reflexive*:

> A binary relation $R$ over a set $A$ is called ***reflexive*** iff for any $x \in A$, we have $xRx$.

Under this definition, the equality relation = is reflexive, but the less-than relation < is not.

Note that for a binary relation $R$ over a set $A$ to be reflexive, $R$ must relate every element $x \in A$ to itself. If we can find even a single element $x \in A$ such that $xRx$ does not hold, then we know that $R$ is not reflexive. In other words, a binary relation $R$ over a set $A$ is not reflexive iff there is some element $x \in A$ such that $xRx$ does not hold.

One new piece of notation: if we want to indicate that $xRy$ is false, we will denote this by drawing a slash through the $R$ in-between $x$ and $y$:

> If $xRy$ is false, we denote this by writing $x\not{R}y$. Equivalently, $x\not{R}y$ iff $(x, y) \notin R$.

In order for a binary relation $R$ over a set $A$ to be reflexive, $xRx$ has to hold for any element $x \in A$. A single counterexample suffices to show that $R$ is not reflexive. However, some relations go way above and beyond this by having *every* element $x \in A$ satisfy $x\not{R}x$. For example, all of the following relations have the property that no object is ever related to itself:

$$x \neq y$$

$x$ is not reachable from $y$

$x$ has more sides than $y$

These relations again concern different types of objects (numbers, nodes, and polyhedra), but they are unified by the fact that they never relate objects to themselves. Relations with this property are called *irreflexive*:

> A binary relation $R$ over a set $A$ is called ***irreflexive*** iff for any $x \in A$, we have $x \not R x$.

A critical detail here is that reflexive and irreflexive are **not** opposites of one another. A reflexive relation is one in which every object is *always* related to itself. An irreflexive relation is one in which every object is *never* related to itself. Relations might sometimes relate objects to themselves and sometimes not. These relations are neither reflexive nor irreflexive. If you want to prove that a relation is reflexive, it is not sufficient to show that the relation is not irreflexive. You'll usually directly demonstrate how every object must necessarily be related to itself.

Earlier, we mentioned that there are two key properties unifying these relations:

$x \leq y$

$x$ is in the same connected component as $y$

$x$ is the same color as $y$

The first of these properties is reflexivity – these relations always relate objects to themselves. However, there is one more key property in play here.

Let's look at $\leq$ for a minute. Notice that if $x \leq y$ and $y \leq z$, then it's also true that $x \leq z$. Similarly, if $x$ and $y$ belong to the same connected component and $y$ and $z$ belong to the same connected component, it's also true that $x$ and $z$ belong to the same connected component. Finally, if $x$ is the same color as $y$ and $y$ is the same color as $z$, then $x$ is the same color as $z$.

In each of these three cases, we said that if $xRy$ and $yRz$ (where $R$ is the appropriate binary relation), then it is also the case that $xRz$. Relations that have this property are called *transitive* and are very important in mathematics:

> A binary relation $R$ over a set $A$ is called ***transitive*** iff for any $x$, $y$, $z \in A$, that if $xRy$ and $yRz$, then $xRz$.

Not all relations are transitive. For example, the $\neq$ relation is not transitive, because $0 \neq 1$ and $1 \neq 0$, but it is not true that $0 \neq 0$. Notice that all we need to show to disprove that a relation is transitive is to find just one case where $xRy$ and $yRz$, but $x \not R z$.

There are two more properties of relations that we should cover before we move on to the next section. Consider this group of relations:

$x = y$

$x \neq y$

$x \leftrightarrow y$

The first and last of these relations are reflexive and transitive, while the second is irreflexive and is not transitive. However, each of these relations does have one property in common. Suppose that $xRy$, where $R$ is one of the above relations. In that case, we can also infer that $yRx$. More specifically: if $x = y$, then $y = x$; if $x \neq y$, then $y \neq x$; and if $x \leftrightarrow y$, then $y \leftrightarrow x$. Relations that have this property are called *symmetric*, since if we flip the objects being related, the relation still holds:

> A binary relation $R$ over a set $A$ is ***symmetric*** iff for all $x, y \in A$, that if $xRy$, then $yRx$.

Many important relations, such as equality, are symmetric. Some relations, however, are not. For example, the relation ≤ over natural numbers is not symmetric. That is, if $x \leq y$, it's not necessarily guaranteed that $y \leq x$. For example, although $42 \leq 137$, it is not the case that $137 \leq 42$. That said, in some cases we can interchange the arguments to ≤; for example, $1 \leq 1$, and if we interchange the arguments we still get $1 \leq 1$, which is true. As with reflexivity, symmetry is an all-or-nothing deal. If you can find a single example where $xRy$ but $y \not R x$, then $R$ is not symmetric.

Some relations are "really, really not symmetric" in that whenever $xRy$, it is guaranteed that $y \not R x$. For example, all of these relations satisfy this property:

$$x < y$$

$$x \subset y$$

$x$ is equal to $y + 1$

Relations like these are called *asymmetric*:

> A binary relation $R$ over a set $A$ is ***asymmetric*** iff for all $x, y \in A$, that if $xRy$, then $y \not R x$.

As with reflexivity and irreflexivity, symmetry and asymmetry are not opposites of one another. For example, the relation ≤ over natural numbers is neither symmetric nor asymmetric.

## 5.1.4 Binary Relations and Graphs

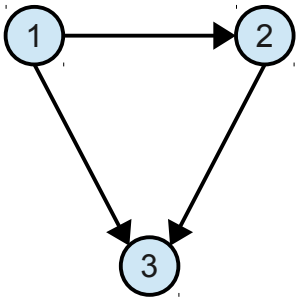
There is an intimate connection between directed graphs and binary relations over a set. Recall that a directed graph is a pair $(V, E)$, where $V$ is a set of nodes and $E$ is a set of edges connecting these nodes together. Each edge $e \in E$ is represented as an ordered pair whose first component is the origin of the edge and whose second component is the destination of the edge.

Recall that we've defined a binary relation $R$ over a set $A$ as the set of ordered pairs for which the relation holds. That is, $xRy$ means that $(x, y) \in R$. In this way, we can think of any binary relation $R$ over a set $A$ as the graph $(A, R)$, where $A$ is the set of nodes and each edge represents a relation between objects.

As an example of this, consider the relation < over the set $\{1, 2, 3\}$. The relation < would then be defined by the ordered pairs

$$< = \{ (1, 2), (1, 3), (2, 3) \}$$

We could interpret this as a graph whose nodes are 1, 2, and 3 and with the indicated edges. Such a graph would look like this:

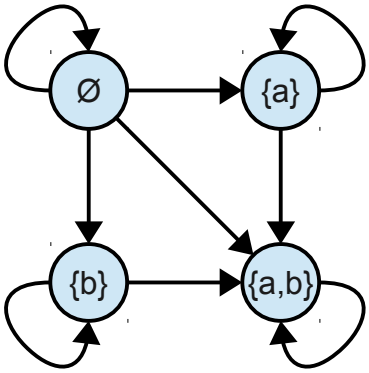



Notice that each directed edge from a number $m$ to a number $n$ represents a case where $m < n$. Similarly, any time that $m < n$, there will be an edge from $m$ to $n$. This means that the above picture is a graphical way of representing the relation < over the set {1, 2, 3}.

Similarly, consider the relation ⊆ over the set { Ø, {a}, {b}, {a, b} }. The relation ⊆ would then be defined by the ordered pairs

$$\subseteq = \{ (Ø, Ø), (Ø, \{a\}), (Ø, \{b\}), (Ø, \{a, b\}),$$
$$(\{a\}, \{a\}), (\{a\}, \{a, b\}),$$
$$(\{b\}, \{b\}), (\{b\}, \{a, b\}),$$
$$(\{a, b\}, \{a, b\}) \}$$

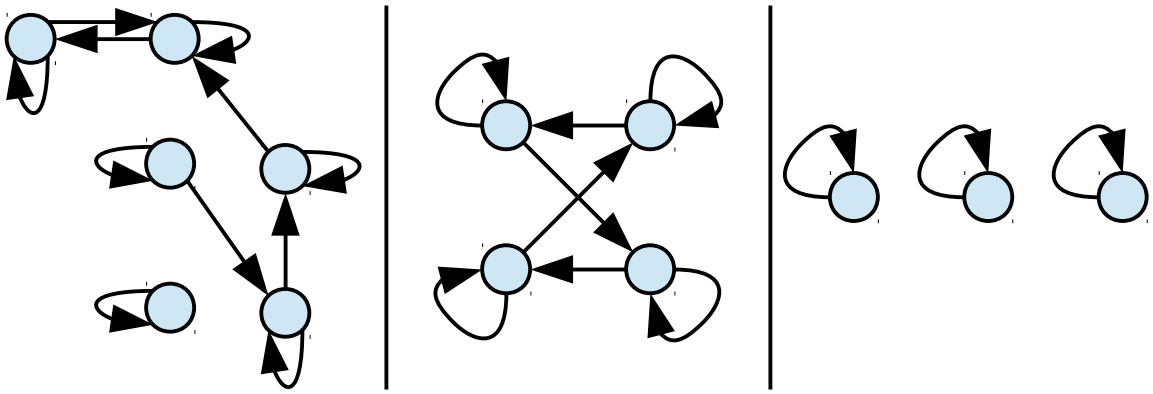
We could then interpret this as the following graph:



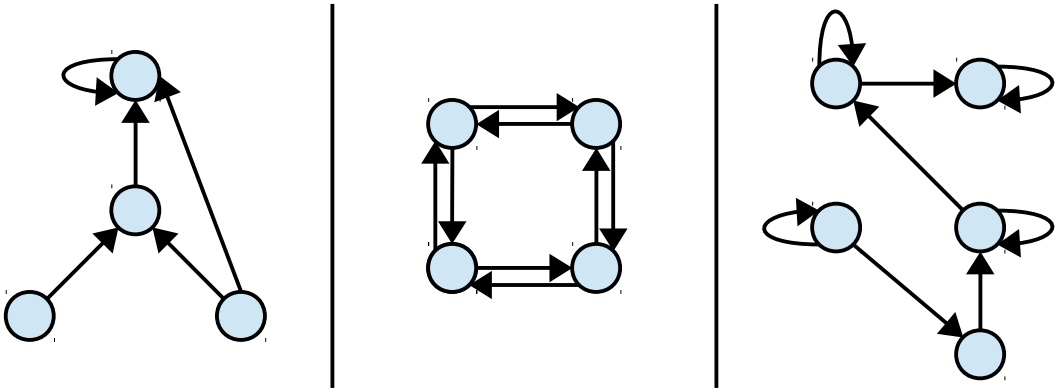

Notice here that a set $S$ has a directed edge to a set $T$ iff $S \subseteq T$. The above picture is just another way of presenting the ⊆ relation.

Given this connection between graphs and relations, we can revisit the definitions from the previous section graphically by visualizing what these relations mean in terms of graph structure. By seeing these definitions from both a symbolic and graphical view, I hope that it is easier to build up an intuition for what these definitions capture.

For example, take reflexive relations. These are binary relations $R$ over sets $A$ such that for any $x \in A$, $xRx$. This means that if we were to create a graph where the nodes are the elements of $R$, then each node must have an edge to itself. For example, all of the following graphs represent reflexive relations:

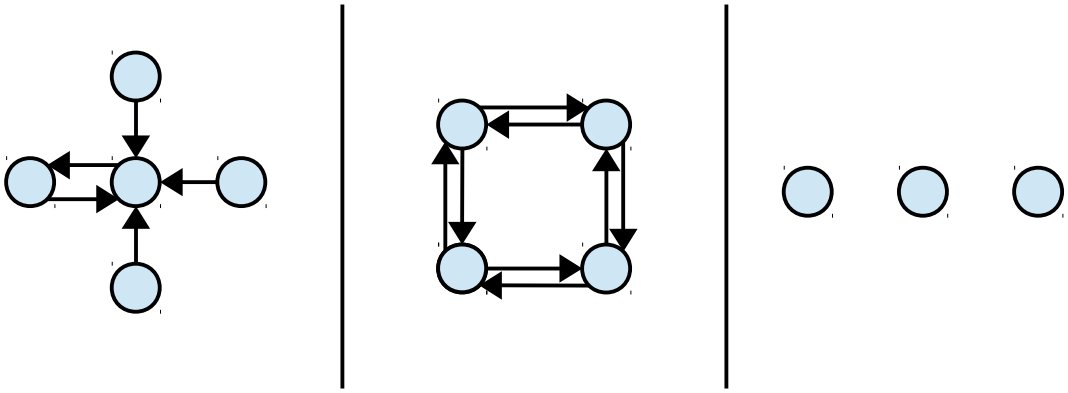


While these graphs represent relations that are not reflexive:

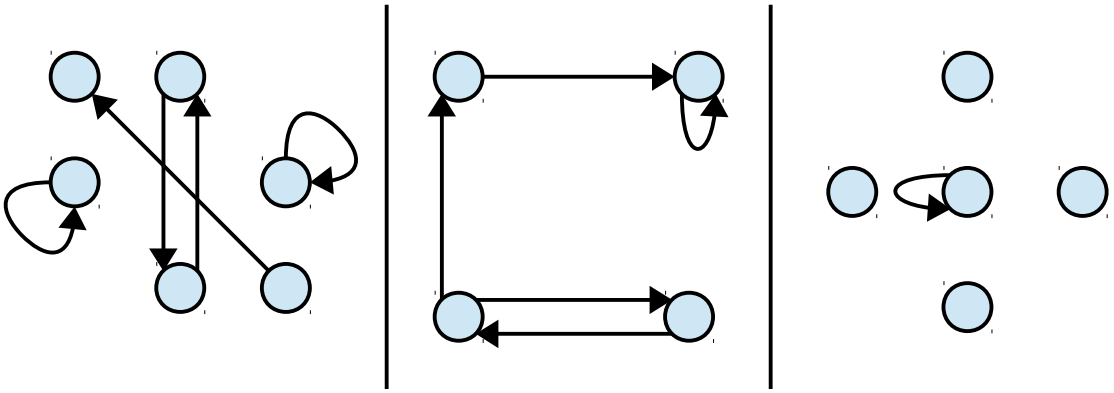


Take a minute to confirm why none of these relations are reflexive.

Similarly, a relation that is *irreflexive* (that is, for any $x \in A$, we know $x \not R x$) would be represented by a graph in which *no* node has an edge to itself. For example, all of the following graphs represent irreflexive relations:



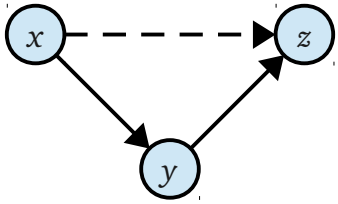
While none of the following graphs represent irreflexive relations:



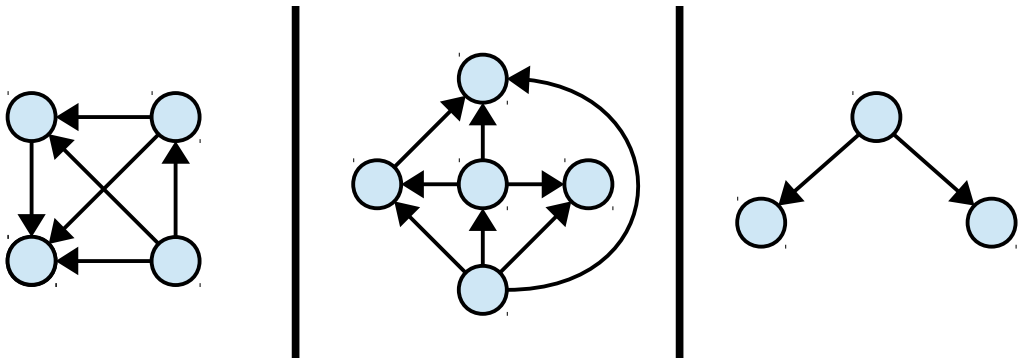
Again, make sure you understand why none of these graphs represent irreflexive relations.

The next property of relations we explored was transitivity. If you'll recall, a binary relation $R$ over a set $A$ is called transitive iff for any $x$, $y$, $z \in A$, that if $xRy$ and $yRz$, then $xRz$. What does this mean from a graph-theoretic perspective? Well, $xRy$ means that there is an edge from $x$ to $y$, and $yRz$ means that there is an edge from $y$ to $z$. Transitivity says that if we have these edges, then it must also be the case that $xRz$, meaning that there is an edge from $x$ to $z$.

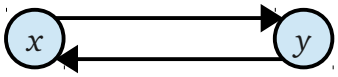
This suggests an elegant, graph-theoretic intuition for transitivity. Suppose that you have three nodes $x$, $y$, and $z$. Transitivity says that if you can get from $x$ to $z$ by going from $x$ to $y$ and from $y$ to $z$, then you can also get directly from $x$ to $z$. This is shown below:




You can thus think of a transitive relation as one that guarantees that all paths can be "shortcutted." If it's ever possible to get from one node to another along any path, there will always be a direct path from the first node to the destination.* As a result, these graphs all represent transitive relations:

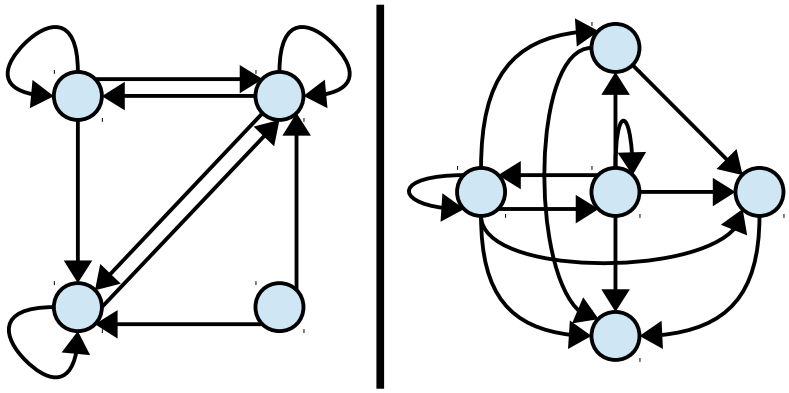


One important detail about transitivity. Suppose that we have a pair of nodes like these, in which each node has an edge to the other:

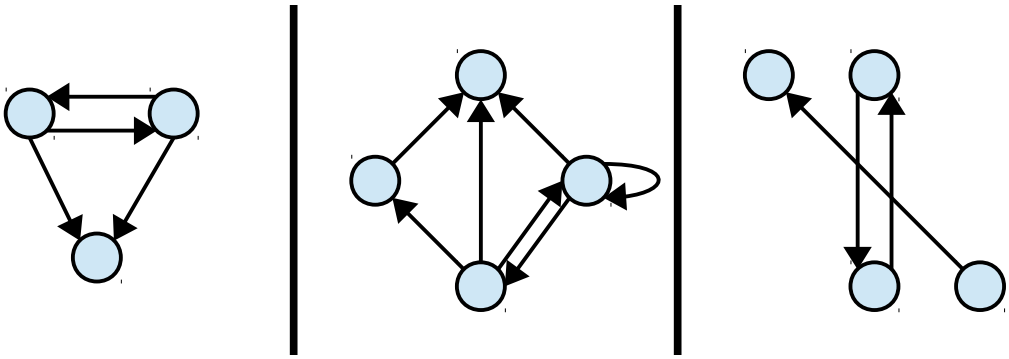



* One of the chapter exercises asks you to formally prove this.

In this case, if the relation is transitive, there must be edges from each node to itself. To see why, note that from the above graph, we know that $xRy$ and that $yRx$. By the definition of transitivity, this means that $xRx$. Similarly, since $yRx$ and $xRy$, we are guaranteed that $yRy$. Consequently, both of the following graphs represent transitive relations:

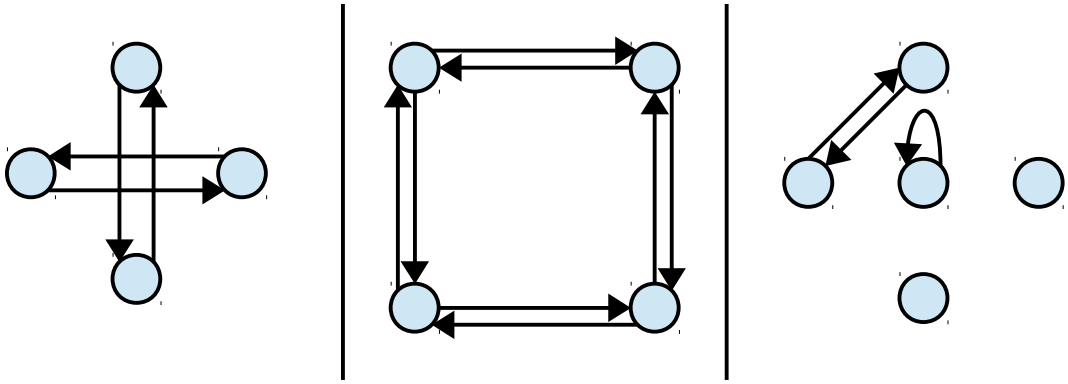


While none of these do:

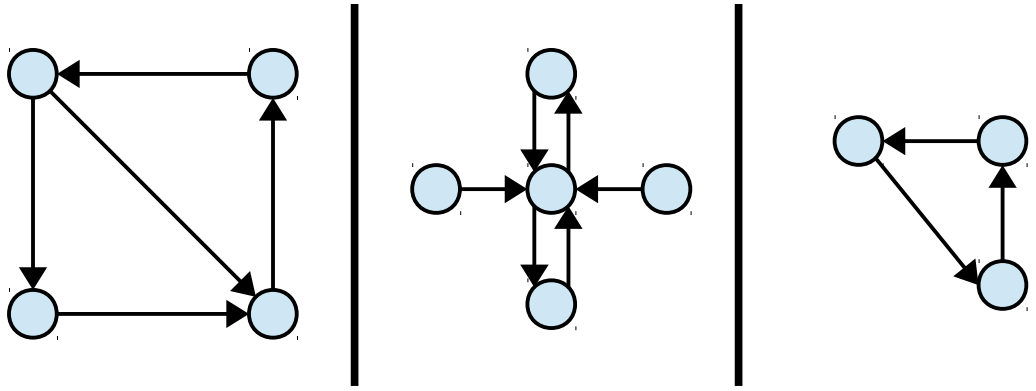


As before, make sure you understand why the above graphs aren't transitive.

Finally, let's consider the last two properties of relations we introduced: symmetry and asymmetry. Recall that $R$ is a symmetric relation over $A$ iff whenever $xRy$, it's also the case that $yRx$. From a graph-theoretic perspective, this means that any time there is an edge from $x$ to $y$, there must also be an edge back from $y$ to $x$. In other words, for any pair of nodes $x$ and $y$, either there are edges going between them in both directions, or there are no edges between them at all. Consequently, the following graphs all represent symmetric relations:



While these graphs do not:

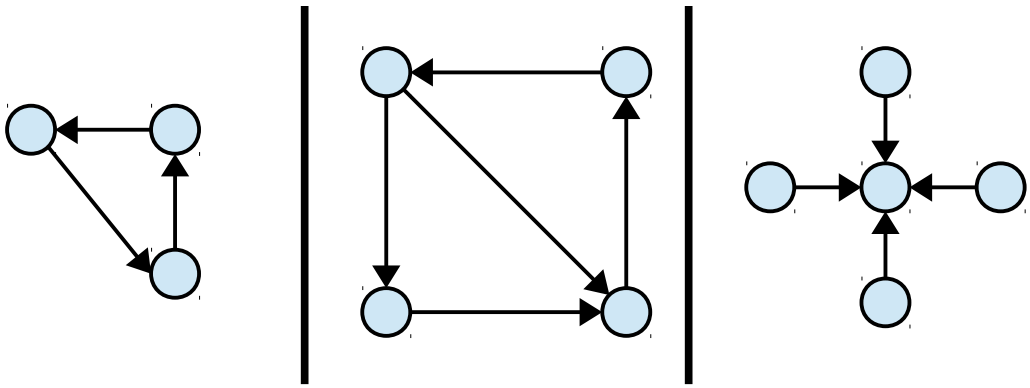


Note that it's perfectly fine for a node to have an edge from itself. This represents $xRx$, which is acceptable in symmetric relations because it's true that if $xRx$, then after swapping $x$ and $x$, we still get that $xRx$.

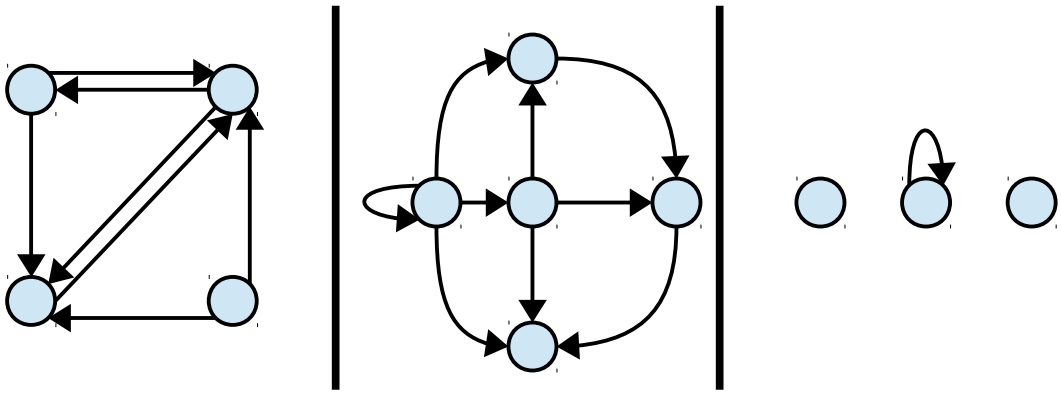
The case of asymmetry is slightly different. In an asymmetric relation, if $xRy$, then $y\not{R}x$. This means that if there is an edge from $x$ to $y$, then there cannot be an edge from $y$ to $x$. In other words, if there is an edge from one node to another, there cannot be another edge going the other way. Consequently, for any pair of nodes, there are either zero or one edges between them.

Additionally, asymmetric relations cannot have any edges from a node to itself. The reasoning is the following: if $xRx$, then since $R$ is asymmetric, we should have $x\not{R}x$, which is impossible. Consequently, no asymmetric relation has self-loops.

Given these descriptions, we have that the following graphs all represent asymmetric relations:



While these graphs do not:



# 5.2 Equivalence Relations

Now that we have some terminology we can use to describe relations, we will explore several important classes of relations that appear repeatedly throughout computer science and discrete mathematics.

The first type of relation that we will explore is the *equivalence relation*. Informally, an equivalence relation is a binary relation over some set that tells whether two objects have some essential trait in common. For example, = is an equivalence relation that tells whether two objects are identically the same. The connectivity relation $\leftrightarrow$ in a graph is an equivalence relation that tells whether two nodes are in the same connected component as one another. The relation "$x$ is the same color as $y$" is an equivalence relation that tells whether two objects share a color.

If we take these three relations as exemplars of equivalence relations, you will notice that they all have three traits in common:

- **Reflexivity**. Every object has the same traits as itself. Thus our relation should be reflexive.
- **Symmetry**. If $x$ and $y$ have some trait in common, then surely $y$ and $x$ have some trait in common.
- **Transitivity**. If $x$ and $y$ have some trait in common and $y$ and $z$ have the same trait in common, then $x$ and $z$ should have that same trait in common.

These three properties, taken together, are how we formally define equivalence relations:

> A binary relation $R$ over a set $A$ is called an ***equivalence relation*** iff it is reflexive, symmetric, and transitive.

Some of the relations that we have seen before are equivalence relations. For example, consider any undirected graph $G = (V, E)$ and the connectivity relation $\leftrightarrow$ on that graph. We proved in Chapter 5 that this relation is reflexive, symmetric, and transitive, though at the time we didn't actually use those names. Consequently, $\leftrightarrow$ is an equivalence relation over nodes in graphs.

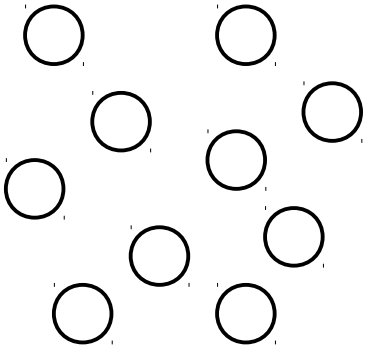
## 5.2.1 Equivalence Classes

An important observation at this point is that while intuitively we want equivalence relations to capture the notion of "$x$ and $y$ have some trait in common," our definition says absolutely nothing about this. Instead, it focuses purely on three observable traits of relations: reflexivity, symmetry, and transitivity.

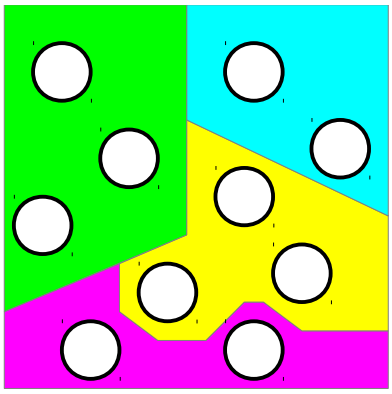
This definition might initially seem somewhat arbitrary. Why is this class of relations at all interesting? And why does this capture our idea of a relation indicating whether objects have traits in common?

One key property of equivalence relations is that we can use them to partition objects into distinct groups, all of which share some key property. For example, given the equivalence relation "$x$ is the same color as $y$," we could break objects apart into groups of objects that all have the same color as one another. Given the equivalence relation $x \leftrightarrow y$, we could break nodes in a graph apart into groups of nodes that are all mutually connected to one another (those groups are the connected components of the graph; more on that later). Even if we take the somewhat silly equivalence relation $x = y$, we can still split objects into groups. We just end up partitioning all objects into groups of one object apiece.

To formalize this definition, let us introduce a few quick definitions. First, we need to define what it means to take some set $S$ and split it apart into different groups. Intuitively, this means that we start out with all of the elements of $S$:



Then cut them apart into smaller sets of elements:



When cutting the elements up into sets this way, there are two key properties that must hold. First, every element of the original set $S$ must belong to one of the smaller sets. Second, no element of the original set $S$ can belong to more than one of the smaller sets. In addition to these two key properties, we'll add one extra requirement. When cutting $S$ up into smaller sets $S_1$, $S_2$, …, we'll require that each of these sets is nonempty. After all, we want to distribute the elements of $S$ into a collection of smaller sets, and allowing one of these smaller sets to be empty doesn't really accomplish anything (it doesn't actually contain any of the elements of $S$).

These requirements are formalized in the following definition:

> Given a set $S$, a ***partition of S*** is a set $X \subseteq \wp(S)$ (that is, a set of subsets of $S$) with the following properties:
>
> 1. The union of all sets in $X$ is equal to $S$.
> 2. For any $S_1, S_2 \in X$ with $S_1 \neq S_2$, we have that $S_1 \cap S_2 = \varnothing$ ($S_1$ and $S_2$ are disjoint)
> 3. $\varnothing \notin X$.

For example, let $S = \{1, 2, 3, 4, 5\}$. Then the following is a partition of $S$:

$$\{ \{1\}, \{2\}, \{3, 4\}, \{5\} \}$$

As is this set:

$$\{ \{1, 4\}, \{2, 3, 5\} \}$$

As is this set:

$$\{ \{1, 2, 3, 4, 5\} \}$$

However, the following set is not a partition of $S$:

$$\{ \{1, 3, 5\}, \{2\} \}$$

Because the union of these sets does not give back $S$. The following set isn't a partition either:

$$\{ \{1, 2, 3\}, \{4\}, \{3, 5\} \}$$

Because 3 is contained in two of these sets. Finally, this set isn't a partition of $S$:

$$\{ \{1, 2, 3\}, \{4, 5\}, \varnothing \}$$

Since it contains the empty set.

One important observation is that it is indeed possible to build a partition over the empty set. Specifically, $\varnothing$ is a partition over $\varnothing$. You can check to see that $\varnothing$ obeys all of the required properties – the union of all the (nonexistent sets) in $\varnothing$ is the empty set itself. There are no two different sets in $\varnothing$, so vacuously all of the sets in $\varnothing$ are disjoint. Finally, $\varnothing \notin \varnothing$. Thus $\varnothing$ is a partition of itself.

There is a close connection between partitions and equivalence relations, as you'll see, but in order to explore it we will need to explore a few quick properties of partitions. First, we'll prove a simple but important lemma that we'll need while reasoning about partitions:

> ***Lemma:*** Let $S$ be a set and $X$ a partition of $S$. Then every element $u \in S$ belongs to exactly one set $Y \in X$.

In other words, this lemma says that if you cut the elements of $S$ apart into non-overlapping groups, every element of $S$ belongs to just one of those groups. This is a key feature of partitions, so before we proceed onward, we'll prove this result.

> ***Proof:*** Let $S$ be a set and $X$ a partition of $S$. We will show that every element $u \in S$ belongs to at least one set $Y \in X$ and to at most one set $Y \in X$.
>
> To see that every element $u \in S$ belongs to at least one set $Y \in X$, note that since $X$ is a partition of $S$, the union of all the sets in $S$ must be equal to $S$. Consequently, there must be at least one set $Y \in X$ such that $u \in Y$, since otherwise the union of all sets contained in $X$ would not be equal to $S$.
>
> To see that every element $u \in S$ belongs to at most one set $Y \in X$, suppose for the sake of contradiction that $u$ belongs to two sets $Y_1, Y_2 \in X$ with $Y_1 \neq Y_2$. But then $x \in Y_1 \cap Y_2$, meaning that $Y_1 \cap Y_2 \neq \varnothing$, a contradiction. We have reached a contradiction, so our assumption must have been wrong.
>
> Thus every element $u \in S$ belongs to at most one set $Y \in X$. ■

Because every element of $S$ belongs to exactly one set $Y \in X$, it makes sense to talk about "the set in $X$ containing $u$." For simplicity's sake, we'll introduce a new piece of notation we can use to describe this set.

If $S$ is a set and $X$ is a partition of $S$, then for any $u \in S$, we denote by $[u]_X$ the set $Y \in X$ such that $u \in Y$.

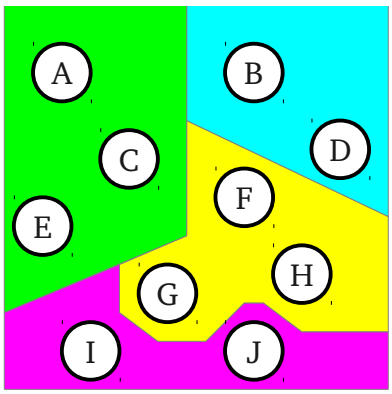
For example, if we let $S = \{1, 2, 3, 4, 5\}$ as before and consider the partition

$$X = \{ \{1, 3, 5\}, \{2\}, \{4\} \}$$

Then $[1]_X = \{1, 3, 5\}$, $[2]_X = \{2\}$, $[3]_X = \{1, 3, 5\}$, etc.

Armed with this notation for discussing partitions, let's start exploring the connection between partitions and equivalence classes. To begin with, let's suppose that we have some set $S$ and a partition $X$ of $S$. We can then define a binary relation $\sim_X$ over $S$ as follows. Intuitively, we will say that $u \sim_X v$ iff $u$ and $v$ belong to the same set in $X$. This means that when we've split the elements of $S$ apart using partition $X$, that $u$ and $v$ are grouped together. More formally, we'll define $\sim_X$ as follows: $u \sim_X v$ iff $[u]_X = [v]_X$. That is, $u$ and $v$ are in the same group.

To give an example of this relation, consider the following partition (which we'll call $X$) of this set (which we'll call $S$):




In this case, $A \sim_X C$, $B \sim_X D$, $I \sim_X J$, etc.

Given this new relation, we have the following theorem:

***Theorem:*** For any set $S$ and partition $X$ of that set, the relation $\sim_X$ is an equivalence relation over $S$.

How exactly will we go about proving this? Whenever asked to show that a relation has some property (for example, that it's an equivalence relation), we'll call back to the definition. In this case, equivalence relations are defined as relations that are reflexive, symmetric, and transitive. Consequently, to prove the above theorem, we'll prove that $\sim_X$ is reflexive, symmetric and transitive.

Intuitively, we can see that $\sim_X$ has these three properties as follows:

- **Reflexivity**: We need to show that for any $u \in S$, that $u \sim_X u$. Calling back to the definition of $\sim_X$, this means that we need to show that $[u]_X = [u]_X$, which is obvious.
- **Symmetry**: We need to show that for any $u, v \in S$ that if $u \sim_X v$, then $v \sim_X u$. This means we need to show that if $[u]_X = [v]_X$, then $[v]_X = [u]_X$. Again, this is obviously true.

- **Transitivity**: We need to show that for any $u$, $v$, $w \in S$, that if $u \sim_X v$ and $v \sim_X w$, then $u \sim_X w$. This means that if $[u]_X = [v]_X$ and $[v]_X = [w]_X$, then $[u]_X = [w]_X$. This is also obviously true by the transitivity of =.

We can formalize this proof below:

***Theorem:*** For any set $S$ and partition $X$ of that set, the relation $\sim_X$ is an equivalence relation over $S$.

***Proof:*** We need to show that $\sim_X$ is reflexive, symmetric, and transitive.

To see that $\sim_X$ is reflexive, we need to prove that for any $u \in S$, that $u \sim_X u$. By definition, this means that we need to show that $[u]_X = [u]_X$, which is true because = is reflexive.

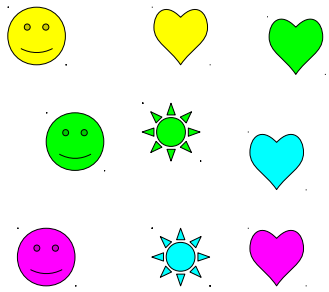
To see that $\sim_X$ is symmetric, we need to show that if $u \sim_X v$, then $v \sim_X u$. By definition, this means that we need to show that if $[u]_X = [v]_X$, then $[v]_X = [u]_X$. This is true because = is symmetric.

To see that $\sim_X$ is transitive, we need to show that if $u \sim_X v$ and $v \sim_X w$, then $u \sim_X w$. By definition, this means that we need to show that if $[u]_X = [v]_X$ and $[v]_X = [w]_X$, then $[u]_X = [w]_X$. This is true because = is transitive.

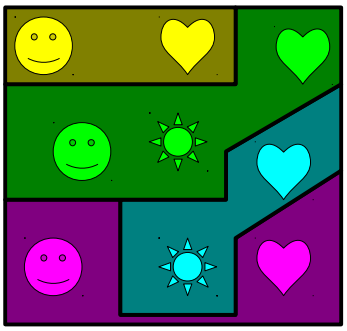
Since $\sim_X$ is reflexive, symmetric, and transitive, it is an equivalence relation. ■

Great! We've just shown that if we start off with a partition, we can derive an equivalence relation from it by saying that any pair of elements in the same group of the partition are equivalent.

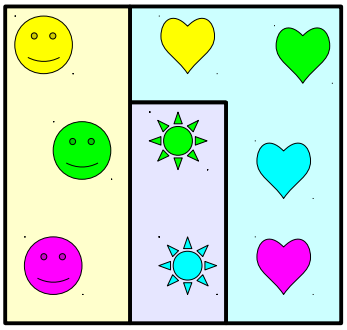
It turns out that we can also go the other way as well: given any equivalence relation $R$ over a set $S$, it's possible to construct a partition of $S$ by grouping together all the objects that are related together by $R$. To motivate this discussion, suppose that we have this collection of objects:



If we group all of these objects together by the equivalence relation "$x$ is the same color as $y$," then we end up with this partition of the set of objects:



Similarly, if we group these objects together by the equivalence relation "$x$ is the same shape as $y$," we get this partition:



The key insight behind why equivalence relations induce a partition of the elements is that it makes sense, given any element $u$, to speak of "all the elements equal to $u$." If you look at either of the above induced partitions, you can see that each set in the partition can be thought of as a set formed by picking any one of the elements in the set, then gathering together all the elements equal to it.

Given an equivalence relation $R$ over a set $A$ and some element $x \in A$, the *equivalence class* of $x$ is the set of all elements of $A$ that compare equal to $A$. This is formalized with the following definition:

> Let $R$ be an equivalence relation over a set $A$. Then for any $x \in A$, the ***equivalence class of x*** is the set $\{ y \in A \mid xRy \}$. We denote this set $[x]_R$.

When discussing equivalence classes, note that it's possible for two different elements of $A$ to have the same equivalence class. For example, in the equivalence relation "$x$ has the same color as $y$," the equivalence classes for any two red objects will be the same. Also notice that we are using similar notations for equivalence classes of an equivalence relations and the set in a partition containing some element. We will justify why this notation is so similar in the remainder of this section, but for now keep in mind that the notations do not necessarily represent the same thing.

Our goal in this section was to show how to turn an equivalence relation into a partition, and to do so we'll use equivalence classes. The key observation is as follows – if we have an equivalence relation $R$ over a set $A$, we will take as our partition the set of all equivalence classes for all the elements of $A$. For example, if our equivalence relation is "$x$ is the same color as $y$," then we would take as our partition of all elements the set of all red objects, the set of all orange objects, the set of all yellow objects, etc. Therefore, in the rest of this section, we will prove that the set of all equivalence classes of $R$ gives a partition of $A$.

First, some notational issues. How exactly do we define "the set of all equivalence classes of $R$?" To do this, we will consider this set:

$$X = \{ [x]_R \mid x \in A \}$$

This set is the set of the equivalence classes of every element in the set $A$. Notice that the rule we are using in the set-builder definition of $X$ might multiply-count the same equivalence class multiple times. For example, if an equivalence class $C$ contains three elements $a$, $b$, and $c$, then $C$ will get included in $X$ three times, since $[a]_R = [b]_R = [c]_R = C$. However, this is not a problem. Remember that sets are unordered collections of *distinct* objects, so if we include some equivalence class multiple times in $X$, it's the same as if we included it only once.

Now that we have this set $X$, how do we prove that it is a partition of $A$? Well, let's look back at the definition of a partition. We'll need to prove three things:

1. The union of all sets in $X$ is equal to $A$.
2. Any two non-equal sets in $X$ are disjoint.
3. $X$ does not contain the empty set.

Let's consider each of these in turn.

For starters, up to this point we've been using the term "the union of all sets in $X$" informally, without giving it a firm mathematical definition. If we want to reason about this union in the proof, we will need to actually define what this means.

In Chapter One, we defined the union of two sets as the set containing all elements in either of the original set. Let's generalize this definition to allow us to compute the union of *any* number of sets.

For example, let's suppose that we have two sets $A$ and $B$ and want to their union. This is the union $A \cup B$. In this case, we were computing the union of the sets contained within the set $\{A, B\}$. If we want to compute the union $A \cup B \cup C$, this would be computing the union of all the sets contained within the set $\{A, B, C\}$. More generally, if we have a set whose elements are other sets, we can take the union of all the sets contained within that set. This motivates the following definition:

> Let $S$ be a set whose elements are other sets. Then $\cup S = \{ x \mid \text{there is some } X \in S \text{ where } x \in X \}$.

For example:

$$\cup\{ \{1, 2, 3\}, \{3, 4\} \} = \{1, 2, 3, 4\}$$

$$\cup\{ \varnothing, \{1\}, \{2\}, \{3\} \} = \{1, 2, 3\}$$

$$\cup\varnothing = \varnothing$$

Given this new definition, let's start trying to prove that the set $X = \{ [x]_R \mid x \in A \}$ is a partition of the set $A$. First, we need to prove that the union of all sets in $X$ is equal to $A$. Using our new notation, this means that we want to prove that $\cup X = A$. Intuitively, this is true because we can show that every element of $A$ belongs to its own equivalence class, which in turn is an element of $X$.

***Lemma:*** Let $R$ be an equivalence relation over $A$, and $X = \{ [x]_R \mid x \in A \}$. Then $\cup X = A$.

***Proof:*** Let $R$ be an equivalence relation over $A$, and $X = \{ [x]_R \mid x \in A \}$. We will prove that $\cup X \subseteq A$ and $A \subseteq \cup X$, from which we can conclude that $\cup X = A$.

To show that $\cup X \subseteq A$, consider any $x \in X$. By definition of $\cup X$, since $x \in X$, this means that there is some $[y]_R \in X$ such that $x \in [y]_R$. By definition of $[y]_R$, since $x \in [y]_R$, this means that $yRx$. Since $R$ is a binary relation over $A$, this means that $x \in A$. Since our choice of $x$ was arbitrary, this shows that if $x \in \cup X$, then $x \in A$. Thus $\cup X \subseteq A$.

To show that $A \subseteq \cup X$, consider any $x \in A$. We will prove that $x \in [x]_R$. If we can show this, then note that since $x \in [x]_R$ and $[x]_R \in X$, we have $x \in X$. Since our choice of $x$ is arbitrary, this would mean that any $x \in A$ satisfies $x \in \cup X$, so $A \subseteq \cup X$.

So let's now prove that $x \in [x]_R$. By definition, $[x]_R = \{ y \in A \mid xRy \}$. Since $R$ is an equivalence relation, $R$ is reflexive, so $xRx$. Consequently, $x \in [x]_R$, as required. ■

Great! We've established that $\cup X = A$, which is one of the three properties required for $X$ to be a partition of $A$.

Notice that in the second half of this proof, we explicitly use the fact that $x$ is reflexive in order to show that $x \in [x]_R$. A corollary of this result is that $[x]_R \neq \varnothing$ for any $x \in A$. Consequently, we can also conclude that $\varnothing \notin X$. Formally:

***Lemma:*** Let $R$ be an equivalence relation over $A$, and $X = \{ [x]_R \mid x \in A \}$. Then $\varnothing \notin X$.

***Proof:*** Using the logic of our previous proof, we have that for any $x \in A$, that $x \in [x]_R$. Consequently, for any $x \in A$, we know $[x]_R \neq \varnothing$. Thus $\varnothing \notin X$. ■

All that is left to do now is to prove that any two sets $[x]_R, [y]_R \in X$ with $[x]_R \neq [y]_R$ are disjoint. To see why this would be true, let's think intuitively about how equivalence classes work. The class $[x]_R$ is the set of all objects equal to $x$, and similarly $[y]_R$ is the set of all objects equal to $y$. If $xRy$, then we would have that $[x]_R = [y]_R$, since the objects equal to $x$ are the same as the objects equal to $y$. On the other hand, if $x\not Ry$, then we should have that no object equal to $x$ would compare equal to $y$.

To formally prove this result, we can proceed by contrapositive. We'll show that if the equivalence classes $[x]_R$ and $[y]_R$ are *not* disjoint, then they must be equal. Intuitively, the proof works as follows. Since $[x]_R \cap [y]_R \neq \varnothing$, there is some element $w$ such that $w \in [x]_R$ and $w \in [y]_R$. This means that $xRw$ and $yRw$. From there, we can use symmetry and transitivity to get that $xRy$ and $yRx$. Once we've done that, we can conclude, using transitivity, that everything equal to $x$ is equal to $y$ and vice-versa. Thus $[x]_R$ and $[y]_R$ must be equal.

We can formalize this here:

***Lemma:*** Let $R$ be an equivalence relation over $A$, and $X = \{ [x]_R \mid x \in A \}$. Then for any two sets $[x]_R$, $[y]_R \in X$, if $[x]_R \neq [y]_R$, then $[x]_R \cap [y]_R = \emptyset$.

***Proof:*** Let $R$ be an equivalence relation over $A$, and $X = \{ [x]_R \mid x \in A \}$. We proceed by contrapositive and show that for any $[x]_R$, $[y]_R \in X$, that if $[x]_R \cap [y]_R \neq \emptyset$, then $[x]_R = [y]_R$.

Consider any $[x]_R$, $[y]_R \in X$ such that $[x]_R \cap [y]_R \neq \emptyset$. Then there must be some element $w$ such that $w \in [x]_R$ and $w \in [y]_R$. By definition, this means $w \in \{ z \in A \mid xRz \}$ and $w \in \{ z \in A \mid yRz \}$. Consequently, $xRw$ and $yRw$. Since $R$ is symmetric, this means that $xRw$ and $wRy$. Since $R$ is transitive, this means that $xRy$. By symmetry, we also have that $yRx$.

We will now use this fact to show that $[x]_R \subseteq [y]_R$. Without loss of generality, we can use this same argument to show that $[y]_R \subseteq [x]_R$, from which we can conclude that $[x]_R = [y]_R$, as required.

To show that $[x]_R \subseteq [y]_R$, consider any $z \in [x]_R$. This means that $xRz$. Since $yRx$ and $xRz$, by transitivity we have that $yRz$. Consequently, $z \in [y]_R$. Since our choice of $z$ was arbitrary, we have that any $z \in [x]_R$ satisfies $z \in [y]_R$. Thus $[x]_R \subseteq [y]_R$, as required. ■

Notice that this theorem relies on the fact that equivalence relations are symmetric and transitive. If we didn't have that property, then we couldn't necessarily guarantee that we could link up the sets $[x]_R$ and $[y]_R$ as we did. In our earlier lemmas, we used the fact that equivalence relations are reflexive. Collectively, this gives some justification as to why we define equivalence relations this way. If a relation is indeed reflexive, symmetric, and transitive, then regardless of its other properties it has to induce a partition of the elements of the underlying set.

By combining the three earlier lemmas together, we get the following overall result:

***Theorem:*** Let $R$ be an equivalence relation over $A$, and let $X = \{ [x]_R \mid x \in A \}$. Then $X$ is a partition of $A$.

The proof of this result is, essentially, "look at the previous three lemmas." Accordingly, we don't gain much by writing it out formally.

If we have an equivalence relation $R$ over a set $A$, then the set $X$ described above (the set of all the equivalence classes of $R$) has a nice intuition – this is the set formed by grouping together all of the elements in $A$ that are equal to one another. In other words, we divide the elements of the set $A$ into groups based on the equivalence relation $R$. The fact that we are "dividing" elements up this way gives rise to the term we use for this set $X$: it's called the *quotient set* of $A$ by $R$:

Let $R$ be an equivalence relation over $A$. The ***quotient set*** of $A$ by $R$ is the set of all equivalence classes of the elements of $A$ under $R$; that is, $\{ [x]_R \mid x \in A \}$. This set is denoted $A / R$.

For example, consider the set $\mathbb{N}$. One equivalence relation we can define over $\mathbb{N}$ is the relation "$x$ has the same parity as $y$," which is denoted $\equiv_2$. (There's a good reason for this notation; see the exercises for more

details). For example, $5 \equiv_2 3$, and $0 \equiv_2 100$. In this case, there are two equivalence classes: the equivalence class containing the even numbers 0, 2, 4, 6, 8, …, and the equivalence class containing the odd numbers 1, 3, 5, 7, 9, … . Consequently, the set $\mathbb{N} / \equiv_2$ would be the set $\{ \{ 2n \mid n \in \mathbb{N} \}, \{ 2n + 1 \mid n \in \mathbb{N} \} \}$.

### 5.2.2 Equivalence Classes and Graph Connectivity

The fact that equivalence relations induce a partition of the underlying set has numerous applications in mathematics. In fact, we can use this general result we've just proven to obtain an alternative proof of some of the results from Chapter Four.

If you'll recall, given an undirected graph $G = (V, E)$, a *connected component* of $G$ is a set of nodes $C$ such that

- If $x, y \in C$, then $x$ and $y$ are connected ($x \leftrightarrow y$).
- If $x \in C$ and $y \in V - C$, then $x$ and $y$ are not connected ($x \nleftrightarrow y$)

We proved in the previous chapter that every node in a graph belongs to a unique connected component in that graph. To prove this, we proceeded by proving first that every node in the graph belongs to at least one connected component, then that each node in the graph belongs to at most one connected component. However, now that we have started exploring equivalence relations, it's possible for us to prove this result as a special case of what we proved in the previous section on partitions.

Remember that the connectivity relationship $\leftrightarrow$ is an equivalence relation; it's reflexive, symmetric, and transitive. Consequently, we know that we can derive a partition $X = \{ [v]_{\leftrightarrow} \mid v \in V \}$ of the nodes in $G$ into equivalence classes based on the $\leftrightarrow$ relationship. What exactly does this partition look like? Well, let's start by thinking about what an equivalence class $[v]_{\leftrightarrow}$ looks like. This is the set $\{u \in V \mid v \leftrightarrow u\}$ of all of the nodes in the graph that are connected to $u$. In other words, the set $[v]_{\leftrightarrow}$ is the connected component containing $v$! Since we know that $X$ is a partition of the nodes in $G$, this immediately tells us that every node belongs to a unique equivalence class. Since the equivalence classes of $\leftrightarrow$ are the same as the connected components of $G$, this means that every node in $G$ belongs to a unique connected component. And we're done!

## 5.3 Order Relations

In the previous section, we explored equivalence relations, which let us group together objects that share some common property. In this section, we will explore several types of relations (collectively referred to as *order relations*) that allow us to rank different objects against one another. These relations will allow us to say that some objects are "greater" or "less" than others, to decide what objects are "better" or "worse" than one another, etc.

### 5.3.1 Strict Orders

Often, we will want to take a set of objects $A$ and rank them against one another (for example, we might want to rank movies, restaurants, etc.) To do this, we might think about defining a binary relation $R$ over the set $A$, where $xRy$ means "$x$ is not as good as $y$," for some definition of "goodness." For instance, we could take the relation $<$ over $\mathbb{N}$, where $x < y$ means "$x$ is smaller than $y$." Taking a cue from Chapter Four, we could also define the relation "$x$ is not as tasty as $y$," giving an ordering over different types of food. We could even think about defining the relation "$x$ is smaller than $y$" over different buildings. Relations that rank objects this way are called *strict orders*, and in this section we will define them and explore their properties.

As with equivalence relations, our goal in this section will be to abstract away from concrete definitions of strict orders and to try to find a small number of properties that must be held by a relation in order for it to qualify as a strict order. Let's begin by thinking about what these properties might be.

To begin with, if we are ranking objects against one another with relations like "$x < y$" or "$x$ is smaller than $y$," how do individual objects compare to themselves? Well, we will never have that $x < x$ for any choice of $x$, and no building is smaller than itself. More generally, no object will ever be strictly worse/smaller than itself. Consequently, all of these relations will be irreflexive.

These relations also have a few other interesting properties. For instance, let's suppose that $x < y$ and $y < z$. From this, we can conclude that $x < z$. Similarly, if $x$ runs faster than $y$ and $y$ runs faster than $z$, then $x$ runs faster than $z$. More generally, if we are ranking objects against one another, we will usually find that the ranking is transitive. This isn't always the case – think about the game "Rock, Paper, Scissors," in which Rock beats Scissors, Scissors beats Paper, and Paper beats Rock – but most rankings that we will encounter are indeed transitive.

Finally, and perhaps most importantly, we will never find a pair of objects that are mutually "better" than one another. For example, if $x < y$, we can guarantee that $y \not< x$. Similarly, if $x$ runs faster than $y$, we know for certain that $y$ does not run faster than $x$. In other words, these relations are asymmetric.

It turns out that these three properties capture the essential properties of relations that rank objects. Irreflexivity guarantees that no object is less than itself. Asymmetry ensures that two objects can't mutually rank higher than one another. Transitivity ensures that the ranking is consistent across all of the objects. Consequently, we use these three properties to formally define a strict order:

> A binary relation $R$ over a set $A$ is called a ***strict order*** iff $R$ is irreflexive, asymmetric, and transitive.

Given this definition, let's revisit some of the relations we've seen before to see which of them are strict orders. To begin with, let's look back at some of the operations that we have seen on sets. For example, let's take the binary relation $\subseteq$ over the set $\wp(\mathbb{N})$[*] (that is, the "is a subset of relation" over all sets of natural numbers). Is this relation a strict order? Well, to check this, we need to see if $\subseteq$ is irreflexive, asymmetric, and transitive.

Of these three properties, we already know that $\subseteq$ is transitive (this was given as an exercise in Chapter Two; if you haven't proven this yet, take a minute to do so). Is $\subseteq$ asymmetric? With a few examples, it might seem to be the case; after all:

$$\{\text{cat, dog}\} \subseteq \{\text{cat, dog, dikdik}\}, \text{ but not the other way around.}$$

$$\varnothing \subseteq \{1, 2, 3\}, \text{ but not the other way around.}$$

$$\mathbb{N} \subseteq \mathbb{R}, \text{ but not the other way around.}$$

However, $\subseteq$ is not actually asymmetric. Recall that every set is a subset of itself. This means that $A \subseteq A$. Asymmetry means that if $A \subseteq A$ holds, then after swapping $A$ and $A$, we should get that $A \subseteq A$ is false. This, of course, isn't true. Consequently, $\subseteq$ isn't asymmetric. This same line of reasoning means that $\subseteq$ isn't irreflexive either, since $A \subseteq A$ is always true. As a result, $\subseteq$ is not a strict order.

---

* You might be wondering why we consider $\subseteq$ over the set $\wp(\mathbb{N})$ and not, say, the set of all sets. It turns out that "the set of all sets" is a mathematically troublesome object. Some definitions of sets says that this set doesn't actually exist, while other definitions of sets allow the set of all sets, but only with very strict restrictions on how it can be used. We'll sidestep this by defining $\subseteq$ only over sets of naturals.

It seems like the only reason that $\subseteq$ isn't a strict order is that it's possible for a set to be a subset of itself. What happens if we disallow this? In that case, we get the $\subset$ relation, the "strict subset" relation. As a refresher, we defined $x \subset y$ to mean "$x \subseteq y$, and $x \neq y$." Let's look at this relation. Is it a strict order?

As before, let's see whether it's irreflexive, asymmetric, and transitive. First, is $\subset$ irreflexive? This means that there is no set $A$ such that $A \subset A$ holds. We can see that this is true, since if $A \subset A$ were true, it would mean $A \neq A$, which is impossible. Second, is $\subset$ asymmetric? That would mean that whenever $A \subset B$, we know that $B \subset A$ does not hold. Well, what would happen if we had both $A \subset B$ and $B \subset A$? That would mean (by definition) that $A \subseteq B$, $B \subseteq A$, but $A \neq B$. This can't happen, because if $A \subseteq B$ and $B \subseteq A$, then $A = B$. We proved this in Chapter Two. Finally, is $\subset$ transitive? That is, if $A \subset B$ and $B \subset C$, does $A \subset C$? If we expand out the definitions, we are asking whether, given that $A \subseteq B$, $B \subseteq C$, $A \neq B$, and $B \neq C$, we can determine whether $A \subseteq C$ and $A \neq C$. Of these two properties, one is easy to establish – since $\subseteq$ is transitive, we know that $A \subseteq C$ must be true. So how about $A \neq C$? One way we can show this is by contradiction. Suppose, hypothetically, that $A = C$. Since $B \subset C$ and $C = A$, this means that $B \subset A$. But that's impossible, since we know that $A \subset B$ and we just proved that $\subset$ is asymmetric. Something is wrong here, so our initial assumption that $A = C$ must have been wrong.

We can formalize the above intuition with the following proof:

***Theorem:*** The relation $\subset$ over $\wp(\mathbb{N})$ is a strict order.

***Proof:*** We show that $\subset$ is irreflexive, asymmetric, and transitive.

To show that $\subset$ is irreflexive, we must show that for any $A \in \wp(\mathbb{N})$ that $A \subset A$ does not hold. To see this, assume for the sake of contradiction that this is false and that there exists some $A \in \wp(\mathbb{N})$ such that $A \subset A$. By definition of $\subset$, this means that $A \subseteq A$ and $A \neq A$. But $A \neq A$ is impossible, since $=$ is reflexive. We have reached a contradiction, so our assumption must have been wrong. Thus $\subset$ is irreflexive.

To show that $\subset$ is asymmetric, we must show that for any $A, B \in \wp(\mathbb{N})$ that if $A \subset B$, then it is not the case that $B \subset A$. We proceed by contradiction; assume that this statement is false and that there exist some sets $A, B \in \wp(\mathbb{N})$ such that $A \subset B$ and $B \subset A$. By definition of $\subset$, this means that $A \subseteq B$, $B \subseteq A$, but $A \neq B$. Since $A \subseteq B$ and $B \subseteq A$, we know that $A = B$, contradicting the fact that $A \neq B$. We have reached a contradiction, so our assumption must have been wrong. Thus $\subset$ is asymmetric.

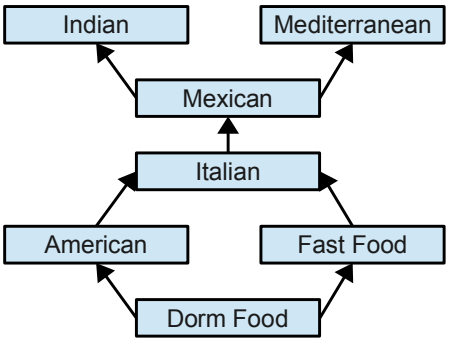
To show that $\subset$ is transitive, we need to show that for any $A, B, C \in \wp(\mathbb{N})$, that if $A \subset B$ and $B \subset C$, then $A \subset C$. Consider any $A, B, C \in \wp(\mathbb{N})$ where $A \subset B$ and $B \subset C$. By definition of $\subset$, this means that $A \subseteq B$, $B \subseteq C$, $A \neq B$, and $B \neq C$. We will show that $A \subset C$, meaning that $A \subseteq C$ and $A \neq C$. Since $A \subseteq B$ and $B \subseteq C$, we know that $A \subseteq C$. To show that $A \neq C$, assume for the sake of contradiction that $A = C$. Since $B \subset C$ and $C = A$, this means that $B \subset A$. But this is impossible, because we also know that $A \subset B$, and $\subset$ is asymmetric. We have reached a contradiction, so our assumption must have been wrong. Thus $A \neq C$. Since $A \subseteq C$ and $A \neq C$, this means that $A \subset C$ as required.

Since $\subset$ is irreflexive, asymmetric, and transitive, it is a strict order over $\wp(\mathbb{N})$. ■

Before we conclude this section, there is one last detail we should explore. We just proved that $\subset$ is a strict order over sets of natural numbers. Notice, however, that not all sets of natural numbers can even be com-

pared by $\subset$. For example, if we take $A = \{1, 2, 3\}$ and $B = \{3, 4, 5\}$, then neither $A \subset B$ nor $B \subset A$ is true. If we are trying to use $\subset$ to rank different sets, we will find that neither $A$ nor $B$ would be considered "greater" or "lesser" than the other.

Many other strict orders cannot be used to establish a total ranking of the elements of the underlying set. For example, consider the relation "$x$ is tastier than $y$" over different types of food. In the previous chapter, I drew a DAG of my own personal food preferences, which looked like this:

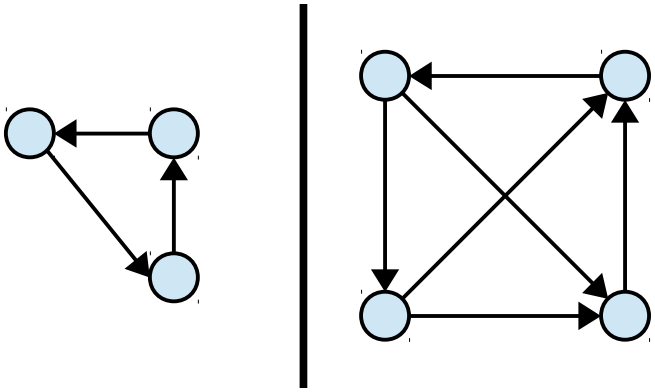



Notice that Indian and Mediterranean cuisine are incomparable. I really like both of them a lot, but I wouldn't say that one of them was necessarily better than the other. They're both really, really good. However, I would say with certainty that they are much better than dorm food!

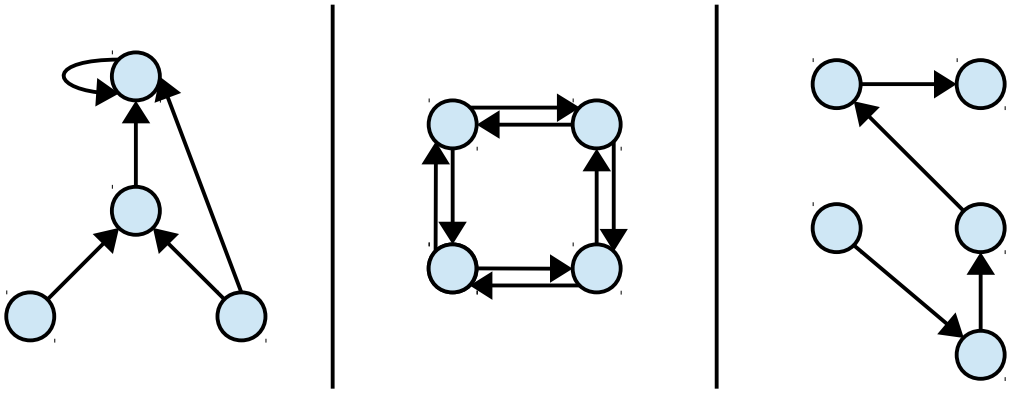
Both of the $\subset$ and "tastier than" relations are strict orders, but neither one of them is guaranteed to rank all elements of the underlying set against one another. On the other hand, some strict orders can rank all objects against one another. For example, the $<$ relation over $\mathbb{N}$ guarantees that if we pick any $x, y \in \mathbb{N}$ with $x \neq y$, we will find either that $x < y$ or $y < x$. This suggests that the label "strict order" actually encompasses several different types of order relations. Some, like $\subset$ and "tastier than" rank their elements, but might have many elements incomparable with one another. Others, like $<$, rank all underlying elements. To distinguish between the two, we will introduce two new definitions. First, we will need a way of formalizing the idea that any two distinct elements are comparable. This is called *trichotomy*:

> A binary relation $R$ over a set $A$ is called ***trichotomous*** iff for any $x, y \in A$, exactly one of the following holds: $xRy$, or $yRx$, or $x = y$.

Calling back to the graphical intuition for binary relations, a trichotomous relation is one where, for any pair of objects, there is exactly one edge running between them. For example, the following relations are trichotomous:



While the following are not:



Given this new definition, we can formally make a distinction between strict orders that can rank all elements against one another versus strict orders that cannot.

> A binary relation $R$ over a set $A$ is called a ***strict total order*** iff $R$ is a strict order and $R$ is trichotomous.

Strict total orders induce a linear ranking on their elements. We can, in theory, line up all of the elements of a set ordered by a strict total order in a way where if $xRy$, then $x$ appears to the left of $y$. For example, if we do this to the natural numbers ordered by $<$, we get the normal ordering of the naturals:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, \ldots$$

To summarize this section: strict orders are orders like $<$, $\subset$, or "tastier than" that allow us to rank objects against one another. Strict total orders are orders like $<$ that completely rank the elements of the underlying set.

## 5.3.2 Partial Orders

In the previous section, we explored strict orders. These relations are called "strict" orders because the relation usually has the form "$x$ is strictly less than $y$," "$x$ is strictly better than $y$," "$x$ is a strict subset of $y$," etc. Such orders give us a way to talk about relations like $<$, $\subset$, etc. However, strict orders don't give us a nice way to talk about relations like $\leq$ or $\subseteq$, which still rank objects against one another, but are slightly more forgiving than $<$ or $\subset$. Whereas the relation $<$ is "$x$ is strictly less than $y$," the relation $\leq$ is "$x$ is not greater than $y$." Whereas "$x$ is taller than $y$" is a strict order, we could also consider the relation "$x$ is not smaller than $y$," which can also be used to rank elements.

Relations like $\leq$ and $\subseteq$ are called *partial orders*. They order the elements of a set, just as strict orders do, but do so by saying that objects are "no bigger than" other objects, rather than objects are "smaller than" other objects.

As with equivalence relations and strict orders, we will define partial orders by looking for properties shared by all partial orders, then distilling the definition down to those key traits. Considering that partial orders are in a sense a "softer" version of strict orders, it might be useful to see what properties of strict orders carry over to partial orders. If you'll recall, the three properties required of strict orders are irreflexivity, asymmetry, and transitivity. How many of these properties still hold for relations like $\leq$, $\subseteq$, and "$x$ is no taller than $y$?"

Immediately, we can see that none of these relations are irreflexive. Any number $x$ satisfies $x \leq x$, any set $A$ satisfies $A \subseteq A$, and any building $b$ satisfies "$b$ is no taller than $b$." While strict orders are *irreflexive*, partial

orders are all *reflexive*, since every object should be no greater than itself, no less than itself, no bigger than itself, no greener than itself, etc.

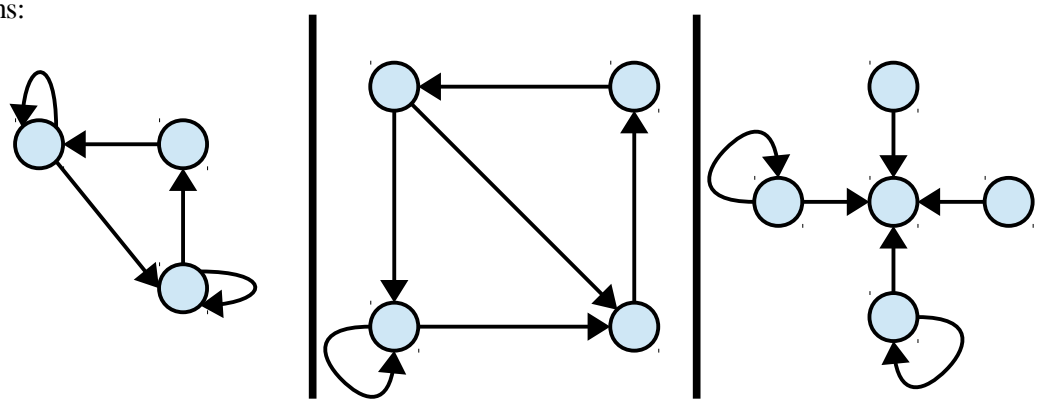
How about asymmetry? Here, the picture is more complicated. While in many cases ≤ acts asymmetrically (for example, 42 ≤ 137, but it is not the case that 137 ≤ 42), the relation ≤ is not asymmetric. Take, for example, 137 ≤ 137, which is true. If ≤ were asymmetric, we would have to have that, since 137 ≤ 137 is true, then after exchanging the order of the values, we should have that 137 ≤ 137 does not hold. Of course, 137 ≤ 137 does hold. This single counterexample means that ≤ is not an asymmetric relation. This same reasoning lets us see that ⊆ is not asymmetric, because any set is a subset of itself, and that "$x$ is no taller than $y$" is not asymmetric either.

That said, these relations are "mostly" asymmetric. If we pick any $x \neq y$, then if $x \leq y$, we're guaranteed that $y \leq x$ won't be true. Similarly, if $A \neq B$ and $A \subseteq B$, then we can be sure that $B \subseteq A$ isn't true. In a sense, these relations are mostly asymmetric, in that they behave like asymmetric relations as long as the two objects being compared aren't identically the same. Relations with this property are extremely common, and so we give a name to this property: *antisymmetry*:

> A binary relation $R$ over a set $A$ is called ***antisymmetric*** iff for any $x$, $y \in A$, if $x \neq y$, then if $xRy$, we have $y\not{R}x$.
>
> Equivalently, a binary relation $R$ over a set $A$ is antisymmetric iff for any $x$, $y \in A$, if $xRy$ and $yRx$, then $x = y$.

I've listed two different ways of defining antisymmetry above. The first definition more closely matches the intuition we built in the above section – namely, a relation is antisymmetric if for any pair of unequal values, the relation acts asymmetrically. The second definition is a different way of thinking about this property. It says that if you can find two objects that each are no greater than one another, we can guarantee that those objects must be one and the same. For example, if $x \leq y$ and $y \leq x$, it's safe to conclude that $x = y$. Similarly, if $A \subseteq B$ and $B \subseteq A$, then $A = B$. At times, one of these definitions will be easier to work with than the other, so it's good to know both of these definitions.

Graphically, what does an antisymmetric relation look like? In many ways, it's similar to an asymmetric relationship. If you pick any pair of distinct nodes in the graph, there can be at most one edge between them. However, unlike an asymmetric relation, it's fine for there to be edges from nodes to themselves. This would indicate, for example, something like "$x$ is no bigger than itself." Below are some examples of antisymmetric relations:



An important detail – just as symmetry and asymmetry are not opposites of one another, symmetry and antisymmetry are not opposites, or are asymmetry and antisymmetry. In fact, every asymmetric relation is also antisymmetric, and (as you'll see in the chapter exercises) it's possible for a relation to be symmetric and antisymmetric at the same time. If you want to prove that a relation is antisymmetric, you will need to explicitly prove that it satisfies the definition given above.

Great! At this point, we've seen that the three relations we've picked as representatives of partial orders are both reflexive and antisymmetric. There is one last property that we haven't yet investigated – transitivity. As we saw when discussing strict orders, all strict orders are transitive. Does the same hold true for partial orders? All three of the sample partial orders we've seen end up being transitive:

If $x \leq y$ and $y \leq z$, then $x \leq z$.

If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$.

If $x$ is no taller than $y$ and $y$ is no taller than $z$, then $x$ is no taller than $z$.

More generally, transitivity is an important property that we want to have of partial orders. Just as with strict orders, we want to ensure that our rankings are consistent across objects.

The three traits we've identified – reflexivity, antisymmetry, and transitivity – end up being the essential traits of partial orders, and in fact this is how we will define them.

> A binary relation $R$ over a set $A$ is called a ***partial order*** iff $R$ is reflexive, antisymmetric, and transitive.

Partial orders are extremely common in discrete mathematics and computer science. It is very common to speak of "partially ordered sets," sets with an associated partial order. In fact, this terminology is so important that we should draw a light yellow box around it:

> A ***partially ordered set*** (or ***poset***) is a pair $(A, R)$ where $A$ is a set and $R$ is a partial order over $A$.

Partial orders and strict orders collectively are called order relations:

> A binary relation $R$ over a set $A$ is called an ***order relation*** iff $R$ is a partial order or $R$ is a strict order.

Although partial and strict orders have different properties, they behave similarly in many contexts. As a result, most of the results from the latter part of this chapter that deal with partial orders or strict orders will work with either type of order interchangably. Consequently, we will discuss properties of order relations in general, mentioning specifically that we are working with partial or strict orders only in the case where a certain property applies to only one type of order.

To give a more elaborate example of a partial order, let's consider the divides relation $|$. Recall that we write $m \mid n$ to mean "$m$ divides $n$." When we restrict ourselves to talking about the natural numbers, we formally defined the $|$ relation as follows: $m \mid n$ iff there exists some $q \in \mathbb{N}$ such that $n = mq$. It turns out that this relation defines a partial order over the set $\mathbb{N}$. To formally prove this, we will need to show that $|$ is reflexive, antisymmetric, and transitive. Let's consider each in turn.

First, we need to show that | is reflexive, meaning that for any $n \in \mathbb{N}$, that $n \mid n$. Intuitively, it should be clear that every number divides itself. Formally, we can show this by noting that $n = 1 \cdot n$, meaning that there is some choice of $q$ (namely, 1) where $n = qn$.

Next, we need to show that | is antisymmetric over $\mathbb{N}$. This is a bit trickier than it might seem. If we don't restrict ourselves to natural numbers and instead consider divisibility over the set of all integers $\mathbb{Z}$, then | is *not* antisymmetric. For example, $-2 \mid 2$ and $2 \mid -2$, but $2 \neq -2$. In order to show that | is antisymmetric, we're going to need to be clever with how we structure our argument.

The approach we will take is the following. Let's suppose that $m, n \in \mathbb{N}$, that $m \neq n$, and that $m \mid n$. This means that there is some $q \in \mathbb{N}$ such that $n = mq$. We want to show that $n \mid m$ is false, meaning that there is no $r$ such that $m = nr$. To do this, let's see what happens if such an $r$ were to exist. Since $n = mq$ and $m = nr$, we get that

$$n = mq = (nr)q = nrq$$

$$m = nr = (mq)r = mrq$$

So $n = nqr$ and $m = mqr$. Now, we know that $m$ and $n$ cannot simultaneously be zero, because $m \neq n$. So that means that for at least one of these equalities, we can divide through by $m$ or $n$ safely, getting tha $1 = qr$. Since $q$ and $r$ are natural numbers, the only way that this is possible is if $q = r = 1$. But if this happens, then $n = mq = m$, meaning that $m = n$, a contradiction.

This is a fairly long-winded argument, but such is the nature of this proof. What we are doing is very specific to the fact that we're working with natural numbers and how we have defined divisibility. As an exercise, try looking over this proof and see what goes wrong if we now allow $m$ and $n$ to be integers rather than natural numbers.

We've established that, over $\mathbb{N}$, | is reflexive and antisymmetric. So how do we show that it's transitive? Well, suppose that $m \mid n$ and $n \mid p$. This means that there exists natural numbers $q$ and $r$ such that $n = mq$ and $p = rn$. Combining these, we get that $p = qrm = (qr)m$. Thus there is some natural number $k$, namely $qr$, such that $p = km$. Therefore $m \mid p$.

We can formalize this reasoning in the following proof:

***Theorem:*** $\mid$ is a partial order over $\mathbb{N}$.

***Proof:*** We will show that $\mid$ is reflexive, antisymmetric, and transitive. To see that $\mid$ is reflexive, we will prove that for any $n \in \mathbb{N}$, that $n \mid n$ (that there exists some $q \in \mathbb{N}$ such that $n = nq$). So let $n$ be any natural number, and take $q = 1$. Then $nq = n \cdot 1 = n$, so $n \mid n$.

To see that $\mid$ is antisymmetric, we will prove that for any $m, n \in \mathbb{N}$, that if $m \mid n$ and $n \mid m$, that $m = n$. Consider any $m, n \in \mathbb{N}$ where $m \mid n$ and $n \mid m$. This means that there exists $q, r \in \mathbb{N}$ such that $n = mq$ and $m = nr$. Consequently:

$$m = nr = (mq)r = mqr$$
$$n = mq = (nr)q = nqr$$

We now consider two cases. First, if $m = n = 0$, then we are done, since $m = n$. Otherwise, at least one of $m$ or $n$ is nonzero; without loss of generality, assume $m \neq 0$. Then since $m = mqr$, we know that $1 = qr$. Since $q, r \in \mathbb{N}$, this is only possible if $q = r = 1$. Consequently, $m = nr = n \cdot 1 = n$, so $m = n$, as required.

To see that $\mid$ is transitive, we will prove that for any $m, n, p \in \mathbb{N}$, that if $m \mid n$ and $n \mid p$, then $m \mid p$. Consider any $m, n, p \in \mathbb{N}$ where $m \mid n$ and $n \mid p$; then there must exist $q, r \in \mathbb{N}$ such that $n = qm$ and $p = rn$. Consequently, $p = rn = r(qm) = qrm = (qr)m$. Since $qr \in \mathbb{N}$, this means that there is some $k \in \mathbb{N}$ (namely, $qr$) such that $p = km$. Thus $m \mid p$, as required.

Since $\mid$ is reflexive, antisymmetric, and transitive over $\mathbb{N}$, $\mid$ is a partial order over $\mathbb{N}$.

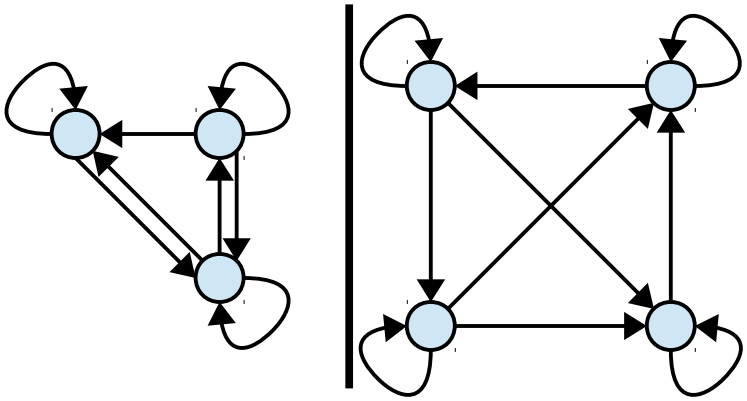
To round out this section, let us draw one further parallel with strict orders. If you'll recall, we noted that some strict orders, like $<$, could be used to rank any arbitrary pair of values. Other strict orders, like $\subset$, could not necessarily rank all values against one another. This led us to draw a distinction between strict orders and strict *total* orders.

This same distinction exists when discussing partial orders. For example, $\leq$, when applied to real or rational numbers, can rank any pair of values. However, $\mid$ cannot; for example, both $3 \mid 5$ and $5 \mid 3$ are false. This motivates a refinement of partial orders into two subgroups - "plain old" partial orders, which are just reflexive, symmetric, and transitive, and "total" partial orders that can always rank elements against one another.

To do this, we will introduce one more definition:

A binary relation $R$ over a set $A$ is called ***total*** iff for any $x, y \in A$, at least one of $xRy$ and $yRx$ is true.

In other words, a total order is one in which any pair of values is guaranteed to be related somehow. Graphically, total relations are relations where any pair of nodes necessarily must have an edge between them. Additionally, in a total relation, all nodes must have edges to themselves, since by the above definition $xRx$ has to hold for all $x \in A$. Consequently, the following graphs represent total relations:



Given this definition, we can now differentiate between partial orders and a stronger class of ordering relations called *total orders*:

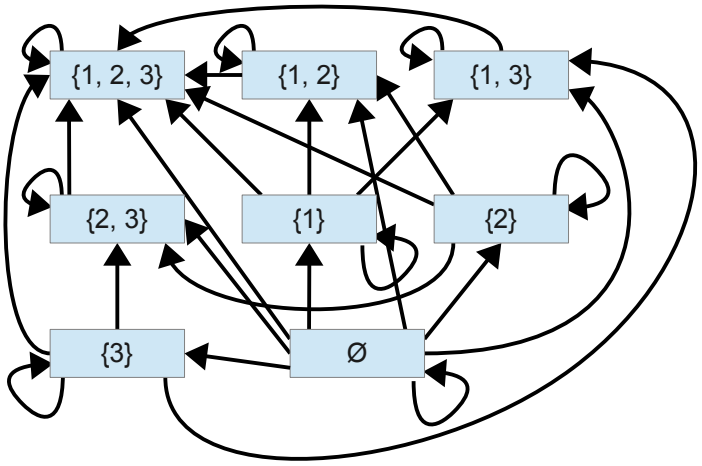
> A binary relation $R$ over a set $A$ is called a ***total order*** iff $R$ is total and $R$ is a partial order.

Total orders (or strict total orders) are the types of relations you would want to use when trying to sort a list of values. They guarantee that there is a "correct" way to sort the list – put the least value first, then the next biggest, then the next biggest, etc. This is only possible if we have a (strict) total order over the elements, since otherwise we might have a situation in which no one value is smallest or largest. The chapter exercises ask you to play around with this to verify why it is correct.

### 5.3.3 Hasse Diagrams

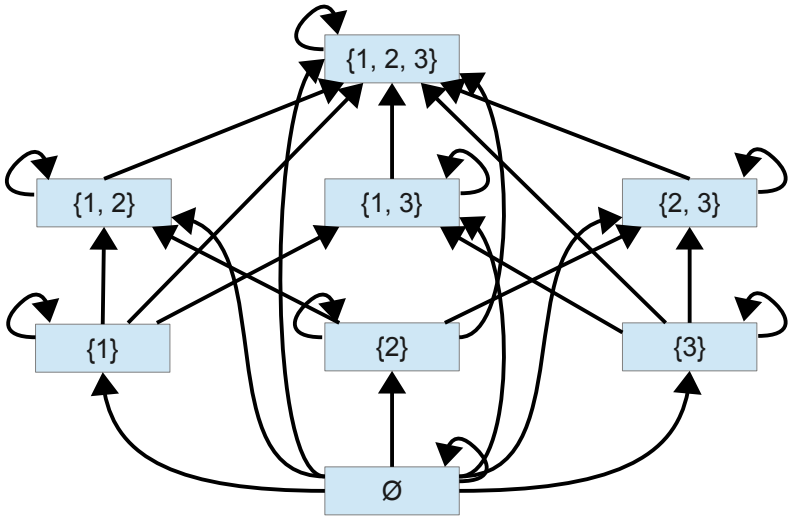
Throughout this chapter, we've intermittently switched to a graphical view of relations to highlight some of the important properties of different types of relations. We have just finished discussing strict orders and partial orders, but haven't yet visualized what they look like. This section explores how to draw strict and partial orders.

Let's begin with a simple partial order: the $\subseteq$ relation defined over $\wp(\{1, 2, 3\})$. This is the subset relation over sets containing the elements 1, 2, and 3. If we draw out the graph of this relation, we get the following:

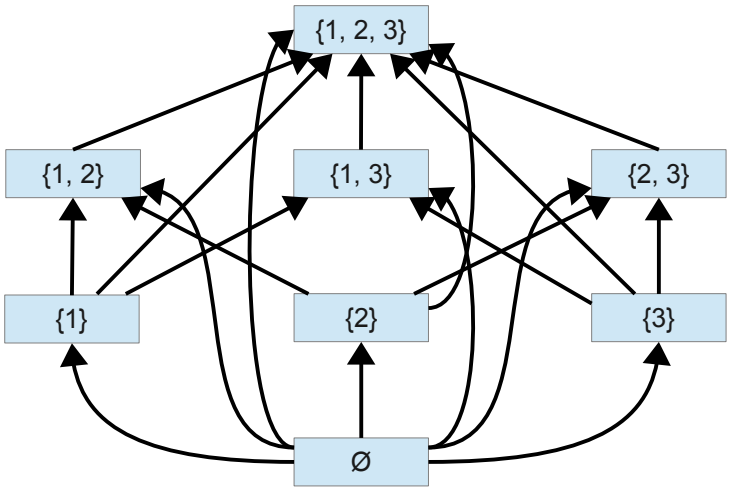


Yikes! That's almost impossible to read! Is there some way that we might simplify this?

First, let's return to our original intuition about partial orders. We started exploring these relations in order to be able to rank elements in some set against one another. Accordingly, let's try redrawing that above graph so that we put “bigger” elements up at the top and “smaller” elements down at the bottom. If we do that, we get this picture:

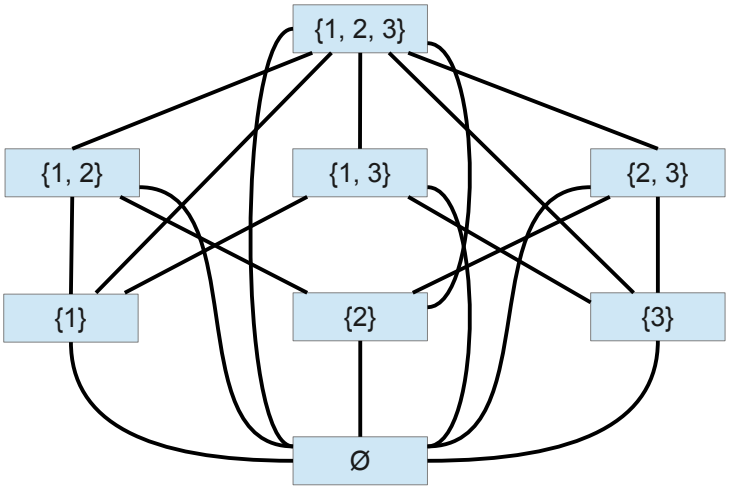



This is slightly easier to read than before, but it's still pretty crowded. For starters, note that every node in this graph has a loop to itself. This is because ⊆, like all partial orders, is reflexive, and graphs of reflexive relations always have every node containing an edge to itself. Of course, *all* partial orders are reflexive. If we are trying to draw out a partial order, we could save some time and effort by simply omitting these self-loops. We know that they're supposed to be there, but adding in those self-loops just makes the drawing a bit harder to read. If we eliminate those loops, we get this drawing, which is slightly easier to read than before:



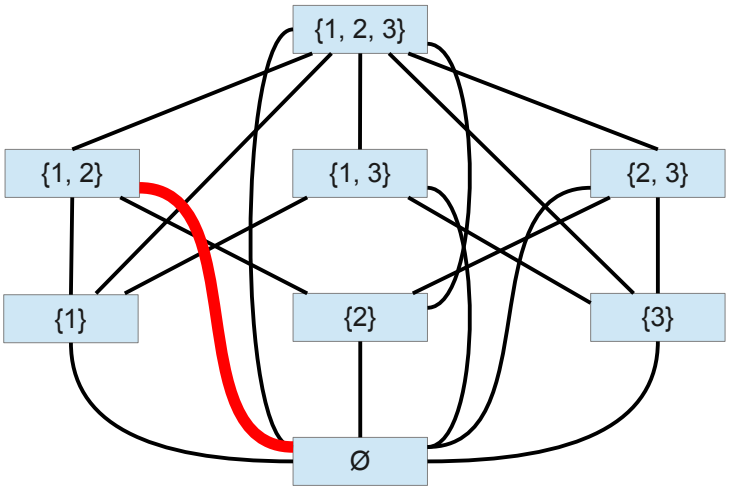

Let's see if we can further simplify this picture. Notice right now that all of the edges we've drawn are directed edges. This makes sense, since we want to be able to tell, for any pair of values, which value (if any) is less than the other. However, remember that ⊆ is antisymmetric. This means that if we have any two distinct values, we can't ever have edges running between them in opposite directions; there will either be an

edge running in one direction, or no edges at all. Since we've drawn this picture by putting larger elements up at the top and smaller elements at the bottom, we can erase all of the arrows from this picture without encountering any problems. We'll just implicitly assume that the arrows always point upward. This means that we don't need to draw out lots of clunky arrowheads everywhere. If we do this, we get the following picture:

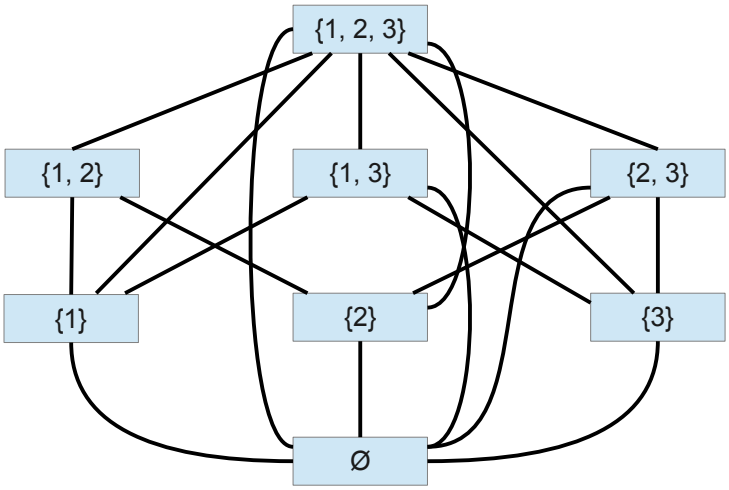



At this point, the structure of the partial order is starting to get a bit easier to read, but it's still pretty cluttered. Let's see if we can further simplify this structure.

One thing that stands out in this picture is that the empty set Ø is a subset of all of the other sets. As a result, there are seven edges emanating from it. Do we really need to draw all of these edges? For example, consider this highlighted edge:

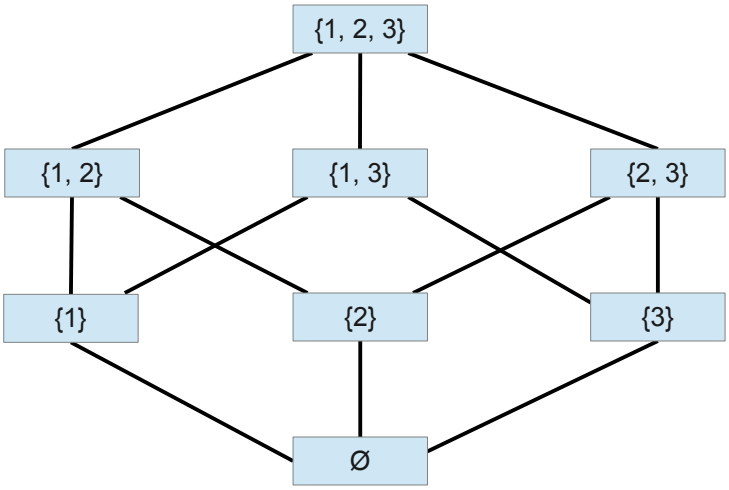



Notice that there's already an edge from Ø to {1} and from {1} to {1, 2}. Since ⊆, like all partial orders, is transitive, the fact that these edges exist automatically tells us that there has to be an edge from Ø to {1, 2}. Consequently, if we were to delete the edge from Ø to {1, 2} in the above drawing, we could still recover the fact that Ø ⊆ {1, 2} by noting that there is still an upward path between those two nodes. This gives the following picture:




More generally, let's suppose that we have three edges $(x, y)$, $(y, z)$, and $(x, z)$. In this case, we can always safely remove $(x, z)$ from the drawing without hiding the fact that $x$ compares no greater than $z$. The reason for this is that, by transitivity, the existence of the edges $(x, y)$ and $(y, z)$ is sufficient for us to recover the original edge.

If we delete all of these redundant edges from the above drawing, we get the following drawing, which has no more redundancies:




This drawing is the cleanest way of representing the relation $\subseteq$ over these sets. We have used our knowledge that $\subseteq$ is reflexive to eliminate self-loops, that $\subseteq$ is antisymmetic to eliminate the arrowheads, and that $\subseteq$ is transitive to eliminate extraneous edges. Given just this graph, we can present all of the information necessary to understand $\subseteq$ in a compact, readable form.

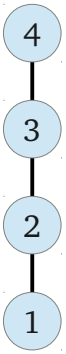
Interestingly, we could have also drawn out a schematic of the strict order $\subset$ this way. Since $\subset$ is irreflexive, we'd have no self-loops to eliminate in the first place. Since $\subset$ is asymmetric, we could similarly eliminate the directions of the arrows and implicitly assume that they're directed upward. Since $\subset$ is transitive, we could still eliminate redundant edges in the graph.

The diagram we have just made is our first example of a *Hasse diagram*, a graphical representation of an order relation. Formally, a Hasse diagram is defined as follows:

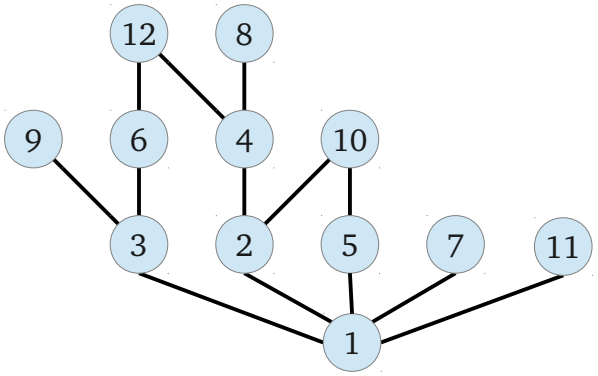
Given a strict or partial order relation $R$ over a set $A$, a ***Hasse diagram*** for $R$ is a drawing of the elements of $A$ with the following properties:

1. All edges are undirected.
2. If there is an edge from $x$ to $y$ and $x$ is below $y$ in the diagram, then $xRy$.
3. There are no self-loops.
4. There are no redundant edges: if there is an edge from $x$ to $y$ and an edge from $y$ to $z$, then there is no edge from $x$ to $z$.

For example, here is a Hasse diagram for the < relation over the set {1, 2, 3, 4}:




And here's a Hasse diagram for | over the natural numbers between 1 and 12:




If we look at these Hasse diagrams, we can start to notice certain differences between the structures of these order relations. For example, look at the Hasse diagram for ⊆. If you'll notice, the set {1, 2, 3} is clearly the "biggest" element, while Ø is clearly the "smallest." Similarly, in the diagram for <, 1 is clearly the "smallest" element, while 4 is clearly the "biggest." On the other hand, if we look at the Hasse diagram for divisibility over the numbers between 1 and 12, there is no clear "biggest" or "smallest" elements. For example, 8, 9, 10, 11, and 12 aren't connected to anything above them, but no single one of them is the "biggest" value in that every other value in the set divides it. However, the value 1 is indeed the "smallest" value here.

To formalize our terminology, let's formally specify what's meant by "biggest" and "smallest" values.

Let $R$ be a partial order over $A$. An element $x \in A$ is called the ***greatest element of A*** iff for all $y \in A$, $yRx$. An element $x \in A$ is called the ***least element of A*** iff for all $y \in A$, $xRy$.

Let $R$ be a strict order over $A$. An element $x \in A$ is called the ***greatest element of A*** iff for all $y \in A$ where $y \neq x$, $yRx$. An element $x \in A$ is called the ***least element of A*** iff for all $y \in A$ where $y \neq x$, $xRy$.

For example, $\varnothing$ is the least element of the partial order defined by $\subseteq$ over $\wp(\{1, 2, 3\})$, while $\{1, 2, 3\}$ is the greatest. There is no greatest element of the naturals between 1 and 12 when ordered by divisibility, but 1 is the least element.

An important observation is that greatest and least elements of some ordered set $A$, if they even exist, are guaranteed to be unique. That is, no set can have two greatest elements or two least elements. We can prove this below for partially-ordered sets; the proof for strictly-ordered sets is similar and is left as an exercise.

***Theorem:*** Let $R$ be a partial order over set $A$. Then if $R$ contains a greatest element, it contains exactly one greatest element.

***Proof:*** Let $R$ be a partial order over $A$, and assume that $g \in A$ is the greatest element of $A$. We will show that there are no other greatest elements of $A$. To do so, we proceed by contradiction; assume that there is some other greatest element $h \in A$ with $g \neq h$. Since $g$ is a greatest element of $A$, we have that $hRg$. Since $h$ is a greatest element of $A$, we have that $gRh$. Since $R$ is a partial order, it is antisymmetric, and so $g = h$. This contradicts the fact that $g \neq h$. We have reached a contradiction, so our assumption must have been wrong. Thus if $A$ has a greatest element, it has a unique greatest element. ■

The fact that there is a unique greatest element (if one even exists in the first place) allows us to talk about *the* greatest element of an ordered set, rather than just *a* greatest element. We can similarly speak about *the* least element, rather than *a* least element.

We can generalize our discussion of greatest and least elements in an entire set to greatest and least elements of subsets of that set. For example, although the naturals between 1 and 15 don't have a greatest element, if we look at just the subset $\{1, 2, 3, 4, 6, 12\}$, then there is a greatest element, namely 12. Consequently, let's extend our definitions from before to let them work with any collection of values from an ordered set:

Let $R$ be a partial order over $A$ and $S \subseteq A$ be a set of elements from $A$. An element $x \in S$ is called the ***greatest element of S*** iff for all $y \in S$, $yRx$. An element $x \in S$ is called the ***least element of S*** iff for all $y \in S$, $xRy$.

Let $R$ be a strict order over $A$ and $S \subseteq A$ be a set of elements from $A$. An element $x \in S$ is called the ***greatest element of A*** iff for all $y \in S$ where $y \neq x$, $yRx$. An element $x \in S$ is called the ***least element of A*** iff for all $y \in S$ where $y \neq x$, $xRy$.

To round out this section, let's quickly revisit the Hasse diagram for divisibility. Notice that the values {8, 9, 10, 11, 12} are all at the top of the diagram. None of these numbers divide any other numbers in the range from 1 to 12. However, none of these values are the greatest value of the partial order, because none of them divide one another. Although these values aren't the *greatest* values of the partial order, we can say that, in some sense, they must be fairly large, since they aren't smaller than anything else. This motivates the following definition:

> Let $R$ be a partial order over $A$ and $S \subseteq A$ be a set of elements from $A$. An element $x \in S$ is called a ***maximal element of S*** iff for all $y \in S$, $x \not R y$. An element $x \in S$ is called a ***minimal element of S*** iff for all $y \in S$, $y \not R x$.
>
> Let $R$ be a strict order over $A$ and $S \subseteq A$ be a set of elements from $A$. An element $x \in S$ is called the ***maximal element of A*** iff for all $y \in S$ where $y \neq x$, $x \not R y$. An element $x \in S$ is called the ***minimal element of A*** iff for all $y \in S$ where $y \neq x$, $y \not R x$.

In other words, a *maximal* element is one that isn't smaller than anything else, and a minimal element is an element that isn't greater than anything else. All greatest and least elements are maximal and minimal elements, respectively, but maximal and minimal elements aren't necessarily greatest and least elements. Although a set can have at most one greatest and least element, sets can have any number of maximal and minimal elements.

### 5.3.4 Preorders ★

Before concluding this section on order relations, it is important to note one important difference between partial orders and strict orders. Suppose that you have a group of students $S$ and want to define an order relation over them based on how many days per week they exercise. Consider the following two relations:

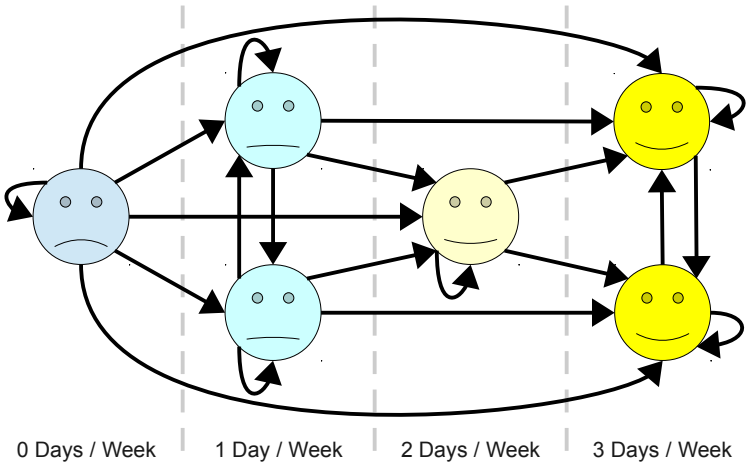
$R_1$, where $x R_1 y$ iff $x$ exercises fewer days per week than $y$.

$R_2$, where $x R_2 y$ iff $x$ exercises no more days per week than $y$.

Are either of these relations partial orders or strict orders? Let's look at each in turn, starting with $R_1$. Without formally proving it, we can see that $R_1$ should be irreflexive, since no one can exercise fewer days per week than herself. $R_1$ is also asymmetric, since if one person exercises fewer days each week than another person, the second person can't possible exercise fewer days per week than the first. Finally, $R_1$ is transitive, since if person $x$ exercises less frequently than person $y$, who in turn exercises less frequently than person $z$, then person $x$ also exercises less frequently than person $z$. Accordingly, $R_1$ is a strict order.

So what about $R_2$? Well, we can see that $R_2$ is reflexive, since everyone exercises no more than themselves. $R_2$ is also transitive. At this point, $R_2$ looks like a promising candidate for a partial order; if we can show that it's antisymmetric, then $R_2$ would indeed be a partial order. However, $R_2$ is *not* antisymmetric. To see this, consider two people $x$ and $y$ that both exercise for the same number of days each week. In that case, we have that $x R_2 y$ and $y R_2 x$, since each exercises no more days than the other. However, it is *not* the case that $x = y$. $x$ and $y$ are different people.

If we were to draw out what this relation looks like (drawing out the full relation, not the Hasse diagram), we get the following:




This points to a key distinction between strict orders and partial orders. A group of people can easily be ranked by some trait using a strict order, but not necessarily by a partial order. Given a group of distinct objects, if we strictly rank the objects by some trait, we (usually) get back a strict order. If we rank the objects with a relation like "$x$'s trait is no greater than $y$'s trait," we often don't get back an antisymmetric relation, and hence don't get back a partial order.

The issue here lies with the definition of antisymmetry – namely, that if $xRy$ and $yRx$, then $x = y$. However, in this case antisymmetry is too strong a claim. If two people exercise no more than each other, it doesn't mean that they're the same person. It just means that they exercise the same amount.

While the relation $R_2$ is not a partial order, we still have that it's reflexive and transitive. In this way, it's similar to a partial order, but the fact that it's not antisymmetric means that it just quite isn't one. Fortunately, we do have a term for a relation like this: we call it a *preorder*.*

> A binary relation $R$ over a set $A$ is called a ***preorder*** iff it is reflexive and transitive.

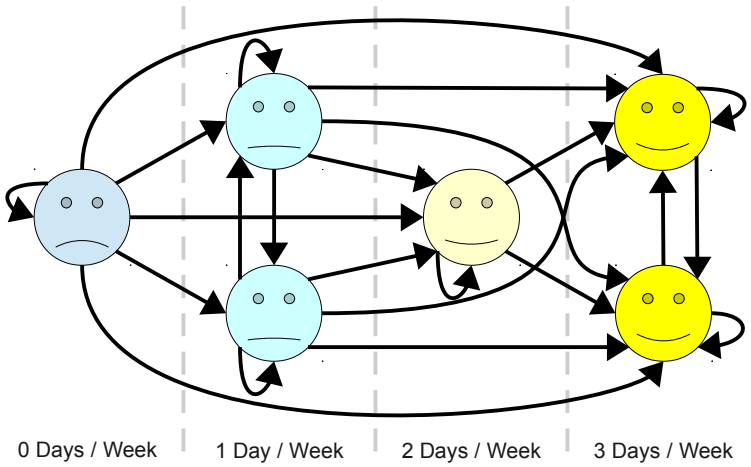
### 5.3.4.1 Properties of Preorders

Why are these relations called preorders?

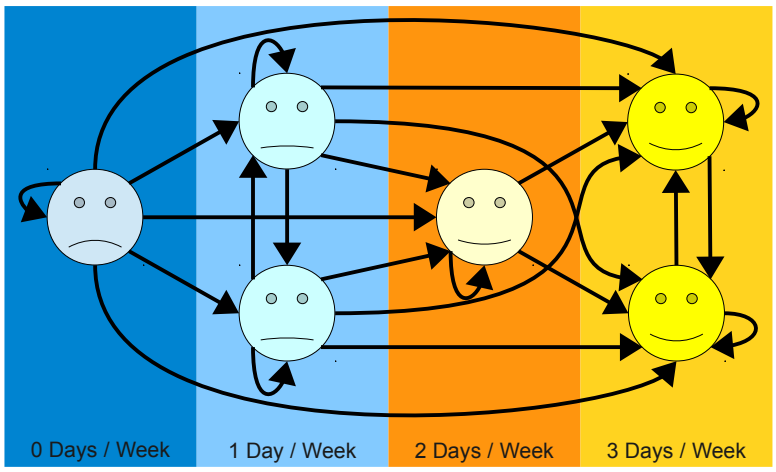
Preorders are closely connected to partial orders and equivalence relations in a variety of ways. Superficially, any equivalence relation is a preorder and any partial order is a preorder, since both of these types of relations are reflexive and transitive.

At a deeper level, though, it's possible to take any preorder and from it derive an important equivalence relation and partial order. To motivate this section, let's review the diagram of the preorder $R_2$ ("$x$ exercises no more than $y$") that we saw on the previous page:

* No, it's not something you pay for online before it comes out.




This relation is not antisymmetric because, as you can see, there are many pairs of people that are related in both directions by $R_2$. Let's investigate this further. What happens if we group together all people $x$ and $y$ where both $xR_2y$ and $yR_2x$? That is, we group people together such that neither person exercises any more than the other. If we do this, then we see the following:




Now that's interesting – this way of dividing up the people being related ends up partitioning all of the people in the graph! We've seen this behavior before when we investigated equivalence relations. Every equivalence relation induces a partition, and any partition induces an equivalence relation. Intuitively, this makes sense – the relation "$x$ and $y$ exercise the same amount" has the feel of an equivalence relation.

Did we just coincidentally stumble upon an equivalence relation that we can build out of the preorder $R_2$? Or is this a deeper result? It turns out that it's the latter case: it's always possible to take a preorder and extract an equivalence relation from it. Let's go and explore why this is.

To begin with, let's generalize what we just did in the previous section. Let's suppose that we have an arbitrary preorder $R$. From this, we can define a new binary relation $\sim_R$ as follows:

$$x \sim_R y \quad \text{iff} \quad xRy \text{ and } yRx$$

In the above case of the preorder "$x$ exercises no more than $y$," the relation $\sim_{R2}$ that we would have found would be the relation "$x$ exercises the same amount as $y$," which is an equivalence relation. If we took the preorder $x \leq y$ (it's also a partial order, but all partial orders are preorders) and considered $\sim_{\leq}$, we'd get the relationship "$x \leq y$ and $y \leq x$;" that is, the relation $x = y$. This is also an equivalence relation.

So why is this? Well, given a preorder $R$ over a set $A$, we know that $R$ is reflexive and transitive. From this, let's see if we can prove that $\sim_R$ is reflexive, symmetric, and transitive, the three properties necessary to show that a relation is an equivalence relation. We can sketch out a proof of these properties below:

- **Reflexivity**: We need to show that for any $x \in A$ that $x \sim_R x$. This means that we have to show that $xRx$ and $xRx$. Since $R$ is reflexive, this is true.
- **Symmetry**: We need to show that for any $x$, $y \in A$, that if $x \sim_R y$, then $y \sim_R x$. Well, if $x \sim_R y$, then $xRy$ and $yRx$ (that's just the definition of $\sim_R$). Simply reordering those statements gives us that $yRx$ and $xRy$. Therefore, $y \sim_R x$.
- **Transitivity:** We need to show that for any $x$, $y$, $z \in A$, that if $x \sim_R y$ and $y \sim_R z$, then $x \sim_R z$. Expanding out the definition of $\sim_R$, this means that if $xRy$ and $yRx$, and if $yRz$ and $xRy$, then we need to show that $xRz$ and $zRx$. Since $R$ is transitive (it's a preorder), this immediately follows.

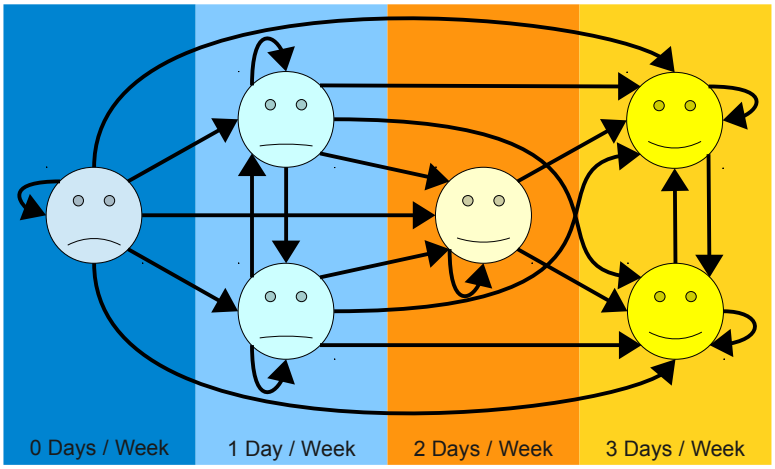
We can formalize this proof here:

> ***Theorem:*** If $R$ is a preorder over $A$, then $\sim_R$ is an equivalence relation over $A$.

> ***Proof:*** Let $R$ be any preorder over a set $A$ and define $x \sim_R y$ iff $xRy$ and $yRx$. We will prove that $\sim_R$ is reflexive, symmetric, and transitive.
>
> To see that $R$ is reflexive, consider any $x \in A$; we will prove that $xRx$. Since $R$ is a preorder, $R$ is reflexive, so $xRx$. Since $xRx$, it's also true that $xRx$ and $xRx$. Consequently, $x \sim_R x$.
>
> To see that $R$ is symmetric, consider any $x$, $y \in A$ such that $x \sim_R y$. We will prove that $y \sim_R x$. Since $x \sim_R y$, by definition of $\sim_R$ we know that $xRy$ and $yRx$. Consequently, it's also true that $yRx$ and $xRy$. Thus $y \sim_R x$.
>
> To see that $R$ is transitive, consider any $x$, $y$, $z \in A$ such that $x \sim_R y$ and $y \sim_R z$. We will show that $x \sim_R z$. Since $x \sim_R y$, we have $xRy$ and $yRx$. Since $y \sim_R z$, we have $yRz$ and $zRy$. Since $xRy$ and $yRz$, we have $xRz$ because $R$ is a preorder and all preorders are transitive. Similarly, since $zRy$ and $yRx$, we have $zRx$. Thus $xRz$ and $zRx$, so $x \sim_R z$, as required.
>
> Since $\sim_R$ is reflexive, symmetric, and transitive, it is an equivalence relation. ■

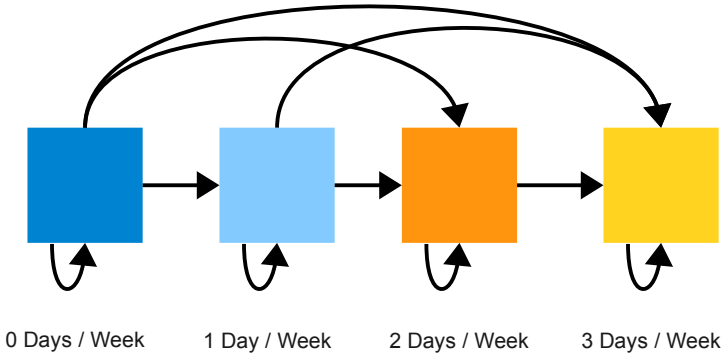
Great! We've shown that starting with a preorder, we can derive an equivalence relation. This is fairly interesting, since we first arrived at preorders by investigating relations that looked more like partial orders ("$x$ exercises no more than $y$") than equivalence relations ("$x$ and $y$ exercise the same amount"). Can we somehow transform our preorder into a partial order?

The answer is yes. To do so, let's take one more look at the graph of the relation "$x$ exercises no more than $y$." Below is the graph, with the equivalence classes highlighted:




Let's investigate how people in different equivalence classes are related to one another. Specifically, look at how the relation relates people across equivalence classes. To do this, let's redraw the graph. Specifically, let's think about the graph of the equivalence classes. We'll draw an arrow from one equivalence class to another if there's a node in one of the equivalence classes that has an edge to a node in the second equivalence class. For completeness, we'll include edges between nodes in an equivalence class and nodes in the same equivalence class.

This process, and the result, are shown below:




Now, take a look at this resulting graph. We can see that it's reflexive, since all the nodes have edges leading into themselves. It's also transitive, though it might take a minute for you to check that. This isn't anything special – the original preorder had these properties. However, we can also see that the graph is antisymmetric, since between any pair of nodes there's at most one edge. This means that the above graph represents a relation that is reflexive, antisymmetric, and transitive – a partial order!

This is not trivial! We began our investigation of "$x$ exercises no more than $y$" trying to see if we could come up with a partial order, but we only got a preorder. Now, we've successfully turned that preorder into a partial order, but at a price. While the original relation is a preorder over *people*, this partial order is a partial order over *equivalence classes* of people.

Again we have to ask ourselves whether or not this is a coincidence. Did we just get lucky by picking the relation "$x$ exercises no more than $y$" and doing this construction? In this case, no, it's not a coincidence. Think about what's going on at a high level. Starting with the preorder "$x$ exercises no more than $y$," the only factor preventing us from getting a partial order was the fact that many different people could all exercise the same amount without being the same person. By changing our preorder to work on equivalence

classes of people rather than people, then we actually can rank everything, since we've condensed all people who work out the same amount down into a single entity (the equivalence class). More generally, starting with a preorder, if we condense equal values into equivalence classes and define a new ordering relation over the equivalence classes, we will find that we have a partial order over those equivalence classes.

To formally prove this, we're going to need to introduce some new terms and definitions. Let's suppose that we start with a preorder $R$ over the set $A$. Our construction was as follows. First, we build the equivalence relation $\sim_R$ from $R$ that equated elements of $A$ that were mutually comparable by $R$. Next, we constructed the equivalence classes of $A$ under $\sim_R$, which gave us the quotient set $A / \sim_R$. (In case you need a refresher, the set $A / \sim_R$ is just the set of all equivalence classes of $A$ under the partial order $\sim_R$).

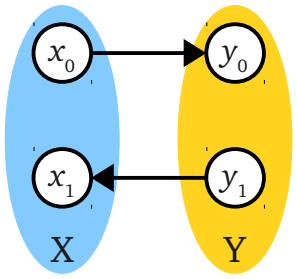
Once we've constructed $A / \sim_R$, we then defined a new relation that worked over those equivalence classes. In particular, we said that this new relation (we'll give it a name and a symbol in a second) related equivalence classes as follows: if $X$ and $Y$ are equivalence classes where some element of $X$ is related by $R$ to some element of $Y$, then $X$ and $Y$ themselves are related. More formally, if there is some $x \in X$ and some $y \in Y$ where $xRy$, then $X$ and $Y$ are related by this new relation. This relation is formed by "lifting" $R$ to work on equivalence classes rather than elements, and we'll denote it $R^+$. Formally, $R^+$ is defined as follows:

$$XR^+Y \quad \text{iff} \quad \text{there exists } x \in X \text{ and } y \in Y \text{ where } xRy.$$

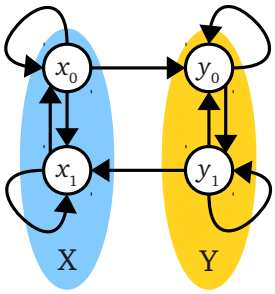
Our goal will be to prove that $R^+$ is a partial order over $A / \sim_R$, the set of equivalence classes of $A$ partitioned by the equivalence relation $R$. To do this, we'll prove that $R^+$ is reflexive, antisymmetric, and transitive.

The proofs that $R^+$ is reflexive and transitive are not particularly tricky. Every equivalence class is related to itself, since every element of each equivalence class relates back to itself. Consequently, each equivalence class is related by $R^+$ to itself, and so $R^+$ is reflexive. The argument for transitivity is similar, though it has a few tricky details.

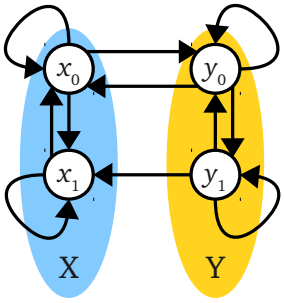
The hardest argument to make is antisymmetry, which isn't too surprising when you consider that this was the property we were lacking in the first place. To prove that $R^+$ is antisymmetric, we need to show that if $XR^+Y$ and $YR^+X$ (where $X$ and $Y$ are equivalence classes) that $X = Y$. Let's see exactly what this means. If $XR^+Y$, then there must be some $x \in X$ and $y \in Y$ such that $xRy$, and if $YR^+X$, then there must be some $x \in X$ and some $y \in Y$ such that $yRx$. However, these choices of $x$ and $y$ might not be the same in each case; after all, it could be possible that we have a setup like this one below:




Here, we can see that $x_0$ relates to $y_0$ and $y_1$ relates to $x_1$. From this, we somehow have to be able to conclude that $X = Y$. To do this, we can use the fact that $x_0$ and $x_1$ are in the same equivalence class, as are $y_0$ and $y_1$. Since the equivalence classes here are equivalence classes for the $\sim_R$ relation, this means that a more precise version of the above picture would be something like this:




Notice that this graph has a cycle that links together all of these elements. In particular, there is a path from $x_0$ to $y_0$ and from $y_0$ back to $x_0$. Because the relation $R$ is transitive, this means that there has to be an edge back from $y_0$ to $x_0$ as well, giving this picture:




And now the kicker. Notice from the above picture that $x_0 R y_0$ and $y_0 R x_0$. This means that $x_0 \sim_R y_0$. But remember: $X$ and $Y$ are equivalence classes under the $\sim_R$ relation. Since $x_0 \in X$ and $x_0 \sim_R y_0$, we have to have that $y_0 \in X$ as well. Consequently, $X$ and $Y$ have an element in common, namely $y_0$. As we proved in an earlier lemma in this chapter, since $X$ and $Y$ are equivalence classes with an element in common, we are guaranteed that $X = Y$.

The following proof formalizes this logic, along with the logic for reflexivity and transitivity:

***Theorem:*** Let $R$ be a preorder over $A$ and $R^+$ the associated relation over $A / \sim_R$. Then $R^+$ is a partial order over $A / \sim_R$.

***Proof:*** Let $R$ be any preorder over a set $A$ and let $R^+$ be the associated relation over $A / \sim_R$ defined as follows: $XRY$ iff there is some $x \in X$ and $y \in Y$ such that $xRy$. We will prove that $R^+$ is a partial order over $A / \sim_R$ by showing that it is reflexive, antisymmetric, and transitive.

To show that $R^+$ is reflexive, we need to show that for any equivalence class $X \in A / \sim_R$, that $XR^+X$. This means that we must show that there is some $x \in X$ and some $y \in X$ such that $xRy$. To see this, note that since $X$ is an equivalence class, that $X = [z]$ for some $z \in A$. Consequently, $z \in X$. Since $R$ is a preorder, it is reflexive, so $zRz$. Thus there exists a choice of $x \in X$ and $y \in X$ such that $xRy$ – namely, $x = y = z$. Thus $R^+$ is reflexive.

To show that $R^+$ is antisymmetric, consider any equivalence classes $X$, $Y \in A / \sim_R$ such that $XR^+Y$ and $YR^+X$. We need to show that $X = Y$. Since $XR^+Y$, there exists some $x_0 \in X$ and $y_0 \in Y$ such that $x_0Ry_0$. Since $YR^+X$, there exists some $y_1 \in Y$ and $x_1 \in X$ such that $y_1Rx_1$. Now, since $x_0 \in X$ and $x_1 \in X$ and $X$ is an equivalence class for $\sim_R$, we know that $x_0 \sim_R x_1$. Similarly, since $y_0 \in Y$ and $y_1 \in Y$, we know that $y_0 \sim_R y_1$. By our definition of $\sim_R$, this means that $x_0Rx_1$, $x_1Rx_0$, $y_0Ry_1$, and $y_1Ry_0$. Since we know that $y_0Ry_1$, $y_1Rx_1$, and $x_1Rx_0$, by transitivity of $R$ we know that $y_0Rx_0$. Consequently, $y_0Rx_0$ and $x_0Ry_0$. By definition of $\sim_R$, this means that $x_0 \sim_R y_0$. Now, since $X$ is an equivalence class under $\sim_R$, and since $x_0 \in X$, the fact that $x_0 \sim_R y_0$ means that $y_0 \in X$ as well. Since $y_0 \in X$ and $y_0 \in Y$ and $X$ and $Y$ are equivalence classes, we thus have that $X = Y$, as required.

To show that $R^+$ is transitive, consider any equivalence classes $X$, $Y$, $Z \in A / \sim_R$ such that $XR^+Y$ and $YR^+Z$. We will prove that $XR^+Z$, meaning that there exists some $x \in X$ and $z \in Z$ such that $xRz$. Since $XR^+Y$, there exists some $x_0 \in X$ and $y_0 \in Y$ such that $x_0Ry_0$, and since $YR^+Z$ there exists some $y_1 \in Y$ and $z_1 \in Z$ such that $y_1Rz_1$. Since $y_0 \in Y$ and $y_1 \in Y$, and since $Y$ is an equivalence class for $\sim_R$, we know that $y_0 \sim_R y_1$. This in turn means that $y_0Ry_1$. Consequently, we have that $x_0Ry_0$, $y_0Ry_1$, and $y_1Rz_1$. By the transitivity of $R$, this means that $x_0Rz_1$. Thus there exists an $x \in X$ and $z \in Z$ (namely, $x_0$ and $z_1$) such that $xRz$. Therefore, we have $XR^+Z$ as required.

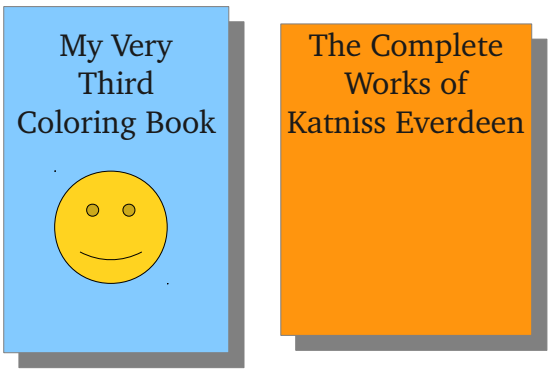
Since $R^+$ is reflexive, antisymmetric, and transitive, it is an equivalence relation. ■

Phew! That was a tricky proof. I personally really like it, since it pulls together many different types of relations and shows how their properties can combine together to build larger structures.

### 5.3.5 Combining Orderings ★

Suppose that you have a stack of books. Each book has a width and a height, both of which are real numbers. This means that we could consider relating books based on their widths or heights. Specifically, we could think about the relations "Book $A$ has a narrower width than book $B$" or "Book $A$ has a smaller height than book $B$." These relations end up being strict orders.

However, what happens if we try to rank two books based on both their width *and* their height? Now, the picture is a bit less clear. For example, consider the following two books:




Here, “My Very Third Coloring Book” is taller than “The Complete Works of Katniss Everdeen,”, but is less wide. Similarly, “The Complete Works of Katniss Everdeen” is wide than “My Very Third Coloring Book,” but less tall. How might we try to compare these books to one another?

We can actually ask this question more generally. Suppose that we have two ordered sets, and we consider a set of objects with two different traits, one drawn from the first ordered set and one drawn from the second ordered set. How might we use the existing ordering relations to construct a new ordering relation over these objects?

There is no one “right way” to do this. In fact, there are many ways that we could combine together two different ordering relations. In this section, we'll explore two.

### 5.3.5.1 The Product Ordering

Let's play around with how we might rank these books and see if we come up with anything. One way that we might try to rank the books is using the following idea: we'll say that one book is “bigger” than another iff it has either a greater width or a greater height (or both; this will be an inclusive or). More formally, let's suppose that we have pairs of values $(w, h)$ drawn from the set $\mathbb{R}^2$ of pairs of real numbers.* We can then define $<_{OR}$ as follows:

$$(w_1, h_1) <_{OR} (w_2, h_2) \quad \text{iff} \quad w_1 < w_2 \text{ or } h_1 < h_2$$

What sort of relation is this? Is it a strict order? Well, let's play around with it and see what properties it has. First, is the relation irreflexive? If we take any pair $(w, h)$, we'll find that it's not related to itself, since no book is wider or taller than itself. Next, is the relation asymmetric? Unfortunately, the answer is no. Consider one book with dimensions 8” × 5” and another book with dimensions 7” × 6”. Using our relation $<_{OR}$, we would have that $(8, 5) <_{OR} (7, 6)$ because $5 < 6$, but would also have that $(7, 6) <_{OR} (8, 5)$, since $7 < 8$. Consequently, this relation is not asymmetric. If we were to change the use of $<$ to $\leq$ above, then it wouldn't be antisymmetric either, since $(7, 6) \neq (8, 5)$. In other words, even though $<$ is a strict order over $\mathbb{R}$, the relation $<_{OR}$ we've just constructed isn't a strict order over $\mathbb{R}^2$.

Intuitively, this somewhat makes sense. An order relation is supposed to rank objects against one another consistently. Our relation $<_{OR}$ doesn't use a consistent ranking between objects, and might end up comparing two objects to one another using different traits (maybe by height the first time and by width the second).

To enforce that we consistently use the same ranking each time, let's try making a slight change to our $<_{OR}$ relation. Instead of ranking one book as bigger than another if *either* its height or its width is bigger, let's rank one book as bigger than another if *both* its height and its width are bigger. To do this, we'll define a new relation $<_{AND}$ as follows:

---

* This allows us to have negative-width or negative-height books. For simplicity, let's ignore that detail for now.

$$(w_1, h_1) <_{\text{AND}} (w_2, h_2) \quad \text{iff} \quad w_1 < w_2 \text{ and } h_1 < h_2$$

Now, we have a consistent way to rank objects against one another. For example, a book of dimensions 3" × 5" is definitely smaller than a book of dimensions 4" × 6", which is in turn definitely smaller than a book of dimensions 4.5" × 6.5". (Both of these are smaller than this set of course notes, which is on 8.5" × 11" paper. So there.)

Note, however, that this ordering is not necessarily a total ordering. Specifically, consider two books, one of whose dimensions are 4" × 8" and one whose dimensions are 5" × 7". In this case, neither book compares bigger than the other according to the $<_{\text{AND}}$ relation, since the 4" × 8" book is taller but narrower. Even though $<$ over $\mathbb{R}$ is a total order, the resulting order $<_{\text{AND}}$ is no longer total. This isn't necessarily a bad thing, though. After all, it's perfectly reasonable to say that neither book is bigger than the other, since neither book is strictly larger.

Let's formalize and generalize the construction we've just come up with. In our book example, we started with the strict order $<$ over the set $\mathbb{R}$, and constructed from it a new relation $<_{\text{AND}}$ over the set $\mathbb{R}^2$. More generally, we could think about starting with any arbitrary strict order $<_A$ over some set $A$ and constructing a relation $<_{\text{AND}}$ over the set $A^2$ as follows:

$$(a_{11}, a_{12}) <_{\text{AND}} (a_{21}, a_{22}) \quad \text{iff} \quad a_{11} <_A a_{22} \text{ and } a_{12} <_A a_{22}$$

But we can carry this even further. Rather than taking a single order relation and generalizing it to work over the Cartesian square of its underlying set, we could start off with two arbitrary order relations and from them construct an ordering relation over the Cartesian product of the two underlying sets. For example, we could try ranking books by two traits – their number of pages (which is a natural number), and the weight (which is a real number). In that case, a book would be described as an element of $\mathbb{N} \times \mathbb{R}$, a natural number paired with an integer. But we could easily modify the above construction so that we can define $<_{\text{AND}}$ over $\mathbb{N} \times \mathbb{R}$ as follows:

$$(p_1, w_1) <_{\text{AND}} (p_2, w_2) \quad \text{iff} \quad p_1 < p_2 \text{ and } w_1 < w_2$$

Consequently, we'll consider the following very general construction, which is called the *product order*:

> Let $(A, <_A)$ and $(B, <_B)$ be strictly-ordered sets (an analogous construction works for posets). Then the relation $<_{\text{AND}}$ over $A \times B$ defined as follows is called the ***product order*** of $A \times B$:
>
> $$(a_1, b_1) <_{\text{AND}} (a_2, b_2) \quad \text{iff} \quad a_1 <_A a_2 \text{ and } b_1 <_B b_2$$

In order to justify the use of this definition, we should definitely be sure to prove that this actually gives back an order relation. Otherwise, this entire exercise has been entirely meaningless! Let's do that right here.

***Theorem:*** Let $(A, <_A)$ and $(B, <_B)$ be strictly-ordered sets and let $<_{AND}$ be the product order over $A \times B$. Then $<_{AND}$ is a strict order over $A \times B$.

***Proof:*** We will show that $<_{AND}$ is irreflexive, asymmetric, and transitive.

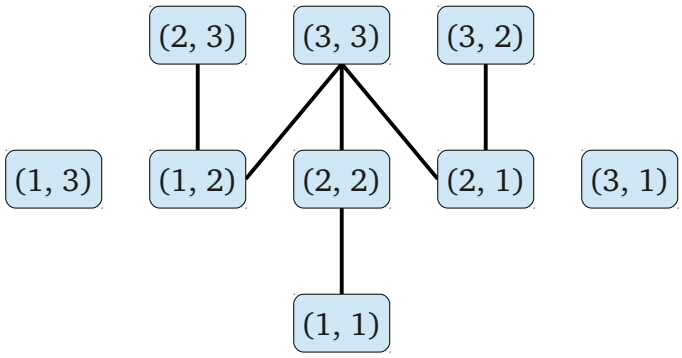
To see that $<_{AND}$ is irreflexive, we will show for any $(a, b) \in A \times B$ that $(a, b) \not<_{AND} (a, b)$. To see this, note that since $<_A$ is a strict order, that $a \not<_A a$. Consequently, it is not true that $a <_A a$ and $b <_B b$. Therefore $(a, b) \not<_{AND} (a, b)$, as required.

To see that $<_{AND}$ is asymmetric, we will show that if $(a_1, b_1) <_{AND} (a_2, b_2)$, then $(a_2, b_2) \not<_{AND} (a_1, b_1)$. To see this, note that if $(a_1, b_1) <_{AND} (a_2, b_2)$, then $a_1 <_A a_2$. Since $<_A$ is a strict order, it is asymmetric, so $a_2 \not<_A a_1$. Therefore, it is not true that $a_2 <_A a_1$ and $b_2 <_B b_1$. Thus $(a_2, b_2) \not<_{AND} (a_1, b_1)$, as required.

To see that $<_{AND}$ is transitive, we will show that if $(a_1, b_1) <_{AND} (a_2, b_2)$ and $(a_2, b_2) <_{AND} (a_3, b_3)$, then $(a_1, b_1) <_{AND} (a_3, b_3)$. To see this, note that if $(a_1, b_1) <_{AND} (a_2, b_2)$, then $a_1 <_A a_2$ and $b_1 <_B b_2$. Similarly, if $(a_2, b_2) <_{AND} (a_3, b_3)$, then $a_2 <_A a_3$ and $b_2 <_B b_3$. Since $<_A$ and $<_B$ are strict orders, they are transitive, so $a_1 <_A a_3$ and $b_1 <_B b_3$. Consequently, we have $(a_1, b_1) <_{AND} (a_3, b_3)$, as required.

Since $<_{AND}$ is irreflexive, asymmetric, and transitive, it is a strict order. ■

To get a better feel for what these relations look like, let's check out the Hasse diagrams for a few small relations formed this way. For example, suppose that we take the set $S = \{1, 2, 3\}$ ordered using the normal $<$ operator. If we then think about $<_{AND}$ over the set $S^2$, then we get the relation given by this Hasse diagram:

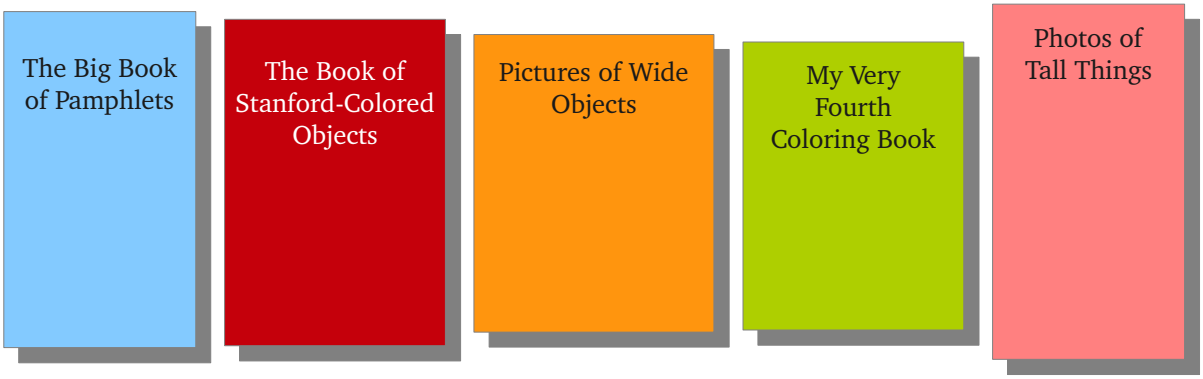



Notice that although $S$ is strictly totally ordered by $<$, the $<_{AND}$ relation is not a strict total ordering over the set $S^2$. The elements (1, 3) and (3, 1) aren't comparable to any other elements of the set, for example.

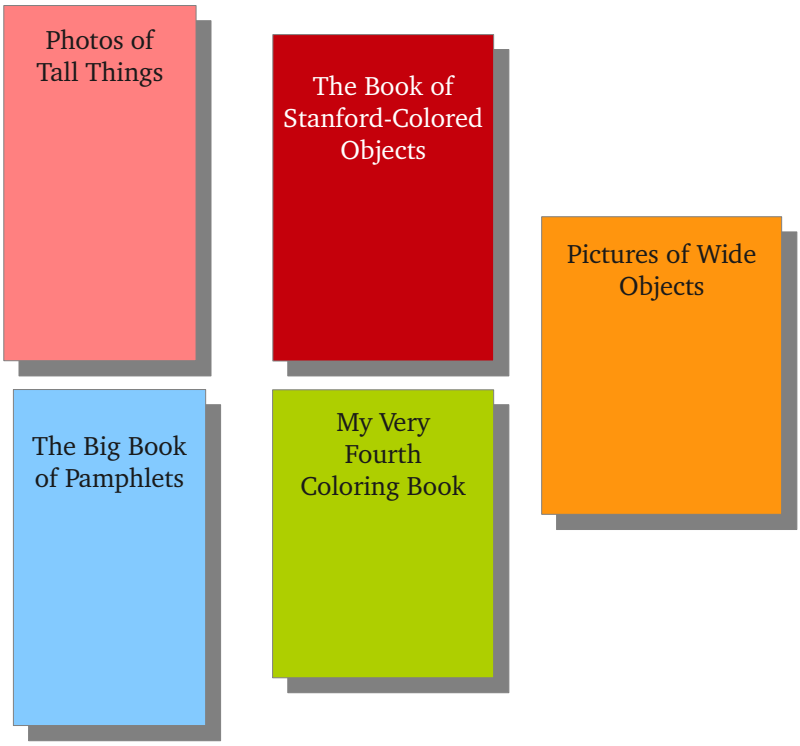
### 5.3.5.2 The Lexicographical Ordering

The product ordering does give a way to take two orderings and combine them together into a single order, but in doing so it loses some properties of the original order. For example, in the case of comparing books, we started with two total orders (namely, two copies of the $<$ order over $\mathbb{R}$) and ended up with a non-total order. Could we somehow combine two orders in a way that preserves total orderings?

To answer this question, let's consider a completely different way of ranking the sizes of books. Consider the following books:

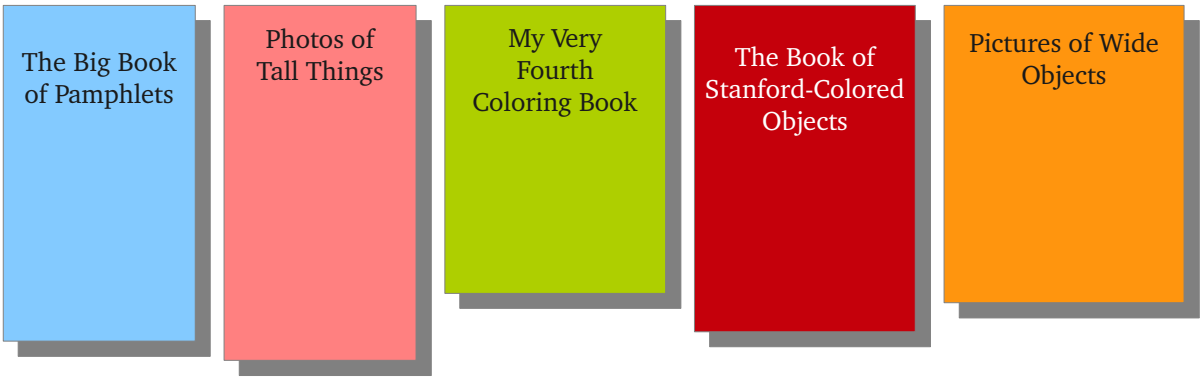



Let's begin by sorting all of these books from left to right by their width – wider books go on the right, and narrower books go on the left:




Notice that there are several different books that have the same width, which I have represented by putting them vertically atop one another. By doing this, we can see that the books now fall into different clusters grouped by their width, where the clusters are then sorted by the height of the books they contain.

Now, let's suppose that we sort each cluster of books by ordering those books by their height. In doing so, we won't change the relative ordering of the clusters. Instead, we're just reordering the books within the clusters. If we do this, we end up getting this ordering:




Notice that we now have completely ordered the books from left to right as follows – first, we order the books by their width, and when two books are tied for the same width we rank them by their height.

The way that we have ordered these books is called the *lexicographical ordering*, which we'll formally define in a short while. Intuitively, we started with two orderings (one on width, one on height) and combined them together as follows. Given any two books, we first look at their width. If one book is wider than the other, we immediately say that that book comes after the other book. Otherwise, if they have the same width, we then compare them by their height. In other words, the height of the book only comes in as a "tiebreaker." The main determinant of the ordering is width.

Abstracting away a bit from books and just looking at pairs drawn from $\mathbb{R}^2$, our ordering (which we'll denote $<_{lex}$) is defined as follows:

$$(w_1, h_1) <_{lex} (w_2, h_2) \quad \text{iff} \quad w_1 < w_2, \text{ or } w_1 = w_2 \text{ and } h_1 < h_2$$

Notice here that the height isn't even considered unless the widths are the same.

In the preceding section on the product order, we abstracted away from working with pairs of real numbers to working with arbitrary strictly ordered sets. We will do so here when formally defining the lexicographical ordering.

Let $(A, <_A)$ and $(B, <_B)$ be strictly ordered sets. The ***lexicographical ordering*** of $A \times B$ is the ordering $<_{lex}$ defined as follows:

$$(a_1, b_1) <_{lex} (a_2, b_2) \quad \text{iff} \quad a_1 <_A a_2, \text{ or } a_1 = a_2 \text{ and } b_1 <_B b_2$$

As before, we should probably stop to prove that this relation actually is a strict order before we start to reason about its other properties. After all, from just what's listed above it's hard to see exactly why it is that this would give us a nice ordering at all! With that in mind, let's prove the following result:

***Theorem:*** Let $(A, <_A)$ and $(B, <_B)$ be strictly ordered sets. Then $<_{lex}$ defined over $A \times B$ is a strict order.

***Proof:*** We will prove that $<_{lex}$ is irreflexive, asymmetric, and transitive.

To see that $<_{lex}$ is irreflexive, consider any $(a, b) \in A \times B$. We will prove that $(a, b) \not<_{lex} (a, b)$. To see this, note that we have that $a \not<_A a$, since $<_A$ is a strict order. We also have that $b \not<_B b$, since $<_B$ is a strict order. Thus it is not true that $a <_A a$, nor is it true that $a = a$ and $b <_B b$. Consequently, by definition, $(a, b) \not<_{lex} (a, b)$.

To see that $<_{lex}$ is asymmetric, consider any $(a_1, b_1)$ and $(a_2, b_2)$ such that $(a_1, b_1) <_{lex} (a_2, b_2)$. We will prove that $(a_2, b_2) \not<_{lex} (a_1, b_1)$. To do this, we consider two cases.

*Case 1:* $a_1 <_A a_2$. Then since $<_A$ is a strict order, we know that $a_2 \not< a_1$ (by asymmetry) and that $a_1 \neq a_2$ (by irreflexivity). Consequently, it is not true that $a_2 < a_1$, nor is it true that $a_2 = a_1$ and $b_2 <_B b_1$. Thus $(a_2, b_2) \not<_{lex} (a_1, b_1)$.

*Case 2:* $a_1 = a_2$. Since $(a_1, b_1) <_{lex} (a_2, b_2)$, this means that $b_1 <_B b_2$. Since $<_B$ is a strict order, we have that $b_2 \not<_B b_1$ (by asymmetry). Since $<_A$ is a strict order, we also know that $a_2 \not<_A a_1$ (by irreflexivity). Thus it is not true that $a_2 < a_1$, nor is it true that $a_2 = a_1$ and $b_2 <_B b_1$. Thus $(a_2, b_2) \not<_{lex} (a_1, b_1)$.

In either case, we have $(a_2, b_2) \not<_{lex} (a_1, b_1)$, so $<_{lex}$ is asymmetric.

Finally, to see that $<_{lex}$ is transitive, consider any $(a_1, b_1)$, $(a_2, b_2)$, and $(a_3, b_3)$ such that $(a_1, b_1) <_{lex} (a_2, b_2)$ and $(a_2, b_2) <_{lex} (a_3, b_3)$. We will prove that $(a_1, b_1) <_{lex} (a_3, b_3)$. To do so, we consider four cases:

*Case 1:* $a_1 <_A a_2$ and $a_2 <_A a_3$. Since $<_A$ is a strict order, it is transitive, so $a_1 <_A a_3$. Thus by definition, $(a_1, b_1) <_{lex} (a_3, b_3)$.

*Case 2:* $a_1 <_A a_2$ and $a_2 = a_3$. Therefore, $a_1 <_A a_3$. Thus by definition, $(a_1, b_1) <_{lex} (a_3, b_3)$.

*Case 3:* $a_1 = a_2$ and $a_2 <_A a_3$. Therefore, $a_1 <_A a_3$. Thus by definition, $(a_1, b_1) <_{lex} (a_3, b_3)$.

*Case 4:* $a_1 = a_2$ and $a_2 = a_3$. Therefore, $a_1 = a_3$. Moreover, since $<_A$ is a strict order, we know that $a_1 \not<_A a_2$ and $a_2 \not<_A a_3$, since $<_A$ is irreflexive. Consequently, since we know $(a_1, b_1) <_{lex} (a_2, b_2)$ and $(a_2, b_2) <_{lex} (a_3, b_3)$, we have $b_1 <_B b_2$ and $b_2 <_B b_3$. Since $<_B$ is a strict order, it is transitive, so $b_1 <_B b_3$. Thus $a_1 = a_3$ and $b_1 <_B b_3$, so by definition $(a_1, b_1) <_{lex} (a_3, b_3)$.

In all four cases, we see $(a_1, b_1) <_{lex} (a_3, b_3)$. Thus $<_{lex}$ is transitive.

Since $<_{lex}$ is irreflexive, asymmetric, and transitive, it is a strict order. ■

This proof is long simply due to the number of cases we have to check. However, at each stage, we can use the relevant properties of strict orders in order to justify our result.

The real beauty of the lexicographical ordering is that if the two relations from which we construct the lexicographical ordering are strict total orders, then the lexicographical ordering is also a strict total ordering.* This has important applications, as you'll see later in the chapter when we talk about well-ordered and well-founded sets. Before concluding this section, we'll prove one last theorem.

> ***Theorem:*** Let $(A, <_A)$ and $(B, <_B)$ be strictly, totally ordered sets. Then the lexicographical ordering $<_{lex}$ over $A \times B$ is a strict total order.

Why does the lexicographical ordering have this property? To understand why, let's suppose that we have two strict total orders and combine them together into the lexicographical ordering. Now consider any two distinct pairs of values. If their first elements aren't the same, then the lexicographical order will make one pair compare larger than the other. Otherwise, if their first elements are the same, then their second values must be different. Thus the lexicographical ordering will rank whichever pair has the larger second value as larger.

Using this intuition, the proof is actually not very difficult:

> ***Theorem:*** Let $(A, <_A)$ and $(B, <_B)$ be strictly, totally ordered sets. Then the lexicographical ordering $<_{lex}$ over $A \times B$ is a strict total order.

> ***Proof:*** Let $(A, <_A)$ and $(B, <_B)$ be arbitrary strictly, totally-ordered sets and consider their lexicographical ordering $<_{lex}$ over $A \times B$. We will show that $<_{lex}$ is a strict total order. Since by our previous theorem we know that $<_{lex}$ is a total order, we only need to show that it is trichotomous.
>
> Consider any $(a_1, b_1), (a_2, b_2) \in A \times B$. We will show that exactly one of the following is true: $(a_1, b_1) <_{lex} (a_2, b_2)$, $(a_1, b_1) = (a_2, b_2)$, or $(a_2, b_2) <_{lex} (a_1, b_1)$. Note that since $<_{lex}$ is a strict order, it is irreflexive and asymmetric. As a result, it is not possible for any two of these three relations between the two pairs to hold simultaneously. Consequently, we just need to show that at least one of the following relations holds between the two pairs.
>
> First, note that if $(a_1, b_1) = (a_2, b_2)$, then we are done. So suppose that $(a_1, b_1) \neq (a_2, b_2)$. This means that $a_1 \neq a_2$ or $b_1 \neq b_2$ (or both). If $a_1 \neq a_2$, then since $<_A$ is a strict total order, it is trichotomous, so either $a_1 <_A a_2$ or $a_2 <_A a_1$. In the first case, $(a_1, b_1) <_{lex} (a_2, b_2)$, and in the second case, $(a_2, b_2) <_{lex} (a_1, b_1)$; either way we are done. Otherwise, $a_1 = a_2$, so we know that $b_1 \neq b_2$. Since $<_B$ is a strict total order, it is trichotomous, so either $b_1 <_B b_2$ or $b_2 <_B b_1$. Since $a_1 = a_2$, in the first case $(a_1, b_1) <_{lex} (a_2, b_2)$, and in the second case, $(a_2, b_2) <_{lex} (a_1, b_1)$. In both cases we are done.
>
> Thus in all possible cases, one of the three aforementioned relations holds between the pairs. Thus $<_{lex}$ is trichotomous, so $<_{lex}$ is a strict total order. ■

* We haven't seen this term in a while, so as a refresher: a strict total order is a strict order $<$ that is *trichotomous*, meaning that for any $x$ and $y$, exactly one of the following is true: $x < y$, or $x = y$, or $y < x$.

# 5.4 Well-Ordered and Well-Founded Sets ★

Our discussion of greatest, least, maximal, and minimal elements of sets indicates that not all ordered sets are alike. Some will have greatest and least values, while others do not. Some orders have maximal and minimal elements, while others do not.

For example, consider the sets $\mathbb{N}$ and $\mathbb{Z}$ of natural numbers and integers, respectively, ordered by the $\leq$ relation. In many ways, these sets are similar. For example, neither $\mathbb{N}$ nor $\mathbb{Z}$ has a greatest element; you can always add one to any natural number or integer to get a greater natural or integer. However, they differ in several key ways. For instance, the set $\mathbb{N}$ has a least element with respect to $\leq$ (namely, 0), while $\mathbb{Z}$ does not. $\mathbb{N}$ also has the property that any nonempty set of natural numbers has a least element (this is the well-ordering principle), while not all sets of integers necessarily have a least element – for example, the set $\mathbb{Z}$ itself has no least element.

Similarly, consider the relation $\subseteq$ over $\wp(\mathbb{N})$. This set is partially-ordered, and it has a least element (namely, $\varnothing$). Unlike the set $\mathbb{N}$ itself, $\subseteq$ over $\wp(\mathbb{N})$ has a greatest element (namely, $\mathbb{N}$). Also unlike $\mathbb{N}$, not all subsets of $\wp(\mathbb{N})$ necessarily have a least element. For example, the set $\{ \{1\}, \{2\} \}$ has no least element according to $\subseteq$, though in this case both elements of the set are minimal.

Now take a totally different set: $\mathbb{Q}$, the set of all rational numbers. This set doesn't have a least or greatest element with respect to $\leq$. Moreover, not all sets of rational numbers necessarily have a least element. Take the set $\{ x \in \mathbb{Q} \mid x > 0 \}$. This set has no least element – if some rational number $q$ were the least element of the set, then, since $q > 0$, we also have that $q / 2 > 0$. Since $0 < q / 2 < q$, this means that $q / 2$ is also in this set, and is a smaller value. Not only does this set not have a least element, it has no minimal elements either, using exactly the same line of reasoning.

The fact that many seemingly similar order relations have completely different behavior with respect to greatest, least, maximal, and minimal elements suggests that there are many different flavors of order relationships beyond just partial orders, strict orders, total orders, and strict total orders. This section explores two more types of orders – well-orders and well-founded orders – that have numerous applications in discrete mathematics and computer science. In doing so, we will once again return to mathematical induction, generalizing the principle of induction to sets other than just the natural numbers.

## 5.4.1 Well-Orders

In Chapter Three, we discussed the well-ordering principle, which states that any nonempty set of natural numbers has a least element. As we saw, this property is somewhat particular to the natural numbers. The set $\mathbb{Z}$ doesn't have this property, nor does $\mathbb{Q}$, nor $\mathbb{R}$, etc.

However, $\mathbb{N}$ is not the only set that happens to have this property. For example, consider a tournament of the game Quidditch from the *Harry Potter* universe. In each game, two teams compete and earn points. Multiple matches are played across the teams. At the end, the team with the highest total score (*not* the greatest number of games won) ends up winning the tournament. From what I've read, I don't think there was ever a Quidditch tournament in which two teams, at the end of the tournament, had exactly the same total score. However, let's suppose that we want to add in a tiebreaking rule just in case this occurs. Specifically, we'll say that if two teams each have the same total number of points, the team with more total games won ends up being the tournament winner.

To represent teams' progresses in this tournament, we can assign each team an ordered pair of two values – first, the total number of points they've earned, and second, the total number of games they've won. For example, a team whose score was (320, 2) would have earned 320 total points and won two games. A team

whose score was (1370, 0) would have earned 1370 total points, but won no games. Here, the team whose score was (1370, 0) is doing better than the team whose score was (320, 2).

Given this setup, we can define a relation $Q$ (for Quidditch) over ordered pairs of natural numbers as follows:

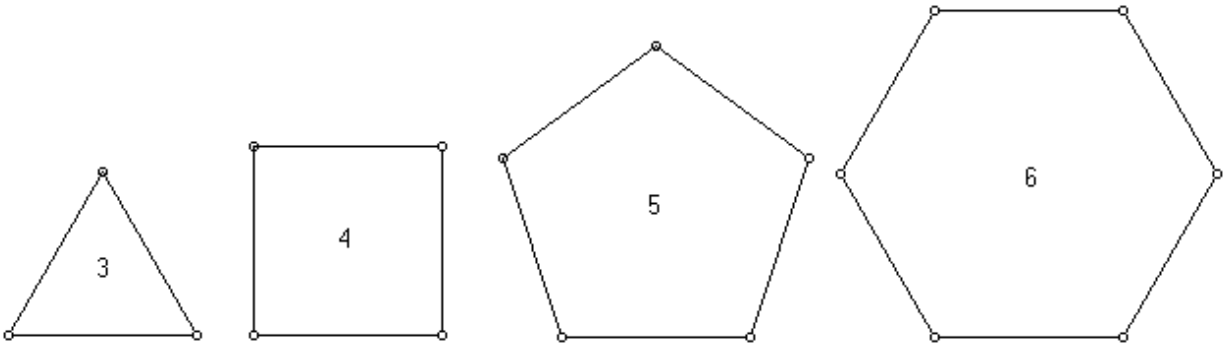
$$(P_1, W_1)\ Q\ (P_2, W_2) \text{ iff } P_1 < P_2, \text{ or } P_1 = P_2 \text{ and } W_1 < W_2.$$

For example, (1370, 0) $Q$ (1370, 1) and (600, 4) $Q$ (1370, 0). If you'll notice, this is just the normal lexicographical ordering over $\mathbb{N}^2$, which as we've seen before is a strict order over $\mathbb{N}$. More importantly for our discussion, this ordering has the property that in any nonempty set of Quidditch scores, there is always a score that is the lowest score out of the set. To see this, note that some collection of scores (maybe just one) will have the lowest total number of points out of all the scores. Of the scores with the least total number of points, one will have the least total number of games won. That score is the lowest score out of all of the scores.

This section explores the properties of relations like $<$ over $\mathbb{N}$ and $Q$ over Quidditch scores where any nonempty set of values drawn from the set has a least element. These orders are called *well-orders*, as defined below.

> An order relation $R$ over a set $A$ is called a ***well-order*** iff for any nonempty set $S \subseteq A$, $S$ has a least element.

The natural numbers $\mathbb{N}$ ordered by $<$ (or $\leq$) and Quidditch scores ordered by $Q$ are examples of well-orders. To give another example, let's consider a set of geometric objects. A *regular polygon* is a polygon where each side has the same length and all the angles between sides are the same. For example, here is a regular triangle, square, pentagon, and hexagon:*




We can think about the set of all regular polygons ignoring size (which we'll call $S$) ordered by the relation $R$ defined as follows: $xRy$ iff $x$ has fewer sides than $y$. This ordering is a strict total ordering, but we won't prove that here.

This ordering is also a well-ordering. Given any nonempty set of regular polygons, some polygon in that set must have the least number of sides. That polygon is then the least element of the set.

Let's now start considering regular polygons with progressively more and more sides. As the number of sides increases, the polygons start to look more and more like a circle. That said, a circle isn't a regular polygon; it's just something that regular polygons begin to approximate more and more. Given this, we can think about the set S' defined as follows: S' = S ∪ { ○ }, where ○ is a circle. If we now rethink our relation R,

* Image credit: http://www.georgehart.com/virtual-polyhedra/figs/polygons.gif

which relates shapes by the number of sides they have, then we can reasonably extend R by adding in the rule that a circle has more sides than any polygon. This gives us this new relation:

$x$R'$y$ iff $x$R$y$ and $x$ and $y$ are polygons and $x$ has fewer sides than $y$, or $y = ○$ and $x \neq ○$.

For example, △R'□, and □R'○. This relation is also a strict total order (again, we won't prove it here).

Interestingly, this new relation is still a well-ordering. We can sketch out of a proof of this here. If we take any nonempty set of elements of S', either that set is the set { ○ }, or that set contains at least one polygon. In the former case, ○ is the least element. In the latter case, whichever polygon has the least number of sides is still the least element of the new set.

#### 5.4.1.1 Properties of Well-Ordered Sets

Now that we have this definition, let's see if we can explore what properties well-ordered sets must have. For starters, let's play around with some of the order relations we've seen so far so that we can see if we can find any general properties that have to hold of well-orderings.

Let's start with the $\subseteq$ relation over a set like $\wp(\mathbb{N})$. Is this a well-ordering? If so, that would mean that if we took any nonempty collection of sets from $\wp(\mathbb{N})$, then we should find that one of those sets is the least, meaning that it's a subset of all of them. Unfortunately, this isn't the case. Note, for example, that { {1}, {2} } doesn't have a least element according to $\subseteq$, since neither of these sets is a subset of the other.

What about the divisibility relation | over $\mathbb{N}$? Is it a well-ordering? Let's try some examples. In the set {1, 2, 3, 4, 5}, there is a least element according to |, namely 1, since 1 divides all of the values in this set. Similarly, in the set { 8, 16, 32, 64, 128 }, the least element is 8, since it divides all the other numbers. However, the set { 2, 3, 4, 5 } does not have a least element, since none of these values divide all of the others.

Neither of these relations are well-orders. Interesting, neither of these relations are *total* orders either. Is this a coincidence? The answer is no. It turns out that it's impossible for any non-total order $R$ over any set $A$ to be a well-order. The reason is that if there are two elements $x, y \in A$ where $x \neq y$, $x \not R y$, and $y \not R x$, then the set $\{x, y\}$ can't have a least element. We can formalize this observation here. The following proof works with partial orders, but we could just have easily worked with strict orders instead:

***Theorem:*** Let $\leq$ be a well-ordered partial order over $A$. Then $R$ is total.

***Proof:*** By contrapositive; we will prove that if $\leq$ is not a total order, then $\leq$ is not a well-order either. Consider any partial order $\leq$ over a set $A$ that is not a total order. Then there must exist $x, y \in R$ where $x \not\leq y$ and $y \not\leq x$. Since $\leq$ is a total order, it is reflexive. Consequently, we know that $x \neq y$, since otherwise we'd have $x \leq y$.

Now, consider the set $\{x, y\}$. This set has no least element, since neither $x \leq y$ nor $y \leq x$. Since this set is nonempty, we thus have that $\leq$ is not a well-order. ■

We've just shown that if a relation is a well-order, then it must also be a (strict) total order. Is the opposite true? That is, if we have a (strict) total order, is it guaranteed to be a well-order? Unfortunately, the answer is no. As an example, take the relation $<$ over the set $\mathbb{Z}$. This relation is a strict order, but it is not a well-order, since $\mathbb{Z}$ itself has no least element. In other words, whether or not a relation is a well-order is separate

from whether or not it is a (strict) total order. All well-orders are (strict) total orders, but not necessarily the other way around.

If you'll recall from earlier in this chapter, we discussed different ways of combining together relations. In doing so, we saw two different ways that we could combine together two different order relations. First, given two ordered sets $(A_1, R_1)$ and $(A_2, R_2)$, we could combine them together by taking their product to get a new relation $R_1 \times R_2$ over the set $A_1 \times A_2$, which was defined as $(a_1, a_2)\ R_1 \times R_2\ (b_1, b_2)$ iff $a_1 R_1 b_1$ and $a_2 R_2 b_2$. We saw that even if $R_1$ and $R_2$ were (strict) total orders, the resulting relation we obtained over $A_1 \times A_2$ was not necessarily a (strict) total order. In particular, this means that if $R_1$ and $R_2$ are well-ordered sets, we are not necessarily guaranteed that the relation $R_1 \times R_2$ obtained this way is well-ordered.

However, we also saw another way of combining relations: the lexicographical ordering. Recall that given two strictly ordered sets $(A_1, R_1)$ and $(A_2, R_2)$, we could construct the lexicographical order $R_{lex}$ over $A_1 \times A_2$ as follows: $(a_1, a_2)\ R_{lex}\ (b_1, b_2)$ iff $a_1 R_1 b_1$ or $a_1 = b_1$ and $a_2 R_2 b_2$. This ordering, we saw, has the property that if $R_1$ and $R_2$ are strict total orders, then $R_{lex}$ is a strict total order as well.

Now, let's revisit this construction from the perspective of well-orders. Let's suppose we have two strictly, well-ordered sets $(A_1, R_1)$ and $(A_2, R_2)$. Is the lexicographical ordering $R_{lex}$ defined over $A_1 \times A_2$ this way necessarily well-ordered?

Let's take a minute to think about this. In order to prove this result, we would need to show that if we take any nonempty subset $S \subseteq A_1 \times A_2$, even infinite subsets, there must be some least element according to $R_{lex}$. What would this element have to look like? Well, we know that for any other element of $S$, this least element either must have a smaller first component or an equal first component and a smaller second component. Fortunately, we can constructively show exactly how we can go about finding a pair with this property.

Let's begin by taking our set $S$ and looking at just the first components of each of the elements of $S$. This gives us back a set of elements from $A_1$. Because we know that $R_1$ is a well-order over $A_1$, this means that there has to be a least element $a_0$ in this set. Now, some of the pairs from $S$ will have $a_0$ as their first component. If there's just one pair in $S$ with $a_0$ as its first component, then we're done – it has to be the lexicographically least element of the set $S$. However, there might end up being a bunch of pairs in $S$ that have $a_0$ as their first component. But not to worry! Let's look purely at those pairs. We found $a_0$ by looking at the first component of all of the pairs in $S$, but we haven't yet looked at the second components of any of those pairs. Starting with just the set of pairs whose first element is equal to $a_0$, let's look at all of their second components. This gives us a set of elements from $A_2$, and since $R_2$ is a well-order over those elements, that set must have a least element, which we'll call $b_0$.

Now, think about the pair $(a_0, b_0)$ and how it relates to all other elements of the set $S$. It's automatically smaller than all of the other pairs in $S$ that don't have $a_0$ as their first element, and of the remaining pairs, it has the smallest of all of the $b_0$ values. Consequently, the pair $(a_0, b_0)$ is the least element of $S$.

This construction only relied on the fact that $(A_1, R_1)$ and $(A_2, R_2)$ were well-ordered sets. As a result, the construction we've just done shows that if we start with two well-ordered sets and construct their lexicographical ordering, we are guaranteed that the resulting set is also well-ordered. This gives us a way to build well-ordered sets out of existing well-ordered sets: we can keep combining them together by constructing their lexicographical orderings.

The following proof formalizes the above intuition. I would suggest reading over it in detail to see how we use all of the definitions we've built up so far in a single proof.

***Theorem:*** Let $(A_1, <_1)$ and $(A_2, <_2)$ be strict, well-ordered sets. Then the lexicographical ordering $<_{lex}$ over $A_1 \times A_2$ is a strict well-ordering.

***Proof:*** Let $(A_1, <_1)$ and $(A_2, <_2)$ be strict, well-ordered sets and $<_{lex}$ the lexicographical ordering over $A_1 \times A_2$. We will prove that $<_{lex}$ is a well-ordering by showing that any nonempty set $S \subseteq A_1 \times A_2$ has a least element.

Consider any nonempty set $S \subseteq A_1 \times A_2$. Let $S_1 = \{a \in A_1 \mid \text{there exists a pair } (a_1, b_1) \in S\}$ be the set of all elements of $A_1$ that are the first component of some pair in $S$. Since $S$ is nonempty, the set $S_1$ is nonempty. Moreover, $S_1 \subseteq A_1$. Since $<_1$ is a well-ordering over $A_1$, there must be a least element of $S_1$; call it $a_0$. Accordingly, there is at least one pair in $S$ whose first component is $a_0$.

Now, let $S_2 = \{b \in A_2 \mid \text{there exists a pair } (a_0, b) \in S\}$ be the set of all elements of $A_2$ that are the second component of some pair in $S$ with $a_0$ as its first component. Since there is at least one pair in $S$ whose first component is $a_0$, the set $S_2$ is nonempty. Moreover, $S_2 \subseteq A_2$. Since $<_2$ is a well-ordering over $A_2$, there must be a least element of $S_2$; call it $b_0$. This means, in particular, that $(a_0, b_0) \in S$.

We claim that $(a_0, b_0)$ is the least element of $S$. To see this, consider any $(a, b) \in S$. Note that $a_0, a \in S_1$ by our construction of $S_1$. Since $a_0$ was the least element of that set, either $a_0 R_1 a$, or $a_0 = a$. In the first case, by the definition of $<_{lex}$, we know that $(a_0, b_0) <_{lex} (a, b)$. In the second case, since $a_0 = a$, we know that $(a, b) = (a_0, b)$. Thus by construction of $S_2$, we have that $b_0, b \in S_2$. Since $b_0$ is the least element of $S_2$, this means that either $b_0 <_2 b$ or $b_0 = b$. In the first case, we have that $(a_0, b_0) <_{lex} (a_0, b)$. In the second case, we have that $(a_0, b) = (a_0, b_0)$, meaning that $(a, b) = (a_0, b_0)$. Thus for any $(a, b) \in S$, either $(a_0, b_0) = (a, b)$, or $(a_0, b_0) <_{lex} (a, b)$. Thus $(a_0, b_0)$ is the least element of $S$.

Since our choice of $S$ was arbitrary, this shows that any nonempty subset of $A_1 \times B_1$ has a least element. Thus $<_{lex}$ is a well-ordering. ■

The structure of this proof is particularly interesting. The first half of the proof builds up two sets, $S_1$ and $S_2$, and uses their existence to find the element $(a_0, b_0)$. The second half then shows that this element must be the least element of the set. Many proofs involving well-orderings work this way. We use the definition of the relation in question to single out some element, then prove that the element we've found is the least element of some set.

## 5.4.2 Well-Founded Orders

As we saw in the last section, the relation $\mid$ over $\mathbb{N}$ is a not a well-ordering because it is not a total order. Although not all nonempty sets of natural numbers has a least element with respect to divisibility (that is, a number that divides all other numbers in the set), any nonempty set of natural numbers does have a *minimal* element with respect to divisibility (that is, a number that isn't divided by any other number in the set). For example, in $\{2, 4, 6, 8\}$, 2 is a minimal element with respect to $\mid$, since nothing else in the set divides it. In $\{ n^2 \mid n \in \mathbb{N} \}$, 1 is a minimal element with respect to $\mid$. In $\{ 4, 5, 6, 7 \}$, *all* elements of the set are minimal elements, since none of them are divided by any other elements of the set.

An order is called a well-order if every nonempty set of elements has a least element according to that order. A slightly weaker definition is that of a *well-founded order*, which is given here:

An order relation $R$ over a set $A$ is called ***well-founded*** iff every nonempty set $S \subseteq A$ has a minimal element with respect to $R$.

Since all least elements are also minimal elements, this means that all well-orders are also well-founded orders. However, some orders, like $|$, are well-founded but are not well-orders. Consequently, well-founded orders are a less precise concept than well-orders, though they are still quite useful.

I've alleged that $|$ is a well-founded order, but we haven't actually proven that yet. How exactly might we show this? Our goal will be to prove that for any nonempty set of natural numbers, there is a minimal element with respect to $|$. By definition, this means that we want to show that any nonempty set of natural numbers contains some number $n$ such that no other number $m$ in the set satisfies $m \mid n$.

Let's work through some examples to see if we can spot a trend. If we take a set like { 2, 3, 5, 7, 11, 13 }, then every element is minimal, since all the numbers are prime. In the set { 4, 6, 10 }, again all elements are minimal, since none divides any of the others. In the set { 2, 3, 4, 5, 6 }, the numbers 2, 3, and 5 are all minimal. In the set { 3, 9, 27, 81, 243 }, then only 3 is minimal.

Notice that in all of the above cases, the least element of the set (with respect to $<$) is always a minimal element with respect to $|$. Intuitively, this makes sense. If you'll recall from Chapter Three, we proved the following result:

***Theorem:*** If $m, n \in \mathbb{N}$, and $n \neq 0$, and $m \mid n$, then $m \leq n$.

As a consequence of this theorem, we would expect that the smallest number in the set would also be minimal, since there wouldn't be any smaller numbers left in the set to divide it. However, it turns out that this is not quite right. Consider, for example, the set { 0, 1, 2, 3 }. Here, the least element is 0. However, 0 is not a minimal element with respect to $|$, since every natural number divides 0 (since $0 \cdot m = 0$ for any $m \in \mathbb{N}$). Although 0 is not a minimal element of the above set, 1 is a minimal element of this set. So to be more precise – we'll say that the least nonzero element of the set will be a minimal element with respect to $|$.

We can formalize this intuition below:

***Theorem:*** $|$ is a well-founded order over $\mathbb{N}$.

***Proof:*** We will show that any nonempty set $S \subseteq \mathbb{N}$ has a minimal element with respect to $|$. Consider any nonempty set $S \subseteq \mathbb{N}$. We consider two cases. First, if $S = \{0\}$. Then 0 is a minimal element of $S$, since it's the only element of $S$.

Otherwise, $S \neq \{0\}$. Let $T = \{ n \in S \mid n \neq 0 \}$ be the set of nonzero numbers in $S$. Since $T \subseteq S$ and $S \subseteq \mathbb{N}$, by transitivity we have that $T \subseteq \mathbb{N}$. Since $S$ is nonempty and not identically $\{0\}$, it must contain at least one nonzero element, and therefore $T \neq \varnothing$. Since $\leq$ is a well-order over $\mathbb{N}$, there must be a least element of $T$ with respect to $\leq$; call it $n_0$. By our construction of $T$, we know that $n_0 \neq 0$.

We claim that $n_0$ is a minimal element of $S$ with respect to $|$. To see this, consider any $n \in S$ with $n \neq n_0$. We will prove that $n \nmid n_0$. We consider two cases:

*Case 1:* $n = 0$. We claim that $0 \nmid n_0$ and proceed by contradiction; suppose that $0 \mid n_0$. This means that there is some $q \in \mathbb{N}$ such that $0 \cdot q = 0 = n_0$. However, $0 \neq n_0$. We have reached a contradiction, so our assumption was wrong. Thus $0 \nmid n_0$.

*Case 2:* $n \neq 0$. We again claim that $n \nmid n_0$ and proceed by contradiction; suppose that $n \mid n_0$. Note that since $n \neq 0$ and $n \in S$, we know that $n \in T$. By our theorem from Chapter Three, we know that since $n_0 \neq 0$ and $n \mid n_0$, that $n \leq n_0$. Since $n_0$ is the least element of $T$ with respect to $\leq$, we know that $n_0 \leq n$. Since $n \leq n_0$ and $n_0 \leq n$, we know $n = n_0$. But this contradicts the fact that $n \neq n_0$. We have reached a contradiction, so our assumption was wrong and therefore $n \nmid n_0$.

In either case, $n \nmid n_0$, so $n_0$ is a minimal element of $S$ with respect to $|$. ■

Take a minute to read over this proof and look at just how many techniques we've employed here. We've used proof by cases and by contradiction. We've relied on the well-ordering of $\mathbb{N}$ by $\leq$ and used antisymmetry of $\leq$. And, we've pulled in a theorem from a few chapters back. Real mathematics always works by building incrementally off of previous techniques, and this proof (though not especially deep) is a great example of just how this can be done.

## 5.4.3 Well-Ordered and Well-Founded Induction

In Chapter Three, we proved the well-ordering principle (that $\mathbb{N}$ is well-ordered by $\leq$) using strong induction as a starting point. One of the chapter exercises asked you to then prove the principle of mathematical induction using the well-ordering principle as a starting point. This suggests that there is a close connection between well-ordering and mathematical induction. It turns out that this connection is surprisingly deep. As you'll see in this section, it's possible to generalize proof by induction from working purely on the natural numbers to working on arbitrary well-ordered or well-founded sets. This generalization of induction is extremely powerful and is used frequently in theoretical computer science and the analysis of algorithms.

To motivate this section, let us briefly review strong induction. If you'll recall, the principle of strong induction is specified as follows:

***Theorem (The Principle of Strong Induction):*** Let $P(n)$ be a property that applies to natural numbers. If the following are true:

$P(0)$ is true.
For any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true,
then $P(n)$ is true.

Then for any $n \in \mathbb{N}$, $P(n)$ is true.

Intuitively, strong induction works as follows. First, prove that some property holds for 0. Then, assuming the property holds for all values less than $n$, prove that it holds for $n$ as well.

As you saw in Chapter Three, it's possible to restate strong induction even more simply. Since the claim "for all $n' \in \mathbb{N}$ with $n' < n$, we know $P(n)$ is true" is vacuously true when $n = 0$, we were able to unify the two parts of a strong induction proof down into a single unified statement, which is shown here:

***Theorem (The Principle of Strong Induction):*** Let $P(n)$ be a property that applies to natural numbers. If for any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true, then $P(n)$ is true, then we can conclude that for any $n \in \mathbb{N}$, $P(n)$ is true.

Let's now take a minute to see exactly why this type of proof works. We'll use as our starting point the fact that $\mathbb{N}$ is well-ordered by $<$ (and also by $\leq$). Let's suppose that we find some property $P(n)$ that satisfies the requirement outlined by strong induction. Why must it be true that $P(n)$ holds for all $n \in \mathbb{N}$? Well, let's think about what would happen if it weren't true for some value. We could think about constructing the following set:

$$S = \{ n \in \mathbb{N} \mid P(n) \text{ is false} \}$$

This is a set of natural numbers. If we assume that there is some $n \in \mathbb{N}$ for which $P(n)$ doesn't hold, then we know that $S$ must be nonempty. Since $S$ is a nonempty set of natural numbers, by the well-ordering principle we know that $S$ must have a least element; let's call it $n_0$. Since $n_0$ is the least element of the set, we know that for all $n \in \mathbb{N}$ with $n < n_0$, that $P(n)$ must be true (otherwise, $P(n)$ would be false for some $n < n_0$, contradicting the fact that $n_0$ is the least natural number for which $P(n)$ fails. But now we have a problem – since $P(n)$ holds for all $n < n_0$, we know that $P(n_0)$ must also be true, contradicting the fact that it doesn't hold. We've arrived at a contradiction, so something must be wrong here. Specifically, it has to our assumption that $P(n)$ didn't hold for some $n \in \mathbb{N}$. Therefore, $P(n)$ must hold for all $n \in \mathbb{N}$.

We can formalize this proof here:

***Theorem (The Principle of Strong Induction):*** Let $P(n)$ be a property that applies to natural numbers. If for any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true, then $P(n)$ is true, then we can conclude that for any $n \in \mathbb{N}$, $P(n)$ is true.

***Proof:*** Let $P(n)$ be a property such that for any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true, then $P(n)$ is also true. We will prove that for all $n \in \mathbb{N}$, that $P(n)$ holds.

Consider the set $S = \{ n \in \mathbb{N} \mid P(n) \text{ is false} \}$. Note that this set is empty iff for all $n \in \mathbb{N}$, $P(n)$ is true. Also note that $S \subseteq \mathbb{N}$. So suppose for the sake of contradiction that $S$ is nonempty. Since $\leq$ is a well-ordering over $\mathbb{N}$, this means that there must be some least element $n_0 \in S$. Since $n_0 \in S$, we know $P(n_0)$ is false.

Since $n_0$ is the least element of $S$, for all $n \in \mathbb{N}$ satisfying $n < n_0$, we have that $P(n)$ holds; otherwise, we would have that $n \in S$ and $n < n_0$, contradicting the fact that $n_0$ is the least element of $S$. By our choice of $P(n)$, since for all $n \in \mathbb{N}$ with $n < n_0$ $P(n)$ holds, we have that $P(n_0)$ must hold. But this is impossible, since we know $P(n_0)$ does not hold.

We have reached a contradiction, so our assumption must have been wrong. Thus $P(n)$ holds for all $n \in \mathbb{N}$. ■

Look over this proof and look very carefully at what properties of the natural numbers we actually used here. Did we use the fact that you can add, subtract, and divide them? Did we use the fact that some numbers are prime and others aren't? Nope! In fact, all we needed for the above proof to work is that $\mathbb{N}$ is well-ordered by the $<$ relation.

This brings us to a key question. In the above proof, we only needed to use the fact that $\mathbb{N}$ was well-ordered to get induction to work. Could we therefore generalize induction to work over arbitrary well-ordered sets?

It turns out that is indeed possible to do this, and in fact we can scale up induction so that we can apply it to any well-ordered set, not just the natural numbers. To do so, let's revisit how strong induction works. The principle of strong induction says that if

> If for any $n \in \mathbb{N}$, if for all $n' \in \mathbb{N}$ with $n' < n$ we know $P(n')$ is true, then $P(n)$ is true,
> then we can conclude that for any $n \in \mathbb{N}$, $P(n)$ is true.

Suppose that we replace the use of $<$ and $\mathbb{N}$ with some arbitrary well-founded strict order $R$ over an arbitrary set $A$. What would happen in this case? If we do this, we end up with the following statement:

> If for any $x \in A$, if for all $x' \in A$ with $x'Rx$ we know $P(x')$ is true, then $P(x)$ is true,
> then we can conclude that for any $x \in A$, $P(x)$ is true.

In other words, suppose that we know that for every element $x$ of the well-ordered set, that whenever $P$ holds for all the elements less than $x$ (according to $R$), it must also be the case that $P(x)$ holds. If this is true, we are guaranteed that $P(x)$ must hold for every element $x$ of the well-ordered set.

Why would this be true? Well, let's think about the least element of $A$, which must exist because $A$ is well-ordered. Since there are no elements less than the least element of $A$, the statement "$P$ holds for all elements less than the least element" is vacuously true. Consequently, we can conclude that $P$ holds for the least element of $A$. This means that the second-least element must also have property $P$, then the third-least, the

fourth-least, etc. This sounds a lot like how normal induction works, except that this time there is no dependency at all on the natural numbers.

We can formalize this reasoning here:

***Theorem (The Principle of Well-Ordered Induction):*** Let $<$ be a strict well-ordering over set $A$, and let $P(x)$ be a property that applies to elements of $A$. If for any $x \in A$, if for all $x' \in A$ with $x'<x$ we know $P(x')$ is true, then $P(x)$ is true, then we can conclude that for any $x \in A$, $P(x)$ is true.

***Proof:*** Let $<$ be a strict well-ordering over $A$ and let $P(n)$ be a property such that for any $x \in A$, if for all $x' \in A$ with $x'Rx$ we know $P(x')$ is true, then $P(x)$ is also true. We will prove that for all $x \in A$, that $P(x)$ holds.

Consider the set $S = \{ x \in A \mid P(x) \text{ is false} \}$. Note that this set is empty iff for all $x \in A$, $P(x)$ is true. Also note that $S \subseteq A$. So suppose for the sake of contradiction that $S$ is nonempty. Since $<$ is a well-ordering over $A$, this means that there must be some least element $x_0 \in S$. Since $x_0 \in S$, we know $P(x_0)$ is false.

Since $x_0$ is the least element of $S$, for all $x \in A$ satisfying $x<x_0$, we have that $P(x)$ holds; otherwise, we would have that $x \in S$ and $x<x_0$, contradicting the fact that $x_0$ is the least element of $S$. By our choice of $P(x)$, since for all $x \in A$ with $x<x_0$ $P(x)$ holds, we have that $P(x_0)$ must hold. But this is impossible, since we know $P(x_0)$ does not hold.

We have reached a contradiction, so our assumption must have been wrong. Thus $P(x)$ holds for all $x \in A$. ■

This is pretty much exactly the same proof that we had for strong induction, except that we've generalized it to work over arbitrary well-ordered sets, not just $\mathbb{N}$. This generalization of induction is extremely powerful and has numerous uses in computer science. We'll see one of them in the next section.

Before concluding, I should mention that it's possible to generalize this result even further to apply to arbitrary well-*founded* sets, not just well-ordered sets. This is the principle of well-founded induction:

***Theorem (The Principle of Well-Founded Induction):*** Let $<$ be a well-founded strict order over set $A$, and let $P(x)$ be a property that applies to elements of $A$. If for any $x \in A$, if for all $x' \in A$ with $x'<x$ we know $P(x')$ is true, then $P(x)$ is true, then we can conclude that for any $x \in A$, $P(x)$ is true.

The proof of this principle is similar to the above proof of well-ordered induction, and is left as an exercise at the end of the chapter.

#### 5.4.3.1 Application: The Ackermann Function

To see the power of well-ordered induction, we will consider one application of this new type of induction. If you'll recall from Chapter 3, there is a close connection between induction and recursion. Specifically, using induction, it's possible for us to prove certain properties of recursive functions.

One interesting aspect of recursive functions that we did not yet explore was determining whether a recursive definition actually defines a function. For example, consider the following two recursive function definitions, which are meant to be applied to natural numbers:

$$f(x) = \begin{cases} 1 & \text{if } x = 0 \\ x \cdot f(x-1), & \text{otherwise} \end{cases} \qquad g(x) = \begin{cases} 1 & \text{if } x = 0 \\ g(2x), & \text{otherwise} \end{cases}$$

Let's look at the definition of this function $f$ on the left. We can try plugging in a few values:

$$f(0) = 1$$

$$f(1) = 1 \cdot f(0) = 1 \cdot 1 = 1$$

$$f(2) = 2 \cdot f(1) = 2 \cdot 1 = 2$$

$$f(3) = 3 \cdot f(2) = 3 \cdot 2 = 6$$

With a bit of inspection we can see that this recursive function definition defines the factorial function. That is, $f(n) = n!$. We could formally prove this with induction if we wanted to, though in the interests of time we'll skip it for now.

What about this second function? Well, if we plug in 0, we get $g(0) = 1$ by definition. But what happens if we try to compute $g(1)$? Well, then we see that

$$g(1) = g(2) = g(4) = g(8) = g(16) = \ldots$$

This chain of values never converges to anything. We keep trying to define $g(n)$ in terms of $g(2n)$, which is in turn defined in terms of some other value of the function. If we were to try to evaluate this function on a computer, we'd get either an infinite loop or a stack overflow, since the computer would keep trying to evaluate the function on larger and larger arguments. The issue here is that the recursive definition of $g(n)$ does not actually define a function at all. The "function" described simply isn't defined for any $n > 0$. As a result, we can't really call $g$ a function at all.

This gets at a deeper truth about recursive function definitions. We cannot just write out any recursive definition that we'd like and expect it to magically correspond to a function. It is extremely possible for a seemingly reasonable recursive definition to end up not giving values to functions at specific arguments. This observation motivated a search at the start of the twentieth century for a better understanding of recursive function definitions. What does a recursive function definition even mean, anyway? Surprisingly, the search for the answer to this question led to the rise of computer science as an academic discipline. Later on, in Chapter 14, we'll see some of the results of this search.

In the meantime, what has any of this got to do with induction? Here's the idea. If we're given an arbitrary recursive definition, we might be interested in determining whether it really does describe a function that's well-defined over all natural numbers. Since the functions are defined recursively, we can try to reason about them inductively.

To motivate this discussion, let's consider the following recursive definition:

$$h(x) = \begin{cases} 0 & \text{if } x = 0 \\ x^{h(x-1)}, & \text{otherwise} \end{cases}$$

If we plug in some values for $h$, we get back the following:

$$h(0) = 0$$

$$h(1) = 1^{h(0)} = 1^0 = 1$$

$$h(2) = 2^{h(1)} = 2^1 = 2$$

$$h(3) = 3^{h(2)} = 3^2 = 9$$

$$h(4) = 4^{h(3)} = 4^9 = 262{,}144$$

$$h(5) = 5^{h(4)} = 5^{262{,}144} \approx 6.2 \times 10^{183{,}230}$$

This function grows *extremely* quickly. As you can see, the value for $h(5)$ is so large that we have to express it in scientific notation. It has over one hundred thousand digits! The value of $h(6)$ is so huge that it's difficult to write it out without a tower of exponents:

$$h(6) = 6^{5^{262{,}144}}$$

Although we can evaluate this function at 0, 1, 2, 3, 4, 5, and 6, are we sure that this recursive definition even gives us a function at all? Intuitively, it seems like it should, since we know it's defined for 0, and that $h(n)$ is defined in terms of $h(n - 1)$ from that point forward. Consequently, we ought to be able to prove by induction that the function actually evaluates to a natural number at each point, even if that natural number is so staggeringly huge we'd have trouble representing it in a simple form.

We can formalize this reasoning below in the following proof:

***Theorem:*** The recursive definition of $h$ defines a function on $\mathbb{N}$.

***Proof:*** We will prove that for any $n \in \mathbb{N}$, that $h(n)$ is a natural number. To do so, we proceed by induction. Let $P(n)$ be "$h(n)$ is a natural number." We prove $P(n)$ holds for all $n \in \mathbb{N}$ by induction on $n$.

As our base case, we prove $P(0)$, that $h(0)$ is a natural number. Note that by definition, $h(0) = 0$, which is a natural number. Thus $P(0)$ holds.

For the inductive step, assume that for some $n \in \mathbb{N}$ that $P(n)$ holds. This means that $h(n)$ is a natural number; let's call that number $k$. We will prove $P(n + 1)$, meaning that $h(n + 1)$ is a natural number. To do this, note that $n + 1 \geq 1$ for all $n \in \mathbb{N}$. This means that by definition of $h(n)$, we have that $h(n + 1) = (n + 1)^{h(n)} = (n + 1)^k$. Since $n + 1$ and $k$ are natural numbers, $(n + 1)^k$ is a natural number. Consequently, $h(n + 1)$ evaluates to a natural number, so $P(n + 1)$ holds, completing the induction. ■

An interesting detail in this proof is that we have shown that $h(n)$ is always a natural number, though we haven't actually said what that natural number is! This sort of proof just guarantees that if we want to evaluate $h(n)$, it's always possible to do so, even though it doesn't tell us anything about what the values will be.

So what does any of this have to do with well-ordered induction? We were able to prove that $h$ is a legal function without using any fancy induction. However, there might be other recursive definitions that do define functions, but which can't easily be shown to be functions using normal induction. A classic example of this is the *Ackermann function*, which is recursively defined as follows:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0, n = 0 \\ A(m-1, A(m, n-1)) & \text{otherwise} \end{cases}$$

This recursive definition is different from what we've seen before in that it's a function of *two* variables, $m$ and $n$, rather than just one. Moreover, just looking over this definition, it's not at all clear that this necessar-

ily even defines a function at all. If we try plugging in some values for $m$ and $n$, it's hard to see that $A(m, n)$ exists for even simple values of $m$ and $n$:

$$A(4, 4) = A(3, A(4, 3)) = A(3, A(3, A(4, 2))) = A(3, A(3, A(3, A(4, 1)))) = \ldots$$

If you're having trouble reading this, don't worry. The recursion is extraordinarily complicated, and almost impossible to trace through by hand. In fact, if we were to trace through it, we'd find that the value of $A(4, 4)$ is

$$2^{2^{2^{2^{2^{2^{2}}}}}} - 3$$

That's a tower of seven 2's all raised to each other's power, minus 3. This value so large that we can't write it in scientific notation without using nested exponents. You should be glad that we didn't try evaluating $A(5, 5)$. This value is so unimaginably huge that we can't even write it out as a tower of nested exponents, since doing so would require more space than fits in the known universe. In fact, mathematicians have had to invent their own special notations to describe just how huge this value is.

Given that the values of this function increase so rapidly that we soon can't even write them out any more, how can we possibly be sure that this even defines a function at all? That is, why is it called the Ackermann *function* and not just the Ackermann *recurrence*? It turns out that despite the fact that these values grows astronomically quickly (which is actually a bit of an understatement, since for $n = 4$ the function already exceeds the number of atoms in the known universe), the recursive definition of $A$ does indeed define a function.

How on earth are we supposed to prove this? To do so, we can proceed in a similar fashion as when we proved that $h(n)$ was a function – we'll show that for any $m$ and $n$, that $A(m, n)$ is a natural number. We don't care how huge of a natural number it is; just that it's a natural number. When we used this argument for $h(n)$, we used normal induction over $\mathbb{N}$, since the arguments to $h$ were natural numbers. Since $A$ takes in two parameters, we'll use induction over $\mathbb{N}^2$, since the arguments to $A$ are pairs of natural numbers. Of course, we can't just use induction over $\mathbb{N}^2$, since induction only works over $\mathbb{N}$. However, we can use well-ordered induction over $\mathbb{N}^2$, provided that we can come up with a well-ordering of $\mathbb{N}^2$.

Fortunately, we happen to have such a well-ordering. Recall that $\mathbb{N}^2$ is just a shorthand for $\mathbb{N} \times \mathbb{N}$, and we happen to have a well-ordering of $\mathbb{N}$ lying around, namely $<$. Consequently, as we saw earlier in this section, if we take the lexicographical ordering $<_{lex}$ defined by combining $<$ over $\mathbb{N}$ with itself, we will end up with a well-ordering. Consequently, let's see if we can prove that the Ackermann function actually is indeed a function by using well-ordered induction over $\mathbb{N}$, using $<_{lex}$ as our well-ordering.

Let's let the property $P$ be $P(m, n) \equiv$ "$A(m, n)$ is a natural number." We'll use well-ordered induction to prove that $P(m, n)$ holds for all $(m, n) \in \mathbb{N}^2$. Looking over how well-ordered induction works, we will proceed as follows:

- First, we assume that for some $(m, n) \in \mathbb{N}^2$, that for all $(m', n') <_{lex} (m, n)$, that $P(m', n')$ holds.
- Next, we prove that under this assumption, $P(m, n)$ holds.

All that's left to do now is to go case-by-case through the definition of $A(m, n)$ proving that each possibility leads to the computation of a natural number, under the assumption that $A(m', n')$ is a natural number when $(m', n') <_{lex} (m, n)$ (that is, when either $m' < m$, or when $m' = m$ and $n' < n$). The resulting proof is shown below:

***Theorem:*** The Ackermann function $A(m, n)$, defined below, is a function over $\mathbb{N}^2$:

$$A(m,n)=\begin{cases} n+1 & \text{if } m=0 \\ A(m-1,1) & \text{if } m>0, n=0 \\ A(m-1,A(m,n-1)) & \text{otherwise} \end{cases}$$

***Proof:*** We will prove that $A(m, n)$ is a natural number for any $(m, n) \in \mathbb{N}^2$. To do this, we proceed by well-ordered induction. Let $P(m, n)$ be "$A(m, n)$ is a natural number" and let $<_{lex}$ be the lexicographical ordering of $\mathbb{N}^2$ by the first component, then the second component. We will prove that $P(m, n)$ holds for all $(m, n) \in \mathbb{N}^2$ by well-founded induction.

Assume that for some $(m, n) \in \mathbb{N}^2$ that for all $(m', n') \in \mathbb{N}^2$ such that $(m', n') <_{lex} (m, n)$, we have that $P(m', n')$ holds and that $A(m', n')$ is a natural number. Under this assumption, we will prove that $P(m, n)$ holds, meaning that $A(m, n)$ is a natural number. To do so, we consider three cases for the values of $m$ and $n$:

*Case 1:* $m = 0$. In this case, by definition, $A(m, n) = n + 1$, which is a natural number.

*Case 2:* $m \neq 0$, but $n = 0$. In this case, by definition, $A(m, n) = A(m - 1, 1)$. Note that $(m - 1, 1) <_{lex} (m, n)$, since $m - 1 < m$. Consequently, by our inductive hypothesis, we know that $A(m - 1, 1)$ is a natural number; call it $k$. Thus $A(m, n) = A(m - 1, 1) = k$, so $A(m, n)$ is a natural number.

*Case 3:* $m \neq 0$, and $n \neq 0$. In this case, by definition, $A(m, n) = A(m - 1, A(m, n - 1))$. First, note that $(m, n - 1) <_{lex} (m, n)$. By our inductive hypothesis, this means that $A(m, n - 1)$ is a natural number; call it $k$. Thus

$$A(m, n) = A(m - 1, A(m, n - 1)) = A(m - 1, k)$$

Next, note that regardless of the value of $k$, that $(m - 1, k) <_{lex} (m, n)$, since $m - 1 < m$. Consequently, by our inductive hypothesis, we know that $A(m - 1, k)$ is some natural number; call it $r$. Thus $A(m, n) = A(m - 1, k) = r$, which is a natural number.

Thus in all three cases, we have that $A(m, n)$ is a natural number. Thus $P(m, n)$ holds, completing the induction. ■

# 5.5 Chapter Summary

- An *n-tuple* is an ordered list of $n$ elements.
- The *Cartesian product* of two sets is the set of all pairs of elements drawn from those sets. It can be generalized to apply to multiple different sets at once.
- The *Cartesian power* of a set is the many-way Cartesian product of that set with itself.

- A *relation* over a group of sets is a subset of their Cartesian product. Intuitively, it represents some property that may hold of groups of elements from those sets.
- A *binary relation* over a set is a relation over the Cartesian square of that set.
- A binary relation is called *reflexive* if it relates every object to itself. A binary relation is called *irreflexive* if it *never* relates an object to itself.
- A binary relation is called *symmetric* if whenever two objects are related by the relation, they are also related in the opposite direction by that relation. A binary relation is called *asymmetric* if whenever two objects are related in one direction, they are not related in the other direction. A binary relation is called *antisymmetric* if any time two *different* objects are related in one direction, they are not related in the other direction.
- A binary relation is called *transitive* if any time a chain of elements are related by the relation, the first and last elements of that chain are related.
- An *equivalence relation* is a relation that is reflexive, symmetric, and transitive.
- Equivalence relations induce a *partition* of their underlying set into different *equivalence classes*. The set of these equivalence classes is called the *quotient set*.
- A *strict order* is a relation that is irreflexive, asymmetric, and transitive. A *partial order* is a relation that is reflexive, antisymmetric, and transitive. These two relations are collectively called *order relations*.
- A *total order* is a partial order that can rank any pair of elements. A *strict total order* is a strict order that can rank any pair of distinct elements.
- A *Hasse diagram* is a visual representation of an ordering relation that omits relations between objects that can be inferred from the properties of ordering relations.
- Given an order relation $R$ and a set $A$, the *greatest element* or *least element* of $A$, if one exists, is the element that is at least as large (or as small) as every object in the set. A *maximal element* or *minimal element* is an element that is not smaller than (or not greater than) any other element of the set.
- A *preorder* is a relation that is reflexive and transitive. From a preorder, it is possible to derive an equivalence relation over elements that are mutually comparable and a partial order over the equivalence classes of that equivalence relation.
- The *product ordering* of two ordered sets is the relation over the Cartesian product of those sets that holds when the corresponding elements of the pairs are less than one another.
- The *lexicographical ordering* of two ordered sets is the relation over their Cartesian product that holds when the first elements of the pairs are related, or when they are unrelated and the second element is related.
- An ordered set is called *well-ordered* if every nonempty subset of that set contains a least element. An ordered set is called *well-founded* if every nonempty subset of that set contains a minimal element.
- Induction can be generalized to apply to well-ordered or well-founded sets.

## 5.6 Chapter Exercises

1. Is $\in$ a strict order? A partial order? An equivalence relation? A preorder?
2. Let $G = (V, E)$ be a directed graph and let $\to$ be the reachability relationship in $G$ (that is, $x \to y$ iff there is a path from $x$ to $y$). Is $\to$ a strict order? A partial order? A preorder? An equivalence relation?
3. Prove that a binary relation $R$ over a set $A$ is a strict order iff it is irreflexive and transitive.
4. Prove that a binary relation $R$ over a set $A$ is a strict order iff it is asymmetric and transitive.
5. Prove that a binary relation $R$ over a set $A$ is a strict total order iff it is trichotomous and transitive.
6. Prove that if $R$ is an asymmetric relation over $A$, then it is antisymmetric.
7. Give an example of an antisymmetric relation $R$ over a set $A$ that is not asymmetric.
8. Give an example of a relation that is both symmetric and antisymmetric.
9. Prove that a binary relation $R$ over a set $A$ is a total order iff it is total, antisymmetric, and transitive.
10. Suppose that $R$ is a strict order over a set $A$. Define the relation $R'$ as follows: $xR'y$ iff $xRy$ or $x = y$. Prove that $R'$ is a partial order.
11. Suppose that $R$ is a partial order over a set $A$. Define the relation $R'$ as follows: $xR'y$ iff $xRy$ and $x \neq y$. Prove that $R'$ is a strict order.
12. Let $R$ be a strict order over a set $A$. Prove that $A$ can have at most one greatest element.
13. Let $A$ be a set and let $\leq_A$ be a partial order over $A$. We say that a sequence $x_1, \ldots, x_n$ is sorted iff $x_1 \leq_A x_2 \leq_A \ldots \leq_A x_n$. Prove that any subset of $n$ elements of $A$ can be sorted iff $A$ is a total order.
14. Let $\leq_A$ be a partial order over $A$. Define the relation $<_A$ over $A$ as follows: $x <_A y$ iff $x \leq_A y$ and $x \neq y$. Prove that $<_A$ is a strict order over $A$.
15. Let $<_A$ be a strict total order over $A$. Define the relation $\leq_A$ over $A$ as follows: $x \leq_A y$ iff $y \not<_A x$. Prove that $\leq_A$ is a total order over $A$.
16. Let $R$ be a transitive relation over the set $A$. Prove that in the graphical representation of the relation (that is, the graph $(A, R)$), that $(u, v) \in R$ iff $v$ is reachable from $u$.
17. How many equivalence relations are there over the set $\{ 1, 2, 3 \}$?
18. How many partial order relations are there over the set $\{ 1, 2, 3 \}$? ★
19. Are there any binary relations over $\mathbb{N}$ that are both equivalence relations and total orders? If so, give an example of one and prove why it is both an equivalence relation and a total order. If not, prove why not.
20. When dealing with binary relations over infinite sets, it can be easy to accidentally conclude that a property holds of the relation that, while true for finite subsets of the infinite set, does not actually hold for the infinite set itself.
    1. Let $A$ be an infinite set and $R$ be a binary relation over $A$. Suppose that for every finite subset $A' \subseteq A$, that $R$, restricted to $A'$, is a total order. Does this necessarily mean that $A$ is a total order? If so, prove it. If not, find a counterexample.

2. Let $A$ be an infinite set and $R$ be a binary relation over $A$. Suppose that for every finite subset $A' \subseteq A$, that $R$, restricted to $A'$, is a well order. Does this necessarily mean that $A$ is a well order? If so, prove it. If not, find a counterexample.

21. Let $S$ be any set and $X_1$ and $X_2$ be partitions of $S$. We say that $X_1$ is *finer* than $X_2$, denoted $X_1 \leq X_2$, iff every set $S_i \in X_1$ is a subset of some set $T_j \in X_2$.

    Prove that the binary relation $\leq$ defined this way over the set of all partitions of $S$ is a partial order.

22. Let $\sim_A$ be an equivalence relation over $A$ and $\sim_B$ be an equivalence relation over $B$. Consider the following two relations over the set $A \times B$:

    $\sim_{OR}$, defined as follows: $(a_1, b_1) \sim_{OR} (a_2, b_2)$ iff $a_1 \sim_A a_2$ or $b_1 \sim_B b_2$.

    $\sim_{AND}$, defined as follows: $(a_1, b_1) \sim_{AND} (a_2, b_2)$ iff $a_1 \sim_A a_2$ and $b_1 \sim_B b_2$.

    Are either of these relations equivalence relations? If so, prove which ones are equivalence relations.

23. Let's define the set $\Sigma = \{ a, b, c, \ldots, z \}$ of all English letters (the reason for the notation $\Sigma$ here will become clearer later on when we discuss formal languages). A *string* of symbols from $\Sigma$ is a finite sequence of letters. For example, "math" and "about" are both strings. We'll denote the *empty string* of no letters using the symbol $\varepsilon$ (the Greek letter epsilon). We can then let $\Sigma^*$ be the set of all strings made from letters in $\Sigma$. Formally, we'll say that $\Sigma^* = \{ w \mid w \text{ is a string of letters in } \Sigma \}$

    The normal way that we sort English words is called the *lexicographical ordering* on strings. In this ordering, we proceed across the characters of two words $w_1$ and $w_2$ from the left to the right. If we find that the current character of one word precedes the current character of the other, then we decide that the first word precedes the second word. For example, "apple" comes before "azure." Otherwise, if we exhaust all characters of one word before the other, we declare that this word comes first. For example, "ha" precedes "happy."

    Is the lexicographical ordering on strings a well-ordering? If so, prove it. If not, find a set of strings that contains no least element.

24. Let $(A, <_A)$ and $(B, <_B)$ be strictly ordered sets. As we saw in this chapter, two ways that we can construct strict orders over $A \times B$ from these ordered sets are the product construction and the lexicographical ordering. There is another ordering called the *antilexicographical ordering*, which is simply the lexicographical ordering, but with the second elements compared first, rather than the first elements. Besides these three orderings, are there any other strict orderings that can be defined over $A \times B$?

25. The $\mid$ relation over $\mathbb{N}$ has a greatest element. What is it?

26. Let $A$ be a finite set and let $R$ be a total order over $A$. Prove that $R$ is a well-order of $A$.

27. Let $(A_1, R_1)$ and $(A_2, R_2)$ be two well-ordered posets with $A_1$ and $A_2$ disjoint. Now, consider the new relation $R$ defined on $A_1 \cup A_2$ defined as follows:

    $xRy$ iff $x, y \in A_1$ and $xR_1y$, or $x, y \in A_2$ and $xR_2y$, or $x \in A_1$ and $x \in A_2$.

    Prove that $(A_1 \cup A_2, R)$ is a poset and that it is well-ordered.

28. Prove the principle of well-founded induction.

29. Prove that if $(A_1, R_1)$ and $(A_2, R_2)$ are two strict, well-founded sets, then the lexicographical ordering $R_{lex}$ over $A_1 \times A_2$ is also well-founded.

30. Prove that the relation $\subseteq$ over $\wp(\mathbb{N})$ is not well-founded. ★

31. Prove that the following recursive definition defines a function. What function is it? ★

$$f(m,n) = \begin{cases} n & \text{if } m = 0 \\ 1 + f(0,n) & \text{if } m = 1 \\ 1 + f(m-2, n+1) & \text{otherwise} \end{cases}$$

32. Prove that the following relation over $\mathbb{N}^2$ is a preorder: $(a, b)R(c, d)$ iff $a + d < b + c$.

33. Consider the set $\mathbb{N}^2 / \sim_R$, where $R$ is the preorder from the previous problem. Do you notice anything interesting about those equivalence classes?

# Chapter 6 Functions and Cardinality

The preceding two chapters (Chapter 4 on graphs and Chapter 5 on relations) explored how different objects can relate to one another and how those relations can create larger structures. This chapter explores *transformations* that convert objects of one type into objects of another type. Interestingly, this chapter will not be focused so much on *how* those transformations are performed. Instead, we will study the effects of those transformations existing in the first place. The major questions we will explore in this chapter are the following:

- How do we mathematically represent transformations that turn objects into other objects?
- How can we combine or modify these transformations to produce new ones? In other words, how do we transform *transformations?*
- How can we use the existence of transformations to reason about the sizes of sets?

This last point may seem completely unrelated to the previous two, but it's extremely important. In fact, one of the major themes of this chapter will be the connection between functions, cardinality, and the nature of infinity. By the time you finish reading this chapter, you will have seen some of the baffling properties of infinity and infinite cardinalities, and will be well-equipped to reason about infinite sets.

## 6.1 Functions

The key object of study of this chapter is the *function*. Intuitively speaking, a function represents a way of transforming objects of one type into objects of some other type. Before we give a formal definition of functions or start studying them in depth, let's go over a few simple examples of functions:

- Suppose we have the set of all strings. We could define a function *length*($w$) which, given a string $w$, produces its length. For example, *length*(dikdik) = 6 and *length*(springbok) = 9. Interestingly, *length*(smile) = 5 and *length*(frown) = 5, so regardless of whether it's true that it takes more muscles to frown than to smile, it takes the same number of characters.
- Suppose that we have the set of all graphs. Then we could define a function *nodes*($G$), which takes in a graph $G$ and returns the set of all the nodes in $G$.
- The function *sin* $x$ transforms real numbers into real numbers in the range [-1, +1]. Specifically, it takes a real number and outputs the $y$ coordinate you would be at if you started at position (1, 0) on the unit circle, then moved a total of $x$ distance counterclockwise around the circle.
- Given any set $S$, we can define a function called the *identity function* that always outputs its input. This function essentially does no transformation at all.

When studying functions, there are a huge number of questions that we might want to discuss. We could study, for example, the means by which a function is computed. For example, how exactly would one evaluate *sin* $x$? Are there different ways of doing it, and are some worse than others? We could also discuss what the most compact way to represent a function is. As an example, consider the following recursive definition of a function:

$$f(a, 0) = 1$$

$$f(a, b + 1) = a \cdot f(a, b)$$

This function turns out to be equivalent to the function $f(a, b) = a^b$. It seems silly to write out the longer version of the function. On the other hand, the longer version of the function does have the nice property that it would be easy to implement on a computer that could add and multiply, but which could not actually do exponentiation natively.

Both of the above questions are interesting ones to study, and we will touch on them later in later chapters. However, for the purposes of this chapter, we will study a different set of questions. First, apart from any particular means of *implementing* a function, what exactly *is* a function? Second, based purely on what outputs a function produces, how can we classify functions into different types? Third, how can we build new functions from existing functions? Finally, how can we rigorously reason about the sizes of infinite sets? (That last question might seem to come out of nowhere, but don't worry. All will be revealed in due time. ☺)

Based on these questions, the definition of a function that we will explore in this chapter will have no dependence whatsoever with how the function could actually be implemented, much in the same way that our discussion of relations never considered how one would actually write a program that could determine whether a relation held between two objects. Later on, when we discuss computability and complexity theory, we will revisit these questions and discuss how hard some functions are to evaluate, and whether there are functions that mathematically are well-defined but could never be computed by any computing machine.

### 6.1.1 Basic Definitions

We'll begin our foray into functions by giving some basic terms and definitions that we will use to study functions.

Intuitively, a function is some operation that takes in an object of one type and produces an object of another type. For example, the function sin $x$ takes in real numbers and produces real numbers, while the function $|x|$, as applied to integers, takes in integers and produces natural numbers.

When describing functions, we typically talk about functions that map from one set of elements to another set of elements. Those sets can be anything – we can have functions from $\mathbb{R}$ to $\mathbb{Z}$, functions from the set of all strings to the set of all unicorns, functions from the set of all puppies to the set of all countries, etc. This gives us a lot of flexibility when discussing functions and leads to the following definitions:

> Let $A$ and $B$ be arbitrary sets. We say that $f$ is a ***function from A to B*** iff $f$ is a means of associating every element of $A$ with a single element in $B$. If $a \in A$, we denote the element in $B$ that $f$ associates it with as $f(a)$.

For example, consider the function *rev* from the set of all strings to the set of all strings. We'll define *rev*(*s*) to be the reverse of the string *s*. For example, *rev*(hi) = ih, and *rev*(racecar) = racecar. Note that this function is legally defined for every element of the input set and produces a single output.

As another example, let's let $f(x) = x - x^2$. This function $f$ will be a function from $\mathbb{N}$ to $\mathbb{Z}$ (though we could define it from $\mathbb{Z}$ to $\mathbb{Z}$ or from $\mathbb{R}$ to $\mathbb{R}$ as well; more on that later). This function provides of way of associating every natural number with some resulting integer. For example, $f(0) = 0$, $f(1) = 0$, $f(2) = -2$, $f(3) = -6$, etc. Notice that this function maps several different inputs to the same output. In particular, $f(0) = 0$ and $f(1) = 1$. This is not a problem: our definition of a function just says that every element of the input set (here, $\mathbb{N}$) has to be associated with a single element of the output set (here, $\mathbb{Z}$), not the other way around. Also note that not all elements of $\mathbb{Z}$ can be produced as the output of this function. For example, no input to this function will produce the output 137. This is also not a problem. Our definition says that every element

of the input set must map to some element of the output set, not that every element of the output set must have something that maps to it. We'll discuss both of these traits (that multiple inputs to $f$ produce the same output and that not all elements of the output set can be formed) later in this chapter.

On the other hand, some operations that might appear to be functions are not technically functions by our definition. For example, consider the "function" $\sqrt{x}$ from $\mathbb{R}$ to $\mathbb{R}$. Unfortunately, this does not define a function; specifically, $\sqrt{x}$ is undefined whenever $x < 0$. In order for some function $f$ to be a function from $A$ to $B$, $f$ has to be defined for *every* element of the domain. That said, we can claim that $\sqrt{x}$ is a function from the set of all nonnegative real numbers to the set of real numbers, since in that case every possible input to the function produces a valid output, or we could have it be a function from the real numbers to the complex numbers. We just can't have it go from reals to reals.

Up to this point we've been using the terms "input set" and "output set" to specify the set of all valid inputs to a function and outputs of a function, though these terms aren't mathematically standard. Mathematicians usually use two other terms to refer to these sets:

> If $f$ is a function from $A$ to $B$, then $A$ is called the ***domain*** of $f$ and $B$ is called the ***codomain*** of $f$. We denote the domain of $f$ by dom $f$ and the codomain of $f$ by cod $f$.

For example, the function $f(x) = x - x^2$ defined above from $\mathbb{N}$ to $\mathbb{Z}$ has $\mathbb{N}$ as its domain and $\mathbb{Z}$ as its codomain. Here, we've specified the codomain of $f$ to be the set $\mathbb{Z}$ even though not all elements of $\mathbb{Z}$ can be generated as possible outputs from $f$. This is totally fine – often, the codomain of a function is a strict superset of what the function can actually produce. Mathematicians sometimes use the term *range* to refer to the set of all possible outputs of a function, regardless of the codomain. We'll discuss the range of a function later on. For now, we will always refer to the set of elements that the function is permitted to choose from as outputs as the codomain, deferring the discussion of whether those elements are used until later.

We will need one final piece of notation before moving on. Often, we will need to write a statement to the effect of "pick an arbitrary function $f$ from $A$ to $B$" or "let $g$ be a function from $S$ to $T$." As a mathematical shorthand, we will use the following notation:

> If $f$ is a function from $A$ to $B$, we write $f : A \to B$.

This notation just describes the domain and codomain of $f$. It doesn't describe how the function is evaluated or what transformation is performed. However, this notation is quite useful, as it lets us more clearly describe what we expect a function to do.

When using this notation, it is common to specify the domain and codomain of a function along with what that function is. For example, we might say

"Consider $f : \mathbb{Z} \to \mathbb{N}$ defined as $f(x) = |x|$"

"Consider an arbitrary $g : A \to B$."

"Let $h : A \to C \cup D$ be chosen arbitrarily"

The first of these introduces a function from integers to natural numbers and then explicitly gives a rule for evaluating that function. The second of these definitions says that $g$ is a function from $A$ to $B$, and that this function is chosen arbitrarily. (This might appear in a proof that's showing a general property of functions

from $A$ to $B$, or it might be used to show that some result holds regardless of how $g$ is chosen). Similarly, the third definition states that $h$ is a function from $A$ to the set $C \cup D$, but leaves the choice of $h$ unspecified.

Of these three definitions, only the first one actually gives us a function we can explicitly evaluate. The other two don't actually define functions; they just say what the function's domain and codomain are.

### 6.1.2 Defining Functions

Up to this point, the functions we've defined have either been purely in English or using algebraic notation. Are these appropriate ways to define functions? Are there other ways? This section explores these questions.

A good general rule of thumb for defining a function is the following: the function definition should have enough detail to unambiguously describe

- The domain and codomain, and
- What the output of the function is for any input.

For example, here is a perfectly valid definition of a function $f$:

$$f : \mathbb{N} \to \mathbb{N}, \text{ where } f(n) = n^2.$$

This contains all the information we need to reason about the function. Both its domain and codomain are $\mathbb{N}$, and given any input natural number $n$, the output is the number $n^2$. On the other hand, this function definition is a bit less precise:

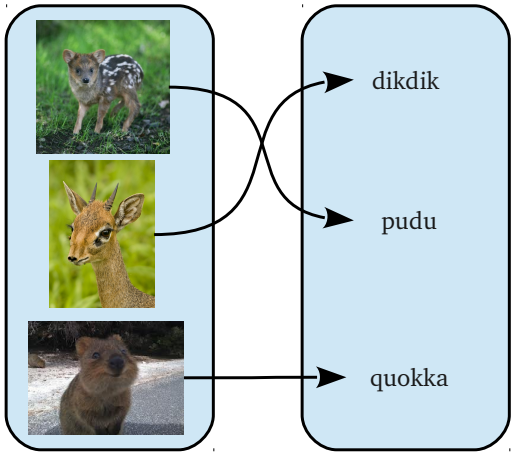
$$f(x) = x^2$$

Here, we don't know the domain and codomain. All of the following are totally valid interpretations of the domain and codomain:

$$f : \mathbb{N} \to \mathbb{N} \quad f : \mathbb{Z} \to \mathbb{N} \quad f : \mathbb{Q} \to \mathbb{Q}$$
$$f : \mathbb{Z} \to \mathbb{N} \quad f : \mathbb{Z} \to \mathbb{Q} \quad f : \mathbb{R} \to \mathbb{R}$$

In all six cases, the above function definition could define a totally valid function. However, the properties of that function are wildly different based on the domain. For example, if $f : \mathbb{N} \to \mathbb{N}$, then $f$ has the property that if $f(x) < f(y)$, then $x < y$. This isn't true when $f : \mathbb{Z} \to \mathbb{Z}$, since we have that $f(1) < f(-2)$, but $-2 < 1$. Consequently, when specifying a function by defining a rule, it is a good idea to specifically describe the domain and codomain.

In some cases, you will have a function $f : A \to B$ where $A$ and $B$ are finite sets (that is, they contain only finitely many elements). In these cases, you can often define the function just by drawing a picture of the elements of $A$, then elements of $B$, and what the function's output is for each input. For example, consider the following function from cute pictures of animals to the name of the animal in question:




Here, the domain and codomain of $f$ are specified by the left-hand and right-hand sides of the drawing, and the values of $f$ for each input are described via the arrows leading each input to its output. Drawings like these don't arise frequently in formal mathematics, but they provide an excellent intuition for many of the definitions and results that we will be proving. We will use them extensively in this chapter, and you're encouraged to draw out pictures like these when you are doing proofs of your own!

One more way that we will often define functions is by specifying a variety of different rules to apply to the input, giving conditions under which each rule should be applied. Functions of this sort are often called ***piecewise functions***, because they are defined one piece at a time.

As an example, suppose that we want to define the absolute value function $|x|$. This function is equal to $x$ when $x \geq 0$ and is equal to $-x$ otherwise. We could write this function out as a piecewise function as follows:

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{otherwise} \end{cases}$$

By definition, a function must map every possible input in the codomain to a *single* output. When defining a piecewise function, it is important to ensure the following:

- Every possible input falls into at least one of the cases, and
- If an input falls into multiple cases, each case produces the same output.

For example, consider the following alternative definition of the absolute value function:

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x \leq 0 \end{cases}$$

Here, when $x = 0$, both cases apply. In this case, this is totally fine, since the two cases produce the same value when $x = 0$. However, the following would ***not*** be a valid definition for the absolute value function:

$$|x| = \begin{cases} x & \text{if } x > 0 \\ -x & \text{if } x < 0 \end{cases}$$

because now the function is undefined when $x = 0$. Remember – functions must associate an output with every possible input in the domain![*]

---

* Technically speaking, we could claim that the above was a valid definition of a function $|x| : \mathbb{R} - \{0\} \to \mathbb{R}$. However, that would be moving the bullseye after we had taken the shot. This is why it's good to always define the domain and codomain first; any time you define a function, you can always double-check that it matches the domain and codomain that you intended it to have.

### 6.1.3 Functions with Multiple Inputs

The definition of functions that we've given above might seem incomplete if you have programming experience. When programming, it's often common to write functions like these:

```
int raiseToPower(int x, int y) {
    int result = 1;
    for (int i = 0; i < y; i++) {
        result *= x;
    }
    return result;
}
```

This function takes in two arguments. However, the mathematical functions we're studying right now, which have just a domain and codomain, only take in a single argument – namely, an element of the domain. Is this really a good formalism for functions when there are many interesting functions that take in multiple arguments?

It turns out that this isn't a problem at all. Let's take the above function as an example, assuming that we're only considering natural numbers as inputs. In this case, we can think of the above function, which appears to take in two arguments, as a function that takes in just one argument – an ordered pair of natural numbers. In other words, we can think of this two-argument function as really being a one-argument function, where that one argument consists of two separate pieces. Mathematically, we could consider defining this function as follows:

$$\textbf{raiseToPower} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \text{ where } \textbf{raiseToPower((x, y))} = \textbf{x}^{\textbf{y}}\textbf{.}$$

More generally, we can always model an $n$-argument function as a function containing a single argument, where that single argument is the $n$-way Cartesian product of an appropriate group of sets. For example, we might represent a function that adds together three real numbers and an integer this way:

$$\textbf{weirdSum} : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{N} \to \mathbb{R}, \text{ where } \textbf{weirdSum((x, y, z, w))} = \textbf{x + y + z + w}$$

Rather than writing out **raiseToPower((x, y))** or **weirdSum((x, y, z, w))**, we'll allow ourselves the notational shorthand of **raiseToPower(x, y)** and **weirdSum(x, y, z, w)**:

> If $f : A_1 \times \ldots \times A_n \to B$, then we denote $f((x_1, \ldots, x_n))$ by $f(x_1, \ldots, x_n)$.

### 6.1.4 A Rigorous Definition of a Function ★

In the previous chapters, we started with intuitive descriptions of some key mathematical object (in Chapter 4, the graph; in Chapter 5, the relation), and then built up a formal definition behind them. For example, we formally defined a graph as a pair $(V, E)$ of a set $V$ of vertices and set $E$ of edges, and we defined a binary relation $R$ over a set $A$ to be a subset of $A^2$ corresponding to the ordered pairs for which the relation holds. Before proceeding onward to explore properties of functions, let's take a minute to give a rigorous, formal definition of a function. We will not use this definition much later in the chapter, but it is good to know because it lets us handle some "edge cases" that arise when working with functions.

In a sense, a function is a very restricted type of a relation. A binary relation over $A$ and $B$ can relate any element of $A$ with as many elements of $B$ as it likes, even if it's zero times. A function has to relate every element of $A$ with just one element of $B$. In fact, we can define a function as a very restricted binary relation over $A$ and $B$ subject to the following constraint: for every element of $A$, there is exactly one element of $b$ that $a$ is related to. This special $b$ that $a$ is related to is what we traditionally think of as the output of the

function. Consequently, we can think of $f(a)$ as just being a shorthand for saying "the single $b$ that $a$ is related to by the function $f$."

This gives one possible definition of a function:

> A function $f : A \to B$ is a binary relation over $A$ and $B$ such that for every $a \in A$, there is exactly one $b \in B$ such that $afb$. We denote this unique choice of $b$ by $f(a)$.

An alternative (but totally equivalent) way of thinking of a function is as a set of ordered pairs, where each ordered pair of the form $(a, b)$ means "the output of $f$ when applied to $a$ yields $b$." Under this definition, the meaning of $f(a)$ is "the single choice of $b$ such that $(a, b) \in f$."

This set of ordered pairs is not arbitrary. For any choice of $a$ in the domain $A$, there has to be exactly one pair in the set whose first element is $a$. Otherwise, $f(a)$ might not be well-defined.

This gives rise to an alternative definition of functions:

> A function $f : A \to B$ is a set $f \subseteq A \times B$ such that for every $a \in A$, there is exactly one $b \in B$ such that $(a, b) \in f$. We denote this unique choice of $b$ by $f(a)$.

This definition might not look like the earlier definition involving relations, but it turns out to be completely equivalent due to how we define binary relations over multiple sets. One of the chapter exercises asks you to prove this.

Why bother coming up with definitions like these? In some cases, it's actually good to have this sort of definition to fall back on. For example, consider the following question: is it possible to have a function $f : \varnothing \to \mathbb{N}$? That is, can we have a function $f$ with the empty set as its domain and the natural numbers as its codomain.

Initially, this might seem preposterous. This function $f$ doesn't define a transformation at all, since there's nothing in the empty set that would be transformed! But let's look back at our definition of a function, specifically, the second definition, which defines a function $f : A \to B$ to be a subset of $A \times B$ with some special properties. In our case, we are curious if it's possible to define a function from $\varnothing$ to $\mathbb{N}$. This function would have to be a subset of $\varnothing \times \mathbb{N}$. As we saw earlier when defining the Cartesian product, the Cartesian product of any set and $\varnothing$ is the empty set. Therefore, if we want to find a function from $\varnothing$ to $\mathbb{N}$, we would need to find a set $f \subseteq \varnothing$. The only legal choice for $f$ would then be $\varnothing$.

Now we have to ask – does $\varnothing$ satisfy the remainder of the criteria for a function? Specifically, is the following statement true?

$$\text{For any } a \in \varnothing, \text{ there is a unique } b \in \mathbb{N} \text{ such that } (a, b) \in f$$

In this case, the statement turns out to be vacuously true – there are no elements $a \in \varnothing$, so *any* statement of the form "For any $a \in \varnothing$, ..." are automatically true. Consequently, under our definition of functions, this function $f$ is a legal function!

In fact, this function has a special name. Just as we have the *empty set*, a set of absolutely no elements, any function $f : \varnothing \to A$, for any set $A$, is called the *empty function*:

For any set $A$, there is one function $f : \varnothing \to A$. This function is called the ***empty function***.

# 6.2 Injections, Surjections, and Bijections

When we studied relations in the previous chapter, we explored certain classes of relations in more depth (the equivalence relations, partial orders, etc.) Although these relations didn't account for all types of relations, they tend to show up in many contexts and therefore were worth studying in the abstract. The same is true of functions; although functions come in all shapes and sizes, there are certain types of functions that appear more frequently than others. This section studies three of these types of functions: injections, surjections, and bijections.

## 6.2.1 Injections and Surjections

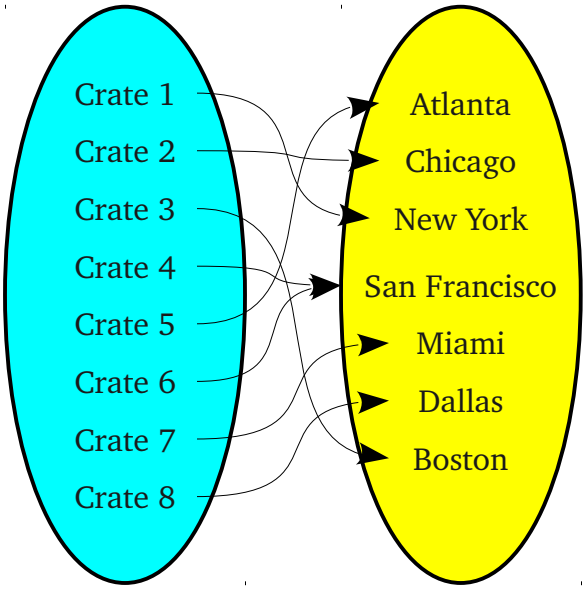
Let's consider an example of a problem we might want to model using functions. Suppose that you're in charge of the CDC and have a bunch of crates of vaccines. You want to distribute those crates across major US cities, and want to do so in a way that guarantees that every city has at least one crate of vaccines shipped to it.

Mathematically, you can think of this as finding a function that maps from the set of all crates of vaccines (let's call it $V$) to the set of major US cities (call it $C$). This function $f : V \to C$ should have the property that every city in the US has at least one crate shipped to it. In other words, we want to ensure that for every city $c \in C$, there is some vaccine crate $v \in V$ such that $f(v) = c$. This means that no city lacks vaccines, though some cities might get multiple vaccine shipments.

Not all functions $f : V \to C$ have this property. For example, a function that ships all the vaccines to San Francisco (i.e. $f(v)$ = San Francisco for all $v \in V$) doesn't ensure all cities are covered. The functions that do have this property are important and are referred to as *surjections*:

A function $f : A \to B$ is called a ***surjection*** iff for any $b \in B$, there is some $a \in A$ where $f(a) = b$. A function that is a surjection is often called a ***surjective function*** or an ***onto function***.

If we draw a function in the style of earlier in this chapter (with circles for the domain and codomain and arrows representing the output of the function), then a surjective function is one where every element of the codomain has at least one arrow entering it. For example, here's one possible surjective function from vaccine crates to cities:




Many common functions are surjective, such as the functions $f(x) = x$ or $f(x) = x^5$ over the real numbers. However, not all functions are surjective; for example, the function $f(x) = x^5$ isn't surjective over the natural numbers, since there is no natural number $x$ for which $f(x) = 2$, and the function $f(x) = x^2$ isn't surjective over the real numbers, since there is no real number $x$ for which $f(x) = -1$.

Now let's suppose that you're the mayor of a city that has received a shipment of vaccines. Unfortunately, you don't have enough vaccines to go around to everyone, and so you need to find a way to determine who gets the vaccines.

Let $I$ denote the set of all the vaccines. You can imagine a set $P$ of people who live in the city. In this case, you want to find a function $g : I \to P$ that distributes the vaccines equitably. One requirement for this function is that everyone should get at most one vaccine – there's no reason for anyone to get two vaccines when someone else didn't get any. As before, not all functions from $I$ to $P$ have this property, but the functions that do are important. In fact, they're so important that we call them *injections*.*

Intuitively, an injection is a function where no two distinct elements of the domain map to the same element in the codomain. This is formalized below:

> A function $f : A \to B$ is called an ***injection*** iff for any $x_1, x_2 \in A$, if $f(x_1) = f(x_2)$, then $x_1 = x_2$. Equivalently, for any $x_1, x_2 \in A$, if $x_1 \neq x_2$, then $f(x_1) \neq f(x_2)$. An injection is sometimes called an ***injective function*** or a ***one-to-one function***.

This definition is subtle, so let's take a minute to review what it says. We'll actually start with the second version of the definition first, since it's easier to understand, then transition to the first version of the definition, which tends to be easier to use in proofs.

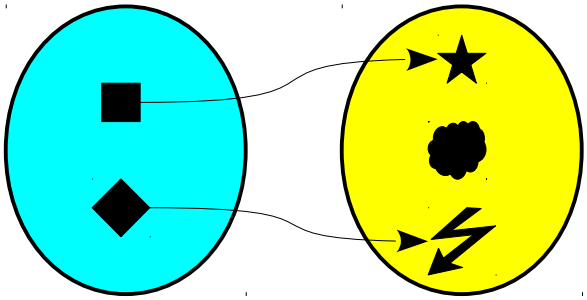
Intuitively, an injective function is one where if you start with two different inputs $x_1 \neq x_2$ to the function, you are guaranteed to get two different outputs $f(x_1) \neq f(x_2)$ after applying the function. This means that there can be at most one element of the domain that maps to any particular element of the codomain, since if you start with two different inputs you're guaranteed to get two distinct outputs.

---

* It's a pure coincidence that our example of divvying vaccines required a function called an injection.

The other, equivalent definition of an injective function is just the contrapositive of this statement: if you know that $f(x_1) = f(x_2)$, then you're guaranteed that $x_1 = x_2$. In other words, if you find two inputs $x_1$ and $x_2$ that produce the same outputs (that is, $f(x_1) = f(x_2)$), it means that you actually found the same input twice (that is, $x_1 = x_2$). This means that $x_1$ and $x_2$ just happen to be different names for the same thing.

Graphically, injective functions are functions where every element of the codomain has either zero or one incoming arrows. You can see this here:



As with surjections, many common functions are injections. For example, the function $f(x) = x$ is an injective function from any domain to itself, and $f(x) = x^2$ is injective if $f : \mathbb{N} \to \mathbb{N}$. However, the function $f(x) = \sin x$ isn't injective because $f(0) = f(2\pi) = 0$, nor is $f(x) = x^2$ over the real numbers injective, since $f(-1) = f(1) = 1$.

## 6.2.2 Images and Preimages ★

Although functions map individual elements of the domain to the codomain, we can generalize this to think about what a function will do to a *set* of elements. This gives rise to *images* and *preimages*, which are the subject of this section. (If it seems like we're abruptly leaving injections and surjections behind, don't worry; they'll come up again in a short while.)

Suppose you have a function $f : A \to B$. If you have a set $X \subseteq A$, you can think about the set of elements that you would get back if you were to apply $f$ to every element contained in $X$. In other words, you can think about the set $\{ f(x) \mid x \in X \}$. Since $f : A \to B$, we know $f(x) \in B$ for any $x \in X$, and so this set must be a subset of $B$. This set is important and is called the *image of X*:

> If $f : A \to B$ and $X \subseteq A$, the ***image of X under f*** is the set $\boldsymbol{f[X] = \{ f(x) \mid x \in X \}}$. (Note that those are square brackets around $X$ rather than parentheses.)

For example, consider the function $f(x) = x^2$ from $\mathbb{R}$ to $\mathbb{R}$. If $X = [-1, 3]$, then $f[X] = [0, 9]$, because as $x$ ranges from -1 to 0, $f(x)$ ranges from 1 to 0 and as $x$ ranges from 0 to 3, $f(x)$ ranges from 0 to 9. Accordingly, the set of all outputs produced by $f$ as applied to the elements of $X$ is the set of real numbers from 0 to 9, inclusive.

One particularly interesting image to look at is the image of the entire domain of the function. Let's suppose $f : A \to B$. What can we say about $f[A]$? This is the set $\{ f(a) \mid a \in A \}$, which consists of all possible outputs of the function. Although it might be temping to claim that this is the codomain of the function, it turns out that this isn't necessarily the case. For example, if we consider the function function $f : \mathbb{R} \to \mathbb{R}$ defined as $f(x) = \sin x$, then $f[\mathbb{R}] = [0, 1]$. More accurately, the set $f[A]$ consists of all possible values the function can ever take on. This is often referred to as the *range* of the function:

The ***range*** of a function $f : A \to B$ is the set $f[A]$. We denote the range of $f$ by ran $f$.

As mentioned earlier, the term “range” is a bit mathematically dicey. Some authors use the term “range” to refer to the entire codomain of the function, while others use “range” in the above sense to refer to the values in the codomain that can actually be produced by the function. For the purposes of this text, we'll use “range” to refer just to the values that can be produced, even if it's distinct from the codomain.

It might seem odd to make this distinction – what's the advantage of saying a function can produce more than it actually does? – but it dramatically simplifies many later terms and definitions. As you'll see later on, in some cases we will want to find specific types of functions from one set $A$ to another set $B$, even if the function doesn't actually produce all the elements of $B$ as outputs. By allowing a function to not use its entire codomain, we make it possible to say something like “consider any function $f : A \to B$ with properties $X$, $Y$, and $Z$” rather than “consider any function $f : A \to Q$, where $Q \subseteq B$, with properties $X$, $Y$, and $Z$.”

Given that there may be a distinction between a function's range and a function's codomain, it's useful to investigate under what circumstances the range and codomain are the same and are different. If you think about it for a while, you'll notice that the range and the codomain are equal precisely when every possible element of the codomain can be produced as the output of the function on some input. Functions with that property are precisely the surjective functions, which guarantee that every possible element of the codomain is “covered” by at least one element of the domain. We formalize this here:

***Theorem:*** If $f : A \to B$, then $f[A] = B$ iff $f$ is surjective.

***Proof:*** Let $f : A \to B$ be a function. Note that by definition, $f[A] = \{ f(a) \mid a \in A \}$. Equivalently, we have $f[A] = \{ b \in B \mid$ there is some $a \in A$ where $f(a) = b \}$. We will show that $f[A] = B$ iff $f$ is surjective by proving both directions of implication.

( $\Rightarrow$ ) First, we prove that if $f[A] = B$, then $f$ is surjective. Consider any $b \in B$. Since $f[A] = B$, this means that $b \in f[A]$. Therefore, there exists some $a \in A$ where $f(a) = b$. Since our choice of $b$ was arbitrary, this means that for any $b \in B$, there is an $a \in A$ where $f(a) = b$. Thus $f$ is surjective.

( $\Leftarrow$ ) Next, we prove that if $f$ is surjective, then $f[A] = B$. Since $f[A] = \{ f(a) \mid a \in A \}$ and we have $f : A \to B$, we know that every element of $f[A]$ is an element of $B$, so $f[A] \subseteq B$. Therefore, to show that $f[A] = B$, we will prove $B \subseteq f[A]$. To see this, consider any $b \in B$. Since $f$ is surjective, there is some $a \in A$ such that $f(a) = b$. By the second formulation of $f[A]$, this means $b \in f[A]$. Since our choice of $b$ was arbitrary, this means $B \subseteq f[A]$, and therefore $f[A] = B$. ■

When working with the image of a function $f : A \to B$, we begin with a set $X \subseteq A$ and end up with a set $f[X] \subseteq B$. We can also consider going the opposite direction. Let's suppose that have a set $Y \subseteq B$ of elements in the codomain. Depending on the function, some elements of $Y$ might not be mapped to by any elements of $A$, and some elements of $Y$ might be mapped to by multiple elements of $A$. We can consider taking the set of all elements of $A$ such that, when $f$ is applied to those elements, yields some element of $B$. This motivates *preimages*, which are essentially images applied in the opposite direction:

If $f : A \to B$ and $Y \subseteq B$, the ***preimage of Y under f*** is the set $\boldsymbol{f^{-1}[Y] = \{ x \in A \mid f(x) \in Y \}}$.

You can think of preimages in the following way. Given a function $f : A \to B$, start off by selecting some elements $Y \subseteq B$ out of the codomain. Now, look back at the domain $A$ and gather together all of the elements of the domain that map into set $Y$. That set is the preimage.

Some examples are probably in order. Let's start off with a simple function $f : \mathbb{R} \to \mathbb{R}$ where $f(x) = 2x$. Let's think about the range $Y = [1, 3]$. What is $f^{-1}[Y]$? This is the set of all the elements of $\mathbb{R}$ that, after $f$ is applied, yield values in the range $[1, 3]$. Since $f(x) = 2x$, this corresponds to the range $[0.5, 1.5]$. Therefore, $f^{-1}[Y] = [0.5, 1.5]$.

Now, let's take a trickier example. Let's let $f : \mathbb{R} \to \mathbb{R}$ be defined as $f(x) = x^2$. Now, let's let $Y = [4, 9]$. In this case, what's $f^{-1}[Y]$? Well, this should be the set of all real numbers for which $x^2$ is in the range $[4, 9]$. This is the set $[-3, -2] \cup [2, 3]$, since those are precisely the numbers whose squares are in the range $[4, 9]$.

Let's do yet another example. As before, let $f : \mathbb{R} \to \mathbb{R}$, but this time define $f(x) = x^2 + 2$. Let $Y = [0, 1]$. What is $f^{-1}[Y]$ now? An important observation is that there are no real numbers $x$ such that $f(x) < 2$, since for any real number $x$ we have $f(x) = x^2 + 2 \geq 2$. Is this a problem? Not at all! The preimage of $Y$ is the set of all $x \in \mathbb{R}$ where $f(x) \in Y$. Since there aren't any $x \in R$ where $f(x) \in Y$, the preimage of $Y$ ends up being the empty set. In other words, $f^{-1}[Y] = \varnothing$.

Now, let's do one final example. Let $f : \mathbb{R} \to \mathbb{R}$ be $f(x) = x^2 + 2$ as before. We just saw that $f^{-1}[Y] = \varnothing$ when $Y = [0, 1]$. Let's now change $Y$ so that $Y = [0, 2]$. Now, what is $f^{-1}[Y]$? Well, for every element of $Y$

other than 2, we know that there are no elements of $\mathbb{R}$ that map to that element. But if we consider the number 2, we see that there is exactly one number $x \in \mathbb{R}$ for which $f(x) = 2$; namely, $x = 0$. Therefore, if we look at all the elements of $\mathbb{R}$ and filter them down to just those for which $f(x) \in Y$, we'll end up with the singleton set $\{0\}$. Therefore, $f^{-1}[Y] = \{0\}$.

Just as images are connected to surjections, preimages are connected to injections. Remember that an injective function is one that, intuitively, never maps two or more elements from the domain to any single element in the codomain. We can try to phrase this in the language of preimages. Let $f : A \to B$ be an injective function. This means that every $b \in B$ has either zero or one element mapping to it. Therefore, the set $f^{-1}[\{b\}]$ (that is, the set of all elements in $A$ that map to $b$) should either contain zero elements or one element. In other words, we'd expect that $|f^{-1}[\{b\}]| \leq 1$. We'll leave this proof as an exercise to the reader. ☺

### 6.2.3 Bijections

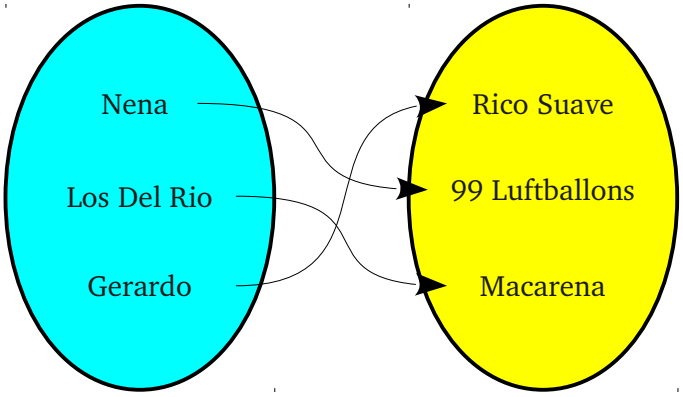
Before concluding this section on injections and surjections, we should talk about one final class of functions: the *bijections*.

Going back to our intuitions about injections and surjections, we said that an injection is a function where every element of the codomain has *at most* one element of the domain mapping to it and that a surjection is a function where every element of the codomain has *at least* one element of the domain mapping to it. What happens if we consider functions where every element of the codomain has *exactly* one element of the domain mapping to it? These functions would be simultaneously injections and surjections, and they're so important that we give them their own name: *bijections*.

> A function is called a ***bijection*** (or ***bijective function***) iff it is injective and surjective.

The above definition just tells us what a bijection *is*. What does a bijection *mean*? Well, every function associates every element of the domain with some element from the codomain. Because a bijection is surjective, every element of the codomain has at least one element in the domain associated with it, and because a bijection is injective every element of the codomain has at most one element in the domain associated with it. Therefore, for every element of the codomain, there is a *unique* element of the domain mapping to it. Consequently, we can speak of "the" element of the domain that maps to a specific element of the codomain.

Graphically, bijective functions are functions that pair of elements of the domain and codomain in a way that covers all elements of both sets. For example, the following function is a bijection:



Many simple functions are bijections. For example, the function $f : \mathbb{R} \to \mathbb{R}$ defined as $f(x) = x^3$ is a bijection, since it's surjective (every number $x$ is mapped to by its cube root) and injective (if $x^3 = y^3$ and $x$ and $y$ are real numbers, then $x = y$). For any set $S$, the identity function $f : S \to S$ defined as $f(x) = x$ is also a bijection. However, the function $f(x) = x^2$ is not a bijection if $f : \mathbb{R} \to \mathbb{R}$ or $f : \mathbb{N} \to \mathbb{N}$ (in neither case is it surjective), though $f(x) = x^2$ is a bijection if $f : \{0, 1\} \to \{0, 1\}$. (Do you see why?)

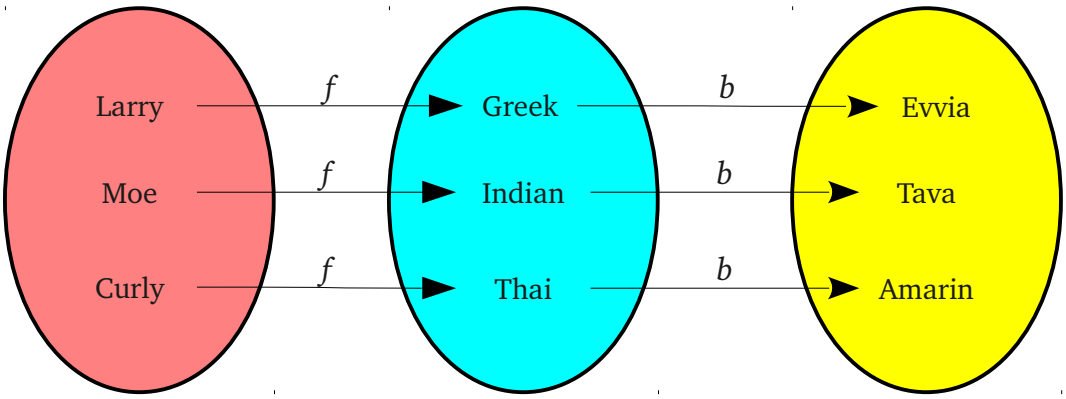
When we introduced cardinality in Chapter 1, we talked about "one-to-one correspondences" between sets. Bijections formalize this concept, since they provide a way of pairing up the elements of the domain and codomain so that every element of each is paired with a unique element of the other. We'll explore this in more detail later in this chapter when we revisit cardinality and provide formal definitions for what it means for two sets to have the same cardinality as one another.

# 6.3 Transformations on Functions

Once we have a set of functions lying around, what can we do with them? How can we combine them together? How can we manipulate them? This section explores those questions.

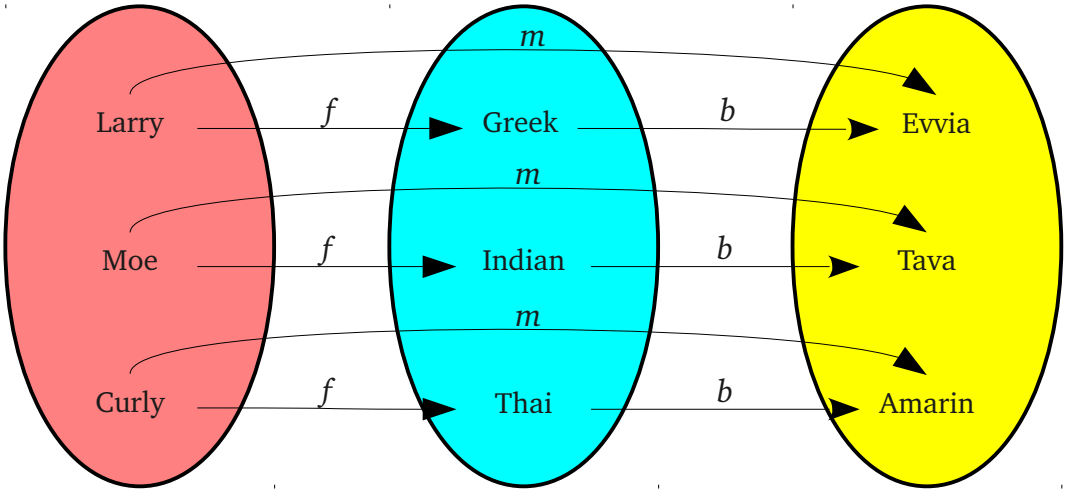
## 6.3.1 Composing Functions

Let's suppose we have three sets $P$, $F$, and $R$. $P$ is a set of people, $F$ is a set of different types of food, and $R$ is a set of restaurants. Now let's imagine that we have two functions: $f : P \to F$ that for each person says which food they like the best, and $b : F \to R$, which maps from foods to the best restaurants that serve that type of food. For example, these functions might look like this:




Now, let's suppose that you are interested in telling each person what restaurant they should go to in order to get their favorite food prepared as well as possible. Given a person $p$, you can determine their favorite food by computing $f(p)$, and from there can apply the function $b$ to determine which restaurant prepares that food best. Therefore, the net result of this operation is $b(f(p))$, the best place to get the food that person $p$ likes most.

In order to get this information, we had to apply two different functions to $p$: first we applied $f$, then we applied $b$. You can imagine defining some new function $m : P \to R$ that does precisely this. We could define this function as $m(p) = b(f(p))$. In effect, this function glues together the functions $f$ and $b$ to produce a new function that skips the intermediary step. Graphically, this function $m$ looks like this:




It's quite common to join functions together in this fashion, and in fact there's a special name for the function produced by sequencing functions like this. We call it the *composition* of the functions:

> Let $f : A \to B$ and $g : B \to C$. The ***composition of f and g***, which we denote $g \circ f$ is a function $g \circ f : A \to C$ such that $(g \circ f)(x) = g(f(x))$.

Let's work through this definition to see what it says. We start off with two functions $f$ and $g$ where $f$'s codomain is $g$'s domain. This means that it's possible to apply $g$ to the outputs of the function $f$. The composition of the functions is the new function $g \circ f$ that's defined as follows: evaluating $g \circ f$ on an input $x$ produces the same result as applying $g$ to $f(x)$. Since we're evaluating $f$ first, the domain of the new function $g \circ f$ is the domain of $f$ (namely, $A$), and since the overall result is the result of evaluating $g$, the codomain of the new function is the codomain of $g$ (namely, $B$). In the above example, our function $m$ was the composition $b \circ f$.

Notationally, the value of $g \circ f$ applied to a value $x$ is denoted $(g \circ f)(x)$. There are two sets of parentheses here. The second set of parentheses (the ones around $x$) are the normal parentheses we place around the argument to a function call. The first set of parentheses, which surround $g \circ f$, are there to clarify a slight notational ambiguity. If we wrote out $g \circ f(x)$, we could potentially parse it incorrectly as "compute the value of $f(x)$, then compose $g$ with that value," rather than "compose $g$ with $f$, then apply that resulting function to $x$."

Before moving on, let's take a minute to clarify a few important properties of compositions. First, note that the order of the functions $g \circ f$ in the composition is significant. $g \circ f$ means "apply $f$ first, then apply $g$," while $f \circ g$ means "apply $g$ first, then apply $f$." This might seem unusual, since when read from the left to the right you might get the impression that $g \circ f$ would apply $g$ first and $f$ second. A good way to remember which function comes first is to think about where the parentheses would be placed if you tried to evaluate $g(f(x))$; here, $f$ is applied first, then $g$.

In general, for any two functions $f$ and $g$, there's no guarantee that either $g \circ f$ or $f \circ g$ are defined. For example, let's take the function $b$ from the above example (mapping from types of food to the best restaurant that serves it) and let $h(x) = x^3$ be a function from real numbers to real numbers. We can't compose these functions with one another, since their domains and codomains are completely incomparable.

In some cases, though, you can find functions that can be composed in either order. For example, let's let $f$, $g : \mathbb{N} \to \mathbb{N}$ be defined as $f(n) = 2n$ and $g(n) = n + 1$. Then $g \circ f$ and $f \circ g$ are both well-defined functions.

Note, however, that these functions are not equal to one another. $(g \circ f)(x) = g(f(x)) = 2n + 1$, while $(f \circ g)(x) = 2n + 2$. Consequently, when composing functions, *order does matter!*

#### 6.3.1.1 Composing Injections, Surjections, and Bijections

Let's suppose $f : A \to B$ and $g : B \to C$ are injective. Is $g \circ f$ necessarily injective? If $g$ and $f$ are surjective, is $g \circ f$ surjective? What if $g$ and $f$ are bijections... does that mean $g \circ f$ is a bijection?

It turns out that the answer to all three of these questions is "yes." Composing two injective functions always results in an injective function, composition two surjective functions always results in a surjection, and composing two bijective functions always results in a bijection. The proofs of these facts aren't particularly difficult (most of it involves manipulating definitions), but they're great examples of how to write proofs involving these definitions. Let's take a look at them and see how they work.

First, let's prove that the composition of two injective functions must be injective. Intuitively, we can see this as follows. Let's let our injections be $f : A \to B$ and $g : B \to C$. Since $f$ is injective, if we feed in any two distinct inputs $x$ and $y$, we know that $f(x)$ and $f(y)$ will be different. Similarly, since $g$ is injective, we know that $g(f(x))$ must be different from $g(f(y))$ because $f(x) \neq f(y)$. Formalizing this as a proof is relatively straightforward.

If you'll recall from earlier, we saw two equivalent definitions of injectivity. The first states that if we have $h : X \to Y$, then $h$ is injective iff for any $x$, $y \in X$ where $x \neq y$, that $h(x) \neq h(y)$. The second states that if $h : X \to Y$, then $h$ is injective iff whenever $f(x) = f(y)$, we have $x = y$. Below are two different proofs that the composition of injections is an injection, each using a different one of the definitions.

***Theorem:*** Let $f : A \to B$ and $g : B \to C$ be injections. Then $g \circ f : A \to C$ is an injection.

***Proof 1:*** Consider any $x$, $y \in A$ where $x \neq y$. Since $x \neq y$ and $f$ is an injection, we have $f(x) \neq f(y)$. Since $f(x) \neq f(y)$ and $g$ is an injection, we have $g(f(x)) \neq g(f(y))$. Since $(g \circ f)(x) = g(f(x))$ and $(g \circ f)(y) = g(f(y))$, this means $(g \circ f)(x) \neq (g \circ f)(y)$. Since our choice of $x$ and $y$ were arbitrary, this means that for any $x$, $y \in A$ where $x \neq y$ that $(g \circ f)(x) \neq (g \circ f)(y)$, so $g \circ f$ is injective. ■

***Proof 2:*** Consider any $x$, $y$ where $(g \circ f)(x) = (g \circ f)(y)$. Since $(g \circ f)(z) = g(f(z))$ for all $z \in A$, this means that $g(f(x)) = g(f(y))$. Since $g$ is injective and $g(f(x)) = g(f(y))$, we have $f(x) = f(y)$. Since $f$ is injective and $f(x) = f(y)$, we have $x = y$. Thus if $(g \circ f)(x) = (g \circ f)(y)$ we have $x = y$, so $g \circ f$ is injective. ■

Both of these proofs are valid lines of reasoning and show off different routes for proving that the composition of injective functions are injective. Hopefully, this will help you get a better sense for how to write proofs of injectivity.

What about compositions of surjections? That too will result in a surjection. To think about why this is, let's suppose $f : A \to B$ and $g : B \to C$ are surjections and consider $g \circ f$. For this function to be a surjection, we need to show that for any $c \in C$, there is some $a \in A$ such that $(g \circ f)(a) = c$. So pick any $c$ you'd like in

$C$. Since $g$ is surjective, there has to be some $b \in B$ such that $g(b) = c$. In turn, since $f$ is surjective, there has to be some $a \in A$ such that $f(a) = b$. Then $g(f(a)) = g(b) = c$, and we're done.

The proof of this is pretty much the above reasoning.

> ***Theorem:*** If $f : A \to B$ and $g : B \to C$ are surjections, then $g \circ f : A \to C$ is a surjection.

> ***Proof:*** We will show that for any $c \in C$, there is an $a \in A$ such that $(g \circ f)(a) = c$. So take any $c \in C$. Since $g$ is surjective, there exists some $b \in B$ such that $g(b) = c$. Similarly, since $f$ is surjective, there exists some $a \in A$ such that $f(a) = b$. Then $(g \circ f)(a) = g(f(a)) = g(b) = c$, as required. ■

An interesting detail about this proof is that it is *nonconstructive*. We don't show how we actually *find* the choices of $b$ and $a$ that come up in the proof. Instead, we rely on the fact that since $g$ and $f$ are surjections, such values must exist. Commonly, when writing proofs about surjections, you will end up using the fact that some object exists, even if you don't explicitly say how to find it.

To wrap up our discussion of function composition, let's talk about composition of bijections. Is the composition of two bijections necessarily a bijection? The answer is yes, and the proof is remarkably simple: since bijections are injective and surjective, the compositions of two bijections must be injective and surjective. Therefore, the composition of two bijections must be a bijection. Formally, we have this:

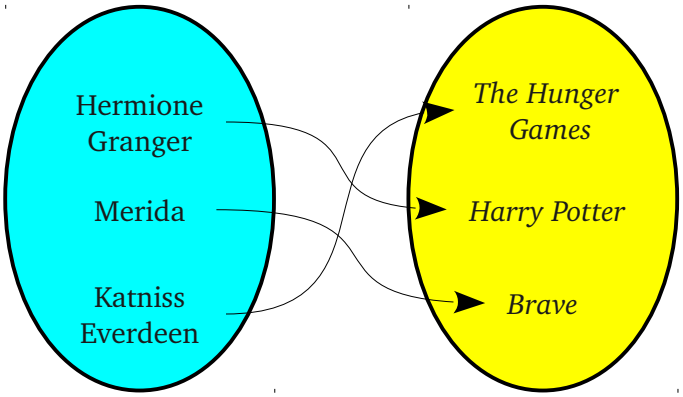
> ***Theorem:*** If $f : A \to B$ and $g : B \to C$ are bijections, then $g \circ f : A \to C$ is a bijection.

> ***Proof:*** Let $f : A \to B$ and $g : B \to C$ be bijections. Therefore, $f$ and $g$ are injective and surjective, so by our previous theorems $g \circ f$ must be injective and surjective. Therefore, $g \circ f$ is a bijection. ■

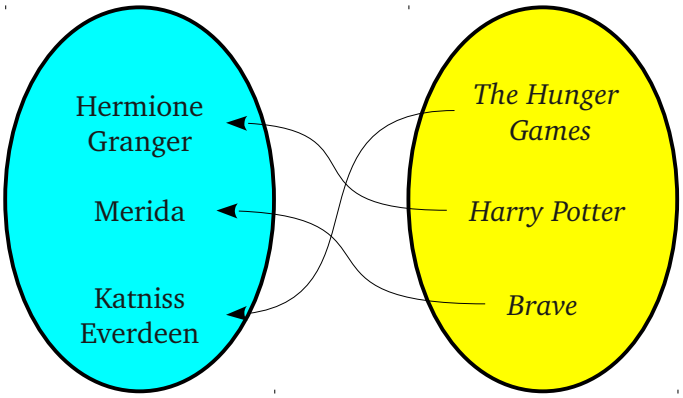
This is probably one of the shortest proofs we've seen so far, but that doesn't mean it's not important. We'll use this fact as a building block when we talk about cardinalities.

## 6.3.2 Inverse Functions

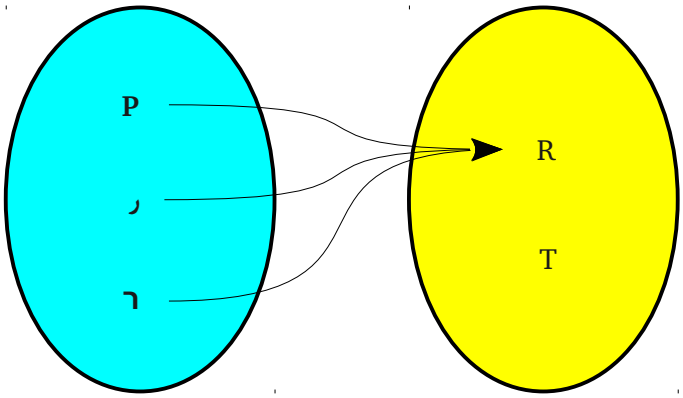
Suppose that you start off with a function $f : A \to B$ which maps from elements of $A$ to elements of $B$. Can we somehow "turn the function around" to end up with some new function $g : B \to A$ that just runs $f$ backwards? For example, consider the following function:



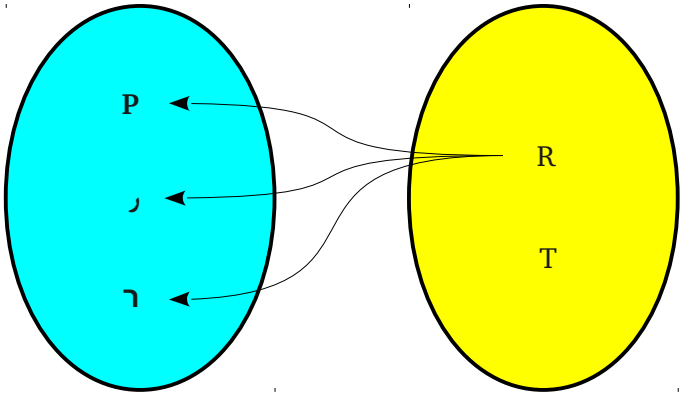

We could try turning this function around by reversing all of the arrows, which gives rise to a new function mapping from the codomain back to the domain:




Can we do this for all functions? Unfortunately, we can't. Functions are inherently asymmetric: every element of the domain is associated with just one element of the codomain, while elements of the codomain might be mapped to by any number of elements from the domain. This is why the graphical representation of a function might have multiple (or no) arrows entering certain elements of the codomain. For example, consider the following function:




If we try to flip around all of the arrows to reverse the function, we end up with the following:




Notice that this isn't a function, since R has multiple outgoing arrows and T has no outgoing arrows.

As you just saw, under certain circumstances we can reverse a function and under certain circumstances we cannot. This naturally leads us to ask under what circumstances we can do this. But we're getting a bit ahead of ourselves. Right now, we're talking about "turning a function around" or "reversing arrows," which are great intuitions but aren't very precise. Let's take a minute to mathematically formalize this intuition so that we can start writing rigorous proofs about what exactly this operation is.

Let's begin by investigating what happens when we reverse all of the arrows in the drawing of a function. Assuming that we end up with a function at all (which isn't guaranteed, as we just saw), what properties will that function have? Think of it this way: if we start off with some function $f : A \to B$ and find that $f(x) = y$, then in our drawing we would have an arrow from $x \in A$ to $y \in B$. If we reverse the arrows, we'll now have an arrow from $y \in B$ to $x \in A$. If we call this new function $g : B \to A$, then we'll have $g(y) = x$. Consequently, we can think about defining what it means to "turn the function around" using the following intuition: start with $f : A \to B$. Then, define $g : B \to A$ so that $g(b) = a$ iff $f(a) = b$. Again, this isn't always going to work (this might not legally define a function), but in the cases where it does it gives us a nice, formal definition.

The operation we've just described is called *inverting* the function $f$. We've denoted the resulting function $g$, but more generally the mathematical notation used here is $f^{-1}$. This is formalized below:

> Let $f : A \to B$ be a function. If it exists, ***inverse of f*** is the function $\boldsymbol{f^{-1}} : B \to A$ defined as follows: $f^{-1}(b) = a$ iff $f(a) = b$. If $f^{-1}$ exists, we say that $f$ is ***invertible*** or an ***invertible function***.

Before moving on, let's play around with some concrete examples of inverse functions so we can build up an intuition for what inverses mean. For example, let's take the function $f(x) = x + 2$ from $\mathbb{R}$ to $\mathbb{R}$. What would $f^{-1}(x)$ be here? Well, by definition, we should have that $f^{-1}(x) = y$ iff $f(y) = x$. Using our definition of $f$, this means that $f^{-1}(x) = y$ iff $y + 2 = x$. This in turn means $f^{-1}(x) = y$ iff $y = x - 2$, so $f^{-1}(x) = x - 2$.

Next, let's think about $f(x) = x^3$ from $\mathbb{R}$ to $\mathbb{R}$. What's $f^{-1}(x)$ in this case? Since $f^{-1}(x) = y$ iff $f(y) = x$, we have that $f^{-1}(x) = y$ iff $y^3 = x$. Since $x, y \in \mathbb{R}$, the only way that $y^3 = x$ is if $y = \sqrt[3]{x}$. Therefore, we have that $f^{-1}(x) = \sqrt[3]{x}$.

How about the function $f(x) = e^x$ from $\mathbb{R}$ to $\mathbb{R}$? Now what's $f^{-1}(x)$? Well, we should have that $f^{-1}(x) = y$ iff $f(y) = x$. In other words, $f^{-1}(x) = y$ iff $e^y = x$. And here we have a problem. What happens if $x \leq 0$? In this case, there is no (real valued) $y$ such that $e^y = x$. Consequently, saying "$f^{-1}(x) = y$ iff $e^y = x$" is problematic

because there is no $y$ with this property. Therefore, in this case, $f^{-1}(x)$ does not exist and $e^x$ from $\mathbb{R}$ to $\mathbb{R}$ is not invertible.

"But wait!," you might say. "I thought that $e^x$ is invertible, and its inverse is ln $x$!" In a sense you are right. Let's denote by $\mathbb{R}^+$ the set of all positive real numbers (using set-builder notation, we can define $\mathbb{R}^+ = \{ x \in \mathbb{R} \mid x > 0 \}$). If we then define $f : \mathbb{R} \to \mathbb{R}^+$ as $f(x) = e^x$ and specifically restrict the codomain of $f$ to be $\mathbb{R}^+$, then $f(x)$ becomes invertible. Specifically, $f^{-1} : \mathbb{R}^+ \to \mathbb{R}$ is the function defined such that $f^{-1}(x) = y$ iff $f(y) = x$, where $x \in \mathbb{R}^+$ and $y \in \mathbb{R}$. With a little algebra, we can see that $f^{-1}(x) = \ln x$, which matches the intuition.

Let's look at one final example. Let's take $f(x) = x^2$, where $f : \mathbb{R} \to \mathbb{R}^+$. (Here, the codomain is appropriate because $x^2 > 0$ for all real, nonzero $x$). What's $f^{-1}(x)$ here? We should have $f^{-1}(x) = y$ iff $f(y) = x$. In turn, $f(y) = x$ iff $y^2 = x$. Now we hit another snag: for any positive real $x$, there are two different values of $y$ such that $y^2 = x$. For example, if $x = 4$, then $2^2 = (-2)^2 = x$, so there are two possible values of $y$ for which $f(y) = x$. Therefore, there would have to be two possible values for $f^{-1}(x)$, which is impossible. We have to conclude that $f^{-1}(y)$ doesn't exist.

Let's take a minute to see why the two functions we couldn't invert weren't invertible. In the case of $e^x$ from $\mathbb{R}$ to $\mathbb{R}$, the problem was that there are elements of the codomain (namely, $\mathbb{R}$) that are never mapped to by $e^x$. Accordingly, when we tried to invert the function, we found that $f^{-1}(x)$ would have to be undefined on certain inputs, which is impossible. If we think about this more generally, we can realize that *any* function that leaves any element of the codomain uncovered can't be invertible for the same reason: when we try to define $f^{-1}(x)$ for some uncovered element $x$ of the codomain, we'll discover that it doesn't map back to anything. Therefore, any function that is invertible has to be surjective.

Let's now look at $x^2$ from $\mathbb{R}$ to $\mathbb{R}^+$. This function wasn't invertible because there are multiple elements of the domain that map to the same element of the codomain, so when we try to define $f^{-1}(x)$ on those elements, we will run into a problem because it would need to map to two or more different elements, which isn't possible. More generally, any function that is not injective can't be invertible because when trying to define $f^{-1}(x)$ on any element of the codomain that's mapped to two or more times, we will encounter this same problem. Therefore, any function that is invertible has to be injective.

If we look at the two preceding paragraphs, we see something interesting: any invertible function has to be both injective and surjective, meaning that any invertible function has to be a bijection. This is not at all obvious from the definition! In fact, this is so significant that we should prove it:

***Theorem:*** If $f$ is invertible, then $f$ is a bijection.

***Proof:*** Let $f : A \to B$ be invertible. We will prove $f$ is a bijection by contradiction; suppose $f$ is not a bijection. We consider two cases:

*Case 1: $f$* is not injective. Then there exists some $x_0, x_1 \in A$ such that $x_0 \neq x_1$ but $f(x_0) = f(x_1)$. Let $y = f(x_0)$ and consider $f^{-1}(y)$. By definition, $f^{-1}(y) = x$ iff $f(x) = y$. Thus $f^{-1}(y) = x_0$ and $f^{-1}(y) = x_1$. But this means $x_0 = x_1$, which is impossible.

*Case 2: $f$* is not surjective. Then there exists some $y \in B$ such that $f(x) \neq y$ for all $x \in A$. So consider $f^{-1}(y)$. By definition of $f^{-1}$, we have $f^{-1}(y) = x$ iff $f(x) = y$. We have $f^{-1}(y) \in A$, so there exists some $z \in A$ such that $f^{-1}(y) = z$. But this means that $f(z) = y$, contradicting the fact that $f(x) \neq y$ for any $x \in A$.

In both cases, we reach a contradiction and so our assumption must be wrong. Thus if $f$ is invertible, $f$ must be a bijection. ■

We have just shown that if a function is invertible, it must be a bijection. Is the converse true? That is, is it also the case that if a function is a bijection, it is invertible? It turns out that the answer is "yes." This is surprising, since the definitions of bijective functions and invertible functions look nothing like one another, yet they're exactly the same?

How exactly would we prove this? If $f : A \to B$ is a bijection, then $f^{-1} : B \to A$, if it exists at all, is defined as $f^{-1}(b) = a$ iff $f(a) = b$. Given this rule describing the behavior of $f^{-1}$, how would we show it exists? To do this, we'll need to call down to a lower-level definition of what a function is. Earlier in the chapter, we saw several equivalent definitions of a function. One slight variation on these definitions is the following: $f : A \to B$ iff

- For any $x \in A$, there is some $y \in B$ for which $f(x) = y$. (Every element of the domain maps to an element of the codomain.)
- If $f(x) = y_0$ and $f(x) = y_1$, then $y_0 = y_1$. (Every element of the domain maps to *exactly one* element of the codomain.)

To show that if $f$ is a bijection then $f$ is also invertible, we will show that the definition of $f^{-1}$ given above satisfies these two requirements.

***Theorem:*** If $f$ is a function, $f$ is invertible iff $f$ is a bijection.

***Proof:*** We proved in the previous theorem that if $f$ is invertible, then $f$ is a bijection. Therefore, all we need to do is show that if $f$ is a bijection, $f$ is invertible.

Consider any bijection $f : A \to B$ and let $f^{-1}(b)$ be defined as $f^{-1}(b) = a$ iff $f(a) = b$. We will prove that $f^{-1} : B \to A$.

First, we prove that for any $b \in B$, there exists some $a \in A$ for which $f^{-1}(b) = a$. Since $f$ is surjective, for any $b \in B$, there exists some $a \in A$ such that $f(a) = b$. Therefore, $f^{-1}(b) = a$, so for any $b \in B$ there is at least one $a \in A$ such that $f^{-1}(b) = a$.

Next, we prove that for any $b \in B$, if $f^{-1}(b) = a_0$ and $f^{-1}(b) = a_1$, then $a_0 = a_1$. By definition, since $f^{-1}(b) = a_0$ and $f^{-1}(b) = a_1$. This means $f(a_0) = f(a_1)$. Since $f$ is injective, this in turn means $a_0 = a_1$, as required.

Thus $f^{-1}$ is a function, so $f$ is invertible, as required. ■

## 6.3.3 Compositions and Inverses

We've just built up two concepts: function composition and inverse functions. What happens if we start looking at how the two interact?

We can make an interesting observation about how inverse functions work. Let's look back at some of the simple functions whose inverses we found when exploring the definition of inverse functions.

- When $f(x) = x + 2$, $f^{-1}(x) = x - 2$. Here, $f : \mathbb{R} \to \mathbb{R}$ and $f^{-1} : \mathbb{R} \to \mathbb{R}$.
- When $g(x) = x^3$, $g^{-1}(x) = \sqrt[3]{x}$. Here, $g : \mathbb{R} \to \mathbb{R}$ and $g^{-1} : \mathbb{R} \to \mathbb{R}$.
- When $h(x) = e^x$, $h^{-1}(x) = \ln x$. Here, $h : \mathbb{R} \to \mathbb{R}^+$ and $h^{-1} : \mathbb{R}^+ \to \mathbb{R}$.

Look at the domains and codomains of these functions and their inverses. If you'll notice, for $f$ and $g$, it's possible to define $f \circ f^{-1}$, $f^{-1} \circ f$, $g \circ g^{-1}$, and $g^{-1} \circ g$ because the domain and codomain are the same. For $h$, even though the domain and codomain are different, we still can define $h^{-1} \circ h$ and $h \circ h^{-1}$. Since these functions are defined, we can consider what happens if we see exactly what they are. The results are a bit interesting:

- When $f(x) = x + 2$, $f^{-1}(x) = x - 2$. Then
  - $(f^{-1} \circ f)(x) = (x + 2) - 2 = x$.
  - $(f \circ f^{-1})(x) = (x - 2) + 2 = x$.
- When $g(x) = x^3$, $g^{-1}(x) = \sqrt[3]{x}$. Then
  - $(g^{-1} \circ g)(x) = \sqrt[3]{x^3} = x$.
  - $(g \circ g^{-1})(x) = \left(\sqrt[3]{x}\right)^3 = x$.

- When $h(x) = e^x$, $h^{-1}(x) = \ln x$. Then
  - $(h^{-1} \circ h)(x) = \ln e^x = x$.
  - $(h \circ h^{-1})(x) = e^{\ln x} = x$.

Now *that's* interesting! In each of these cases, when we compose a function and its inverse, we end up with a function that takes in $x$ and evaluates to $x$ as well. This function is significant and we call it the *identity function*:

> For any set $A$, the ***identity function over A*** is the function $\text{id}_A : A \to A$ defined as $\text{id}_A(x) = x$.

Now, let's suppose that we have an invertible function $f : A \to B$. From what we saw above, if we take the inverse function $f^{-1} : B \to A$ and then compose $f^{-1} \circ f : A \to A$, we would expect to find that $f^{-1} \circ f = \text{id}_A$. (We haven't formally seen yet what it means for two functions to be equal. This just means that the two functions have the same domain, codomain, and always produce the same outputs as one another.) Similarly, if we compose $f \circ f^{-1} : B \to B$, we would expect to find $f \circ f^{-1} = \text{id}_B$. Notice that we get back an identity function in both cases, but the domain and codomain of that identity function are different; the function $f^{-1} \circ f$ maps from elements of $A$ to elements of $A$, while the function $f \circ f^{-1}$ maps from elements of $B$ to elements of $B$.

At this point, we're just conjecturing that these statements are true. We have some examples to back it up, but that doesn't mean this will always be the case. To confirm that these statements are true, let's try proving it.

Intuitively speaking, we can think about the function $f^{-1} \circ f$ as follows. Remember that by our definition of inverse functions, we have $f^{-1}(y) = x$ iff $f(x) = y$. This means that if you evaluate $f(x)$ and get back some value $y$, then plugging $y$ into $f^{-1}$ will produce $x$ as its output. Consequently, if we compose $f^{-1}$ and $f$ to get $f^{-1} \circ f$ and then evaluate $(f^{-1} \circ f)(x)$, we should get back $x$ because $f^{-1}$ will "undo" the application of $f$. This is formalized below:

> ***Lemma:*** Let $f : A \to B$ be an invertible function. Then $f^{-1} \circ f = \text{id}_A$.

> ***Proof:*** Let $f : A \to B$ an invertible function. Then $f^{-1} : B \to A$ exists and $f^{-1} \circ f : A \to A$. To prove that $f^{-1} \circ f = \text{id}_A$, we need to show for all $x \in A$ that $(f^{-1} \circ f)(x) = \text{id}_A(x)$. Since by definition we have $\text{id}_A(x) = x$ for all $x \in A$, this means we need to show that for all $x \in A$, we have $(f^{-1} \circ f)(x) = x$.
>
> By definition of function composition, we have $(f^{-1} \circ f)(x) = f^{-1}(f(x))$. Consider any $x \in A$. Now, let $y = f(x)$, which means that $f^{-1}(f(x)) = f^{-1}(y)$. By the definition of $f^{-1}$, we have $f^{-1}(b) = a$ iff $f(a) = b$. Therefore, $f^{-1}(y) = x$. Putting everything together, we have $(f^{-1} \circ f)(x) = f^{-1}(f(x)) = f^{-1}(y) = x$. Since our choice of $x$ was arbitrary, this means $(f^{-1} \circ f)(x) = x$ for all $x \in A$, so $f^{-1} \circ f = \text{id}_A$. ■

We've just shown that if we take $f^{-1} \circ f$ for an invertible function $f$, we get back $\text{id}_A$. We also claimed that if we take $f \circ f^{-1}$, we should get back $\text{id}_B$. How exactly might we prove this?

To show $f \circ f^{-1} = \text{id}_B$, we need to show that for any $x \in B$, that $f(f^{-1}(x)) = x$. When proving $f^{-1} \circ f = \text{id}_A$, we were able to simplify $f^{-1}(f(x))$ by letting $y = f(x)$, then using the fact that by definition, $f^{-1}(y) = x$. Could we use the same trick here? Let's try it! This time, to simplify $f(f^{-1}(x))$, let's set $y = f^{-1}(x)$. What does that tell

us about $x$ and $y$? Well, we know that $f^{-1}(x) = y$ iff $f(y) = x$. Therefore, $f(f^{-1}(x)) = f(y) = x$, and we're done! As above, let's quickly formalize this argument before moving on.

***Lemma:*** Let $f : A \to B$ be an invertible function. Then $f \circ f^{-1} = \text{id}_B$.

***Proof:*** Let $f : A \to B$ an invertible function. Then $f^{-1} : B \to A$ exists and $f \circ f^{-1} : B \to B$. To prove that $f \circ f^{-1} = \text{id}_B$, we need to show for all $x \in B$ that $(f \circ f^{-1})(x) = \text{id}_B(x)$. Since by definition we have $\text{id}_B(x) = x$ for all $x \in B$, this means we need to show that for all $x \in B$, we have $(f \circ f^{-1})(x) = x$.

By definition of function composition, we have $(f \circ f^{-1})(x) = f(f^{-1}(x))$. Consider any $x \in B$. Now, let $y = f^{-1}(x)$, which means that $f(f^{-1}(x)) = f(y)$. By definition of $f^{-1}$, we have $f^{-1}(b) = a$ iff $f(a) = b$. Therefore, $f(y) = x$. Putting everything together, we have $(f \circ f^{-1})(x) = f(f^{-1}(x)) = f(y) = x$. Since our choice of $x$ was arbitrary, this means $(f \circ f^{-1})(x) = x$ for all $x \in B$, so $f \circ f^{-1} = \text{id}_B$. ■

Combining these two lemmas together gives us the following theorem:

***Theorem:*** If $f : A \to B$ is invertible, then $f^{-1} \circ f = \text{id}_A$ and $f \circ f^{-1} = \text{id}_B$.

Nifty! This gives us a new perspective on inverse functions: if you compose a function on the left or the right with its inverse, you end up getting back an identity function on the appropriate set.

Earlier, when discussing the connection between inverses and bijections, we first saw that all invertible functions were bijections, then proved a stronger result that said that all bijections were also inverse functions. Right now, we've shown that *if* $f : A \to B$ is invertible, *then* $f^{-1} \circ f = \text{id}_A$ and $f \circ f^{-1} = \text{id}_B$.

Now, suppose that you have a function $f : A \to B$ and a function $g : B \to A$ such that both $g \circ f = \text{id}_A$ and $f \circ g = \text{id}_B$. Does this necessarily mean that $g = f^{-1}$? In other words, if you can compose $f$ and $g$ together in two different ways to get back two identity functions, are $f$ and $g$ guaranteed to be inverses of one another?

It's not immediately clear whether this would be true or not. On the one hand, if we know absolutely nothing about $f$ and $g$ than the fact that they compose in this fashion, why on earth would this guarantee that the two functions have to be inverses of one another? On the other hand, if the functions "undo" one another, then wouldn't that mean that they're inverses?

A good tactic when you're unsure of whether a result is true is to try to spell out what results you'd need to prove in order to prove it, then playing around with those results until either you (a) find a reason why it should be true, or (b) get stuck. If you find (a), it probably means you're on the path toward a proof and can try to show the result is true. If you find (b), you might want to try searching for a counterexample.

In this case, we have $f : A \to B$ and $g : B \to A$. We know $g \circ f = \text{id}_A$ and $f \circ g = \text{id}_B$. If we want to prove that $g = f^{-1}$, we need to show two things. First, we need to convince ourselves that $f^{-1}$ even exists in the first place – if it doesn't, then there's no possible way that $g = f^{-1}$! Let's start with that. This might initially seem tricky: how do you show an object exists without necessarily finding it? Fortunately, we have our result from earlier that says that if $f$ is a bijection, then $f^{-1}$ exists. Therefore, let's try to show that $f$ is a bijection. If we can do that, we're making progress toward the ultimate result. If we can't do that, then we're done!

So why would $f$ need to be a bijection for $g \circ f = \text{id}_A$ and $f \circ g = \text{id}_B$? Let's think about what happens if $f$ isn't a bijection. That would mean that it's either not injective, not surjective, or neither injective nor surjec-

tive. If $f$ isn't surjective, it means that there must be some $b \in B$ such that $f(a) \neq b$ for any $a \in A$. "Okay," you might say, "so what?" Well, one of our assumptions is that $f \circ g = \text{id}_B$. In particular, that would mean that $(f \circ g)(b) = b$, where $b$ is the specific $b$ we just mentioned (the $b \in B$ that's not covered by $f$). But since $(f \circ g)(b) = f(g(b))$, and $f(a) \neq b$ for *any* $a \in A$, we know that $(f \circ g)(b) \neq b$. That's impossible, since we know $(f \circ g)(b) = b$. Therefore, simply because $f \circ g = \text{id}_B$, we're *guaranteed* that $f$ has to be surjective. That wasn't at all obvious from our assumptions, but it must be true!

Similarly, let's think about what happens if $f$ isn't injective. So, there must be some $a_0$, $a_1 \in A$ where $a_0 \neq a_1$, but $f(a_0) = b = f(a_1)$ for some $b \in B$. Does that matter? Well, when working with surjectivity, we saw that if $f$ wasn't surjective, it wasn't possible for $f \circ g = \text{id}_B$ to be true. What happens if we look at the fact that $g \circ f = \text{id}_A$? This means that $(g \circ f)(a_0) = a_0$ and that $(g \circ f)(a_1) = a_1$. Now, we can tease apart the definition of $g \circ f$ to see that this says $g(f(a_0)) = a_0$ and $g(f(a_1)) = a_1$. But wait a minute – we know that $f(a_0) = b = f(a_1)$. That means that $g(b) = a_0$ and $g(b) = a_1$, which in turn means that $a_0 = a_1$. That's impossible, since we know $a_0 \neq a_1$! Therefore, we *necessarily* must have that $f$ is injective. As before, this is not at all obvious from the assumptions we're making!

At this point, let's take a break to formalize the two arguments we just made. Even if the overall line of reasoning that if $g \circ f = \text{id}_A$ and $f \circ g = \text{id}_B$ turns out to be false, we still just found a nontrivial result that we might want to use later on.

***Theorem:*** Let $f : A \to B$ and $g : B \to A$. If $g \circ f = \text{id}_A$ and $f \circ g = \text{id}_B$, then $f$ is a bijection.

***Proof:*** By contradiction; suppose $f$ is not a bijection. We consider two cases:

*Case 1: $f$* is not surjective. Then there exists some $b \in B$ such that $f(a) \neq b$ for all $a \in A$. Since we know $f \circ g = \text{id}_B$, we know $(f \circ g)(b) = b$. But $(f \circ g)(b) = f(g(b)) \neq b$, since $f(a) \neq b$ for all $a \in A$ and therefore for $g(b)$ in particular. This contradicts that $(f \circ g)(b) = b$.

*Case 2: $f$* is not injective. Then there exists some $a_0$, $a_1 \in A$ such that $a_0 \neq a_1$ but $f(a_0) = f(a_1)$. Let $b \in B$ be the value such that $f(a_0) = b = f(a_1)$. Since $g \circ f = \text{id}_A$, we know $(g \circ f)(a_0) = a_0$ and that $(g \circ f)(a_1) = a_1$. Therefore, $g(f(a_0)) = a_0$ and $g(f(a_1)) = a_1$. Rewriting these expressions in terms of $b$, we see $g(b) = a_0$ and $g(b) = a_1$, from which we get $a_0 = a_1$. This contradicts that $a_0 \neq a_1$.

In both cases we reach a contradiction, so our assumption must have been wrong. Thus $f$ must be a bijection. ■

Cool! At this point, we know that $f^{-1}$ exists. So now we need to decide whether the simple assumptions that $g \circ f$ and $f \circ g$ are identity functions means that $g = f^{-1}$. Given that these assumptions seemed to magically imply that $f$ has to be a bijection, it just might be possible!

Let's play around with the definitions and see what we find. To prove that $g = f^{-1}$, we need to prove that for all $b \in B$ that $g(b) = f^{-1}(b)$. We've defined $f^{-1}(b)$ so that $f^{-1}(b) = a$ iff $f(a) = b$, so this means that we need to prove that for all $b \in B$, that $g(b) = a$ iff $f(a) = b$. This statement is a biconditional, so let's try proving both definitions of implication.

First, let's see if we can show that if $g(b) = a$, then $f(a) = b$. Since we know $f \circ g = \text{id}_B$, we know that for any $b \in B$ we should have $(f \circ g)(b) = b$. Let's let $a = g(b)$. Then we have $(f \circ g)(b) = f(g(b)) = f(a)$. We also have that $(f \circ g)(b) = b$, so combining these together we get that if $g(b) = a$, then $f(a) = b$. Cool!

For the other direction of implication, we can use pretty much the exact same argument, but reversed. We'll try to show that if $f(a) = b$, then $g(b) = a$. To do that, we'll use the fact that $g \circ f = \text{id}_B$ to get that $(g \circ f)(a) = a$, then use the observation that $(g \circ f)(a) = g(f(a)) = g(b)$. This means $g(b) = a$, and we're done!

Formalizing this line of reasoning tends to be pretty straightforward. In fact, here it is!

***Theorem:*** Let $f : A \to B$ and $g : B \to A$. If $g \circ f = \text{id}_A$ and $f \circ g = \text{id}_B$, then $g = f^{-1}$.

***Proof:*** By our earlier theorem, we know that $f$ is invertible, so $f^{-1}$ exists. We therefore need to show that $g = f^{-1}$ by showing that $g(b) = f^{-1}(b)$ for all $b \in B$. By definition, $f^{-1}(b) = a$ iff $f(a) = b$. To show that $g(b) = f^{-1}(b)$, we therefore need to show that $g(b) = a$ iff $f(a) = b$. To do this, we prove both directions of implication.

( $\Rightarrow$ ) First, we prove that if $g(b) = a$, then $f(a) = b$. Since $f \circ g = \text{id}_B$, we have

$$b = (f \circ g)(b) = f(g(b)) = f(a)$$

So $f(a) = b$, as required.

( $\Leftarrow$ ) Next, we show that if $f(a) = b$, then $g(b) = a$. Since $g \circ f = \text{id}_A$, we have

$$a = (g \circ f)(a) = g(f(a)) = g(b)$$

So $g(b) = a$, as required. ■

This gives us a simple and direct way to check if one function is an inverse of another: check whether composing the two functions in both possible ways yields the identity function. If so, then the functions are inverses of one another. If not, then they are note.

## 6.3.4 Inverses of Inverses

To conclude our treatment of inverse functions, let's prove one final result about inverse functions. Let's suppose $f : A \to B$ is invertible. This means $f^{-1} : B \to A$ exists. What do we know about $f^{-1}$? Well, we know by definition that $f^{-1}(y) = x$ iff $f(x) = y$, and as we proved earlier, we know that $f^{-1} \circ f = \text{id}_A$ and that $f \circ f^{-1} = \text{id}_B$. We also know that since $f$ is invertible, $f$ is a bijection. However, we haven't yet established the following:

- Is $f^{-1}$ invertible? Equivalently, is $f^{-1}$ a bijection?
- If $f^{-1}$ is invertible, what is its inverse? That is, what is $(f^{-1})^{-1}$?

Let's see if we can resolve these questions before we continue.

Let's have $f : A \to B$ be invertible. We proved that $f^{-1} \circ f = \text{id}_A$ and that $f \circ f^{-1} = \text{id}_B$. Using the theorem from the previous section, we can conclude that $f^{-1}$ must be the inverse of $f$. However, we can *also* apply the theorem in a different way to conclude that $f$ must be the inverse of $f^{-1}$. This gives us the following result:

***Theorem:*** If $f$ is invertible, then $f^{-1}$ is invertible and $(f^{-1})^{-1} = f$. Consequently, $f^{-1}$ is a bijection.

# 6.4 Cardinality

At the very start of our exploration of the mathematical foundations of computing, we sketched a proof that there are problems that cannot be solved by computers. This proof relied on Cantor's Theorem, the fact that the cardinality of any set is always strictly smaller than the cardinality of its power set.

When we wrote that "proof," we had not actually discussed how to write a formal mathematical proof. We were imprecise with our definitions and did not rigorously structure our arguments. Given the mathematical machinery that we've built up so far, we are now ready to revisit that result and formally prove Cantor's theorem, from which the troubling but important result that unsolvable problems exist follows as a corollary.

In order to do this, we need to begin by revisiting cardinality. If you'll recall, we intuitively defined the cardinality of a set to be the number of elements it contains. We saw that this definition was troublesome when applied to infinite sets, since different infinite sets can have different sizes. To address this, we described how to check whether two sets have equal cardinalities (namely, finding a one-to-one correspondence between those sets), then used this definition in our exploration of Cantor's Theorem. Our very first step, therefore, will be to give a rigorous and precise definition of cardinalities.

## 6.4.1 Cardinalities Exist Between Sets

When we write expressions involving set cardinalities, we typically write expressions like $|A| = |B|$ or $|A| < |B|$. These expressions suggest that for sets $A$ and $B$ there exist actual quantities called $|A|$ and $|B|$ that are somehow related to one another (perhaps they're equal, or perhaps one is larger than the other).

When set theory was first developed in the late 1800s, mathematicians initially treated set cardinality this way. It was assumed that you could come up with a collection of quantities called "cardinalities" that could be used to measure infinite sets. For example, all the natural numbers are cardinalities of finite sets, while cardinalities like $\aleph_0$ could be used to measure infinite sets.

Around the turn of the twentieth century, it was discovered that this definition was problematic. Depending on what properties were assumed about how sets behaved, it was shown that there could exist sets with incomparable cardinalities – neither set was "bigger" than the other, yet the two sets couldn't have the same size. This made it difficult to define what exactly a "cardinality" was.

The ultimate resolution to the debate was the recognition that there can be multiple different definitions of what cardinality means based on what assumptions are initially made about the properties of sets. In the most common formalization of set theory (called *ZFC*, for "Zermelo-Fraenkel axioms with Choice"), cardinalities are actual, mathematical objects with well-defined properties. In other formalizations of set theory, there are no actual objects called cardinalities. Rather, when we say "$A$ has greater cardinality than $B$" or "$A$ and $B$ have the same cardinality," we are making a judgment about how $A$ and $B$ relate, rather than saying that $|A|$ and $|B|$ exist and relate to one another in some way. This is trickier and more nuanced view of cardinality, since in a sense it says that cardinalities exist "between" sets as judgments about size, rather than as actual concrete objects that can be manipulated directly.

In this chapter (and going forward), we will adopt this latter view. Specifically, we will give definitions for statements like $|A| = |B|$, $|A| \leq |B|$, and $|A| < |B|$ that do not rely on the objective existence of objects like $|A|$ and $|B|$. In this way, we will only discuss *relative* cardinalities of sets rather than *absolute* cardinalities of

sets. This lets our discussion avoid important questions like what cardinalities even are in the first place, which (while fascinating) quickly leads to difficult foundational questions about the nature of mathematics itself. If you're curious to see why exactly this is, consider taking a course in set theory.

## 6.4.2 Equal Cardinalities

We'll begin our formalization of cardinality by talking about what it means for two sets to have the same cardinality as one another. In Chapter One, we described sets as having equal cardinality if there was a one-to-one correspondence between the elements of the two sets. Using our more precise mathematical terminology, a one-to-one correspondence is the same as a bijection. Therefore, we can formalize our definition of equal cardinalities as follows:

> Two sets $A$ and $B$ have the same cardinality (denoted $|A| = |B|$) iff there exists a bijection $f : A \to B$.

This definition is a bit subtle, so let's dissect it a bit. First, note the notation: we write $|A| = |B|$ to denote the statement "$A$ and $B$ have the same cardinality." As mentioned in the introduction to this section, this statement is *not* a judgment about the objects $|A|$ and $|B|$; in fact, we're not even assuming they exist. Rather, treat $|A| = |B|$ as a statement about $A$ and $B$ rather than the cardinalities $|A|$ and $|B|$.

As an example, suppose we let $A = \{1, 2, 3\}$ and $B = \{a, b, c\}$. Then $|A| = 3$ and $|B| = 3$, so using our naïve conception of cardinality we'd expect $|A| = |B|$. To formally prove this, though, we'd need to find an bijection $f : A \to B$. Below is one such function that does this:

$$f(x) = \begin{cases} a & \text{if } x = 1 \\ b & \text{if } x = 2 \\ c & \text{if } x = 3 \end{cases}$$

It seems strange to define equality of cardinality this way when these two sets "obviously" have the same size. The reason we're adopting this more heavyhanded approach is that it generalizes nicely to infinite sets, where our conceptions about what objects are "obviously" the same size as one another is decidedly less obvious.

Next, note that the definition says that $|A| = |B|$ iff *there exists* a bijection $f : A \to B$. There can be a staggeringly large number of functions between two sets $A$ and $B$, and if even a single one of them ends up being a bijection we will say that the sets have the same cardinality. This is one of the reasons why infinity can be so counterintuitive. If we let $E = \{ n \in \mathbb{N} \mid n \text{ is even} \}$ be the set of all even natural numbers, it just "feels" like there are more natural numbers than even numbers. However, the fact that we can find even one bijection $f : E \to \mathbb{N}$ (for example, $f(n) = 2n$) means that the two sets do indeed have the same cardinality. Even though it's possible to find functions from $E$ to $\mathbb{N}$ that *aren't* bijections (for example, the function $g(n) = n$), this doesn't mean that $|E| \neq |\mathbb{N}|$. It just means that this one particular functions wasn't a bijection. In fact, showing that two sets *don't* have the same cardinality is quite involved. We'll talk about that more in the next section when we discuss diagonalization.

Now that we have a rigorous definition of equal cardinality, let's start playing around with the definition to see if we can uncover some of its properties. If you look at our notation for when $A$ and $B$ have the same cardinality (namely, $|A| = |B|$), it seems reasonable to expect that the relation "has the same cardinality as" would behave like an equivalence relation. We'd expect that

- $|A| = |A|$ for any set $A$.
- If $|A| = |B|$, then $|B| = |A|$.
- If $|A| = |B|$ and $|B| = |C|$, then $|A| = |C|$.

It turns out that actually proving these statements are true is a bit trickier than it might look. For example, let's take the simple statement that if $|A| = |B|$, then $|B| = |A|$. It might seem like it's *really obvious* that this statement is true – after all, if $x = y$, then $y = x$! However, that's just a consequence of the notation we've chosen. By writing "$A$ has the same cardinality as $B$" as $|A| = |B|$, we've implied that $|B| = |A|$. If we chose a different notation for the statement "$A$ has the same cardinality as $B$," say, something like $A \Rightarrow B$, then it wouldn't at all be obvious that if $A \Rightarrow B$, then $B \Rightarrow A$.

In order to prove this statement, we'll need to call back to the definition. Let's suppose that $|A| = |B|$. To prove that $|B| = |A|$, we need to show that there must be a bijection $f : B \to A$. It now becomes clearer why this is a bit tricky: we don't know anything about $A$ and $B$ except that $|A| = |B|$, and yet we have to find a bijective function $f : B \to A$. How exactly can we do this?

Well, let's look at what we have right now. We're assuming $|A| = |B|$, so even though we don't know anything about what elements $A$ and $B$ contain, we do know that there must be a bijection $g : A \to B$ (this is just the definition of $|A| = |B|$). Given this function as a starting point, could we find a bijection $f : B \to A$?

Fortunately, the answer is "yes." Notice that $g : A \to B$ starts at $B$ and ends at $A$. We want to find a function $f : B \to A$ that starts at $B$ and ends at $A$. This seems a lot like how inverse functions work. Thinking back to what we've seen before, you might remember that we've proven the following:

- If $f$ is a bijection, $f$ is invertible.
- If $f$ is a bijection and $f$ is invertible, then $f^{-1}$ is also a bijection.

These two statements taken together give us a way of finding the function we need in order to prove that $|B| = |A|$. Specifically, since $|A| = |B|$, we know there is some bijection $g : A \to B$. Its inverse $g^{-1} : B \to A$ exists, and moreover it's a bijection between $B$ and $A$. Therefore, there has to be at least one bijection from $B$ to $A$: just take $g^{-1}$!

We can formalize this as a proof without needing to add much more:

***Theorem:*** For any sets $A$ and $B$, if $|A| = |B|$, then $|B| = |A|$.

***Proof:*** Suppose $|A| = |B|$. This means there must be some bijection $g : A \to B$. Since all bijections are invertible, this means that $g^{-1} : B \to A$ exists. By our earlier theorem, $g^{-1}$ is a bijection. Therefore, there is a bijection from $B$ to $A$ (namely, $g^{-1}$), so $|B| = |A|$. ■

This theorem isn't all that complicated on its own, though we had to do a lot of math earlier on when working with inverse functions in order to get the key result (that $g^{-1}$ is a bijection when $g$ is invertible). Notice that the entire proof hinges around finding a bijection from $B$ to $A$. We made no reference to the "number" of elements in $A$ or $B$, nor did we try to select specific examples of $A$ and $B$. Many proofs about cardinalities work this way.

Additionally, note that the above proof simply guarantees that there has to be some bijection from $B$ to $A$. We never actually said what that bijection is. Sure, we know the bijection must be the inverse of the bijec-

tion from $A$ to $B$, but our proof never actually knew what the bijection was. In fact, if someone came to you and said "I promise you that $A$ and $B$ have the same cardinality" without telling you a bijection from $A$ to $B$, the above proof would guarantee you that $B$ and $A$ have the same cardinality, but wouldn't tell you how to find the bijection between them!

Let's do another example and try to prove transitivity: that if $|A| = |B|$ and $|B| = |C|$, then $|A| = |C|$. As before, from the notation this result might seem "obvious," but if we call back down to the definitions the picture is decidedly more complicated. Here, we're assuming there is a bijection $f : A \to B$ and a bijection $g : B \to C$ and trying to conclude that there must be a bijection $h : A \to C$. How exactly might we do this?

If you'll remember to our discussion of function composition, you might remember that we proved that the composition of two bijections is itself a bijection. Notice that it's legal here to compose $g$ and $f$ together, since $f$ has codomain $B$ and $g$ has domain $B$. Therefore, $g \circ f : A \to C$ is a bijection from $A$ to $C$, which is exactly what we're looking for! The proof of this result is essentially the argument from this paragraph made a bit more rigorous, and is given bellow:

***Theorem:*** If $|A| = |B|$ and $|B| = |C|$, then $|A| = |C|$.

***Proof:*** Since $|A| = |B|$ and $|B| = |C|$, there must be bijection $f : A \to B$ and $g : B \to C$. By our earlier result, because $f$ and $g$ are bijections, we know $g \circ f$ is a bijection as well. Moreover, $g \circ f : A \to C$. Therefore, there is a bijection from $A$ to $C$ (namely, $g \circ f$). Thus $|A| = |C|$. ■

Finally, let's prove that equality of cardinality is reflexive: that is, $|A| = |A|$ for any set $A$. In the previous two proofs, we started with an initial assumption (in the case of symmetry, that $|A| = |B|$; in the case of transitivity, that $|A| = |B|$ and $|B| = |C|$), then used that assumption to show that some sets must have equal cardinality. In this case, we don't begin with any such assumptions. We need to directly show $|A| = |A|$ regardless of what $A$ is.

To show $|A| = |A|$, we need to find a bijection $f : A \to A$. Our choice of $f$ will depend on what $A$ is (because $A$ will be different every time). This might seem like an impossible task – how can you possibly define a function over a set $A$ if you don't know anything about what $A$ contains? – but fortunately there's an extremely simple function we can use. What happens if we use the function $\text{id}_A$? This function does indeed go from $A$ to $A$. If we can show it's a bijection, then we're done; $|A| = |A|$ for any $A$ because $\text{id}_A$ serves as a bijection from $A$ to itself.

With a bit of thought, we can see why $\text{id}_A$ has to be a bijection. It's injective, since if $\text{id}_A(a_0) = \text{id}_A(a_1)$, then $a_0 = a_1$. It's surjective, since if you take any $a \in A$, there is some $a' \in A$ such that $\text{id}_A(a') = a$; just take $a' = a$.

Armed with this insight, let's prove $|A| = |A|$ for any set $A$.

***Theorem:*** For any set $A$, we have $|A| = |A|$.

***Proof:*** Consider any set $A$. We claim $\text{id}_A : A \to A$ is a bijection, from which $|A| = |A|$ immediately follows. To prove that $\text{id}_A$ is a bijection, we prove that it is injective and surjective.

To see that $\text{id}_A$ is injective, consider any $a_0, a_1 \in A$ for which $\text{id}_A(a_0) = \text{id}_A(a_1)$. We will prove that $a_0 = a_1$. To see this, note that since $\text{id}_A(a_0) = \text{id}_A(a_1)$, we have $a_0 = a_1$, as required.

To see that $\text{id}_A$ is surjective, consider any $a \in A$. We will prove that there is an $a' \in A$ such that $\text{id}_A(a') = a$. To see this, take $a' = a$; then $\text{id}_A(a') = \text{id}_A(a) = a$, as required. ■

## 6.4.3 Countable Sets

Now that we have a formal definition of equal cardinalities, we can start rigorously exploring cardinality as applied to infinite sets. The remainder of this chapter will focus on this question, first by exploring infinite sets with equal cardinalities, then exploring infinite sets with unequal cardinalities.

One of the most commonly-used infinite sets is $\mathbb{N}$, the set of all natural numbers. Many infinite sets have the same cardinality as $\mathbb{N}$. In Chapter 1, you saw that the set of all even natural numbers and the set of all perfect squares has the same cardinality as $\mathbb{N}$. These sets are all infinite sets, but since there are multiple different infinities it's actually somewhat imprecise simply to call them "infinite." When we want to say that a set specifically has the same size as the set of all natural numbers, we use the term *countably infinite*:

A set $S$ is called ***countably infinite*** iff $|\mathbb{N}| = |S|$.

The term "countable" here comes from the following observation. Suppose that $S$ is a countably infinite set. This means that there has to be a function $f : \mathbb{N} \to S$. Consequently, we can list off all of the elements of $S$ in some order as $f(0)$, $f(1)$, $f(2)$, $f(3)$, … . In this way, we can "count off" the elements of $S$, so $S$ is "countable." We'll talk about *uncountable* sets later on in the chapter when we discuss diagonalization, and will rely extensively on this intuition.

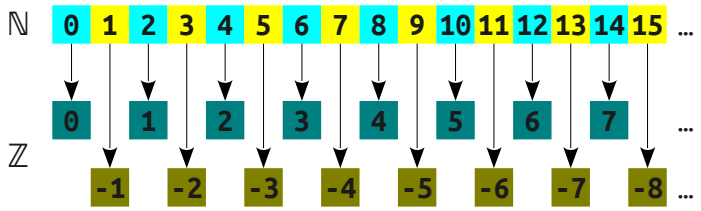
### 6.4.3.1 $\mathbb{Z}$ is Countable

So far, the countably infinite sets we've seen so far have all been subsets of $\mathbb{N}$. $\mathbb{N}$ itself is trivially countably infinite (because $|\mathbb{N}| = |\mathbb{N}|$), as is the set of all even numbers, the set of all odd numbers, the set of all perfect squares, etc. Using some trickier math, we can also show that the set of all prime numbers is countably infinite.

However, there are many sets that intuitively seem "bigger" than $\mathbb{N}$, but are still countably infinite. A great example of this is the set $\mathbb{Z}$ of all integers. Initially, it might seem like $\mathbb{Z}$ is twice as large as $\mathbb{N}$, since $\mathbb{N}$ contains the nonnegative integers 0, 1, 2, 3, …, while $\mathbb{Z}$ contains both the nonnegative and negative integers. In fact, the integers -1, -2, -3, -4, … etc. are all contained in $\mathbb{Z}$ but not in $\mathbb{N}$.

However, that shouldn't deter us. There's the same number of even natural numbers as there are natural numbers even though it seems like there should be twice as many natural numbers as even natural numbers, so it doesn't seem totally unreasonable that $\mathbb{Z}$ should be countably infinite.

In fact, we can use the fact that there's the same number of even naturals as there are naturals and the same number of odd naturals as there are naturals in order to set up a bijection between $\mathbb{N}$ and $\mathbb{Z}$. Here's the idea: we can treat the integers as two infinite sequences, the sequence 0, 1, 2, 3, …, and the sequence -1, -2, -3, … . We'll try to set up a bijection so that the even natural numbers in $\mathbb{N}$ map to 0, 1, 2, 3, … in $\mathbb{Z}$ and the odd natural numbers in $\mathbb{Z}$ map to -1, -2, -3, … in $\mathbb{Z}$. This is shown below:




The result of trying to set up a bijection like this one is that we try to "list off" the integers in the order 0, -1, 1, -2, 2, -3, 3, -4, 4, … . Intuitively, we can see that every integer has to be in this list somewhere, since if you're thinking of the integer $x$ and keep listing off numbers in this order, the absolute values of the listed numbers will continue to increase until $x$ is generated.

At this point, all we have is an intuition for how this bijection would be set up. We have yet to actually *define* the bijection or to prove that it's correct. Let's fix that. We're going to try to define a bijective function $f : \mathbb{N} \to \mathbb{Z}$ that matches the above behavior. Intuitively, we defined this function by interleaving two different sequences, one defined only for the even numbers and one defined only for the odd numbers. Consequently, it's probably easiest to define this function as a piecewise function with separate cases for when $n$ is even or odd.

In the case where $n$ is even, we want to have $f(0) = 0$, $f(2) = 1$, $f(4) = 2$, $f(6) = 3$, etc. In this case, the pattern seems to be that $f(n) = n / 2$. In the case where $n$ is odd, we want to have $f(1) = -1$, $f(3) = -2$, $f(5) = -3$, $f(7) = -4$, etc. There's clearly a pattern here, but it's a little bit less obvious than before. With a bit of math (or just a good guess), we can find that in this case, $f(n) = -(n + 1) / 2$. Combining these together, we end up with this function:

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ -(n+1)/2 & \text{if } n \text{ is odd} \end{cases}$$

So now we have our function $f : \mathbb{N} \to \mathbb{Z}$. In order to prove $|\mathbb{N}| = |\mathbb{Z}|$, we need to prove that $f$ is a bijection. We can do this in several ways. One option would be to prove that it's injective and surjective, though since the function is defined piecewise this ends up actually being a bit messy. A cleaner way to do this is to use our result from earlier that states that a function is a bijection iff it's invertible. Therefore, if we can find an inverse function for $f$, we know it's a bijection.

So how do we find the inverse for $f$? Well, one way to do that would be to find a function $g$ such that $g \circ f = \text{id}_{\mathbb{N}}$ and $f \circ g = \text{id}_{\mathbb{Z}}$. Let's see if we can do this.

Let's look at this first constraint: $g \circ f = \text{id}_{\mathbb{N}}$. This means that we need to define $g$ such that $(g \circ f)(n) = n$ for all $n \in \mathbb{N}$. Since $(g \circ f)(n) = g(f(n))$, this means we need to find $g$ such that $g(f(n)) = n$ for all $n \in \mathbb{N}$. Since our function $f$ is defined piecewise, this means that we want to define $g$ such that the following is true:

- If $n$ is even, then $f(n) = n / 2$ and we need to define $g$ such that $g(n / 2) = n$. This happens when we define $g(x) = 2x$.
- If $n$ is odd, then $f(n) = -(n + 1) / 2$ and we need to define $g$ such that $g(-(n + 1) / 2) = n$. If we do a bit of arithmetic, we can find that this happens when $g(x) = -2x - 1$.

So this looks good – if $n$ is even, we define $g$ one way, and if $n$ is odd, we define $g$ another way! However, there's a subtlety to this argument that we need to be careful about. In particular, we do **not** want to define $g$ this way:

$$g(x) = \begin{cases} 2x & \text{if } x \text{ is even} \\ -2x - 1 & \text{if } x \text{ is odd} \end{cases} \quad \textit{(this is wrong!)}$$

This function is incorrect! In particular, look at $g(3)$; it comes out to -7. That's a problem, because we're trying to define $g : \mathbb{Z} \to \mathbb{N}$, and -7 is not a natural number!

The problem here is that the cases we listed for $g$ above were chosen so that $g(f(n)) = n$. In other words, the cases correspond to whether the inputs $n$ to the function $f$ are even or odd, rather than to the inputs $x$ to the function $g$ are odd or even. We want to choose $g$ so that if the input to $f$ is even, then $g$ evaluates to $2x$ and so that if the input to $f$ is odd, then $g$ evaluates to $-2x - 1$. Since we need to define $g$ in terms of its inputs and not in terms of the inputs to $f$, we need to be a bit more careful with how we set up $g$.

Fortunately, there's an easy fix for this. If we compute $f(n)$ when $n$ is even, then we get $n / 2$ which will be nonnegative. If we compute $f(n)$ when $n$ is odd, then we get $-(n + 1) / 2$, which will be negative. Consequently, we can decide which case we're in in $g$ by looking at whether the input is negative or nonnegative. If the input to $g$ is nonnegative, then $f(n)$ was evaluated on even $n$. If the input to $g$ is negative, then $f(n)$ was evaluated on odd $n$. Therefore, we can define the cases of $g$ as follows:

$$g(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ -2x - 1 & \text{if } x < 0 \end{cases} \quad \textit{(this is correct!)}$$

Now, we can quickly check that $g(f(n)) = n$ by going through two cases, one for when $n$ is even and one for when $n$ is odd. Fortunately, the math checks out.

Given our guess of $g$ as $f^{-1}$, we can try to write a short proof that $f$ is a bijection. To do this, we'll prove both that $g \circ f = \text{id}_{\mathbb{N}}$ and $f \circ g = \text{id}_{\mathbb{Z}}$. Although we only defined $g$ such that the first of these statements check out, it turns out that both cases will hold.

***Theorem:*** $|\mathbb{N}| = |\mathbb{Z}|$.

***Proof:*** We will exhibit a bijection $f : \mathbb{N} \to \mathbb{Z}$. Define $f$ as

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ is even} \\ -(n+1)/2 & \text{if } n \text{ is odd} \end{cases}$$

To prove that $f$ is a bijection, we will show that it is invertible. To do so, we will prove that the following function $g : \mathbb{Z} \to \mathbb{N}$ is the inverse of $f$:

$$g(x) = \begin{cases} 2x & \text{if } x \geq 0 \\ -2x - 1 & \text{if } x < 0 \end{cases}$$

To show that $g = f^{-1}$, we will show that $g \circ f = \text{id}_{\mathbb{N}}$ and $f \circ g = \text{id}_{\mathbb{Z}}$.

Since $f : \mathbb{N} \to \mathbb{Z}$ and $g : \mathbb{Z} \to \mathbb{N}$, we know $g \circ f : \mathbb{N} \to \mathbb{N}$. Thus to show that $g \circ f = \text{id}_{\mathbb{N}}$, we only need to show that $(g \circ f)(n) = n$ for all $n \in \mathbb{N}$. So consider any $n \in \mathbb{N}$; we consider two cases:

*Case 1:* $n$ is even. Then $(g \circ f)(n) = g(f(n)) = g(n / 2)$. Since $n \in \mathbb{N}$, we know $n \geq 0$, so $n / 2 \geq 0$ as well. Therefore, $g(n / 2) = 2(n / 2) = n$, as required.

*Case 2:* $n$ is odd. Then $(g \circ f)(n) = g(f(n)) = g(-(n + 1) / 2)$. Since $n \in \mathbb{N}$, we know $n \geq 0$, so we have $n + 1 > 0$. Because $n$ is odd, $n + 1$ is even, and since $n + 1 > 0$, this means $n + 1 \geq 2$. Therefore, $(n + 1) / 2 \geq 1 > 0$, so $-(n + 1) / 2 < 0$. Then $g(-(n + 1) / 2) = -2(-(n + 1) / 2) - 1 = n$, as required.

Thus $(g \circ f)(n) = n$ for all $n \in \mathbb{N}$, so $g \circ f = \text{id}_{\mathbb{N}}$.

Next, we will prove that $f \circ g = \text{id}_{\mathbb{Z}}$. Since $g : \mathbb{Z} \to \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{Z}$, we have $f \circ g : \mathbb{Z} \to \mathbb{Z}$, so to show that $f \circ g = \text{id}_{\mathbb{Z}}$ we need to show $(f \circ g)(x) = x$ for all $x \in \mathbb{Z}$. So consider any $x \in \mathbb{Z}$; we consider two cases:

*Case 1:* $x \geq 0$. Then $(f \circ g)(x) = f(g(x)) = f(2x)$. Since $2x$ is even, $f(2x) = (2x) / 2 = x$, as required.

*Case 2:* $x < 0$. Then $(f \circ g)(x) = f(g(x)) = f(-2x - 1)$. Since $-2x - 1$ is odd, $f(-2x - 1) = -((-2x-1)+1)/2 = x$, as required.

Thus $(f \circ g)(x) = x$ for all $x \in \mathbb{Z}$, so $f \circ g = \text{id}_{\mathbb{Z}}$. Thus $g = f^{-1}$, so $f$ is invertible and therefore a bijection. Therefore, $|\mathbb{N}| = |\mathbb{Z}|$. ■

This proof is a bit lengthy and is a lot of math, which I think is rather unfortunate – the intuition is pretty clear, but once we get into messy symbolic manipulation the beauty is hidden. However, the proof does get the job done, and we now have that $\mathbb{Z}$ is countable, meaning there's the same number of integers as there are natural numbers. That's not at all obvious!

### 6.4.3.2 $\mathbb{N}^2$ is Countable

We have now seen several examples of countably infinite sets: $\mathbb{N}$, $\mathbb{Z}$, the set of all even naturals, and the set of all odd naturals. Although $\mathbb{Z}$ seems twice as big as $\mathbb{N}$, which in turn seems twice as big as the set of all even natural numbers, all of these sets have the same size.

It turns out that there are other countable sets that, at first glance, seem *significantly* larger than any of these sets. One of the more interesting examples of this is the set $\mathbb{N}^2$, the set of all pairs of natural numbers. (Recall that the notation $\mathbb{N}^2$ is shorthand for $\mathbb{N} \times \mathbb{N} = \{ (m, n) \mid m \in \mathbb{N} \text{ and } n \in \mathbb{N} \}$).

Compared with $\mathbb{N}$ and $\mathbb{Z}$, the set $\mathbb{N}^2$ just seems downright huge. Visually, you can think of $\mathbb{N}^2$ as an infinite two-dimensional grid like this:

| | **0** | **1** | **2** | **3** | **…** |
|---|---|---|---|---|---|
| **0** | (0, 0) | (0, 1) | (0, 2) | (0, 3) | … |
| **1** | (1, 0) | (1, 1) | (1, 2) | (1, 3) | … |
| **2** | (2, 0) | (2, 1) | (2, 2) | (2, 3) | … |
| **3** | (3, 0) | (3, 1) | (3, 2) | (3, 3) | … |
| **…** | … | … | … | … | … |

To give you a sense for how massively huge this set is, every row and column of this set has cardinality $|\mathbb{N}|$, as does the diagonal consisting of (0, 0), (1, 1), (2, 2), etc., and many other subsets. Intuitively, this set just *feels* like it's much bigger than $\mathbb{N}$.

Amazingly, though, this turns out not to be the case. In the late 19th century, Georg Cantor (the same mathematician responsible for Cantor's Theorem) proved the following result:

> ***Theorem:*** $|\mathbb{N}| = |\mathbb{N}^2|$.

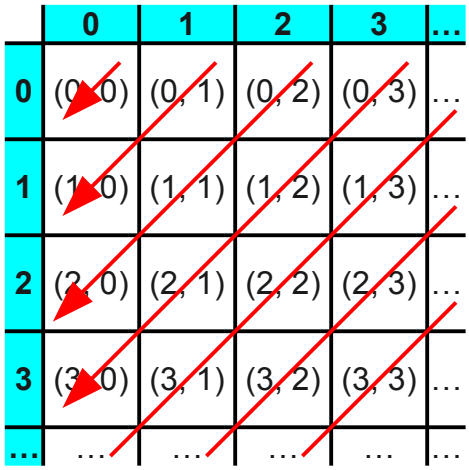
To prove this result, Cantor came up with a marvelous function called the *Cantor pairing function* that is a bijection from $\mathbb{N}^2$ to $\mathbb{N}$. The function is deceptively simple:

> The ***Cantor pairing function*** is the function $\pi : \mathbb{N}^2 \to \mathbb{N}$ where $\pi(a, b) = (a + b)(a + b + 1) / 2 + a$.

Where on earth did this function come from? How on earth is it a bijection?

There is a beautiful intuition for this function, though actually proving that it's a bijection is nontrivial. Recall that, intuitively speaking, a countably infinite set is one where all the elements of the set can be listed off as the 0th, 1st, 2nd, etc. in an order that includes all of the elements of the set. The Cantor pairing function is best described intuitively by explaining the order in which it lists off the elements of $\mathbb{N}^2$; given this ordering, it's actually not all that hard to see why it has to be a bijection.

Cantor's observation was to look at the two-dimensional grid of $\mathbb{N}^2$ and to notice that it's possible to draw infinitely many diagonal lines across the grid, like this:




Look at these diagonal lines in ascending order of length. The shortest diagonal has length 1 and consists of (0, 0). The second-shortest diagonal has length 2 and consists of (0, 1) and (1, 0). The third-shortest diagonal has length 3 and consists of (0, 2), (1, 1), and (2, 0). Notice that the sum of the first and second numbers in each pair on a diagonal is exactly the same: in the first diagonal it's 0, in the second it's 1, in the third it's 2, etc. More generally, the $n$th diagonal consists of all pairs of numbers that sum up to exactly $(n - 1)$.

Cantor's observation was that all pairs in $\mathbb{N}^2$ can be listed off in the order specified by these diagonals: first, we list off all the pairs on the first diagonal (whose sum is 0), then all the pairs on the second diagonal (whose sum is 1), then all the pairs on the third diagonal (whose sum is 2), etc. Within each diagonal, we can list off the pairs in ascending order of their first component. For example, the fourth diagonal consists of all the pairs that sum to three, and so we'd list them off as (0, 3), (1, 2), (2, 1), (3, 0).

Overall, this gives back the following listing of all the pairs in $\mathbb{N}^2$:

(0, 0), (0, 1), (1, 0), (0, 2), (1, 1), (2, 0), (0, 3), (1, 2), (2, 1), (3, 0), (0, 4), (1, 3), (2, 2), (3, 1), (4, 0), …

Given this ordering, it's a bit clearer why this would be a bijection. This function is surjective since for any pair $(m, n)$, there are only finitely many pairs in $\mathbb{N}^2$ whose components sum to less than $m + n$. After we've listed all of them off, we just have to wait until all numbers on the diagonal that come before $(m, n)$ are listed off before we eventually get $(m, n)$. The function is injective because we list the diagonals in ascending order and never repeat any numbers on the diagonal.

I'll leave the details of how to go from this intuition to the actual definition of the pairing function as an exercise. As a hint, think about how many elements come on the diagonal that precedes the diagonal of pairs whose elements sum to $k$, then think about how many elements precede a particular element on a diagonal.

Rigorously proving that this function is a bijection is hard. The function isn't at all obviously a bijection, and proving injectivity in particular is quite a challenge. Rather than dedicate several pages of hard math to formalizing the result, I'll leave it as a starred exercise at the end of the chapter.

### 6.4.4 Comparing Cardinalities

We now have a way of defining what it means for one set's size to be equal to another's; $|A| = |B|$ iff there is a bijection from $A$ to $B$. However, we don't currently have a way to rank cardinalities against one another. For example, what would it mean for $|A| \leq |B|$ to be true? How about $|A| < |B|$? This section explores that question.

Our initial goal will be to define what $|A| \leq |B|$ would mean. Intuitively, this means that there are at least as many elements in $B$ as there are in $A$, but there could potentially be more. How could we formalize that?

Well, when defining $|A| = |B|$, we used bijections because a bijection between $A$ and $B$ conceptually means that we can pair off elements of $A$ and $B$ such that all elements of $A$ and $B$ are covered. If $|A| \le |B|$, then we should be able to pair off elements of $A$ with elements of $B$ without running out of elements of $B$ in the process. However, we might not cover all the elements of $B$, since $B$ is *at least as large* as $A$ rather than *exactly the same size*.

To formalize the idea that "every element of $A$ is paired with a unique element of $B$" using functions, we can think about finding a function $f : A \to B$ that represents the pairing. The function $f$ will associate some element of $B$ with each element of $A$. If $f$ never associates the same element of $B$ with two distinct elements of $A$, then we can think of the function as defining a pairing of elements in $A$ with some (or all) of the elements of $B$. This property corresponds exactly to injectivity – we want to make sure that $f$ always maps different elements of $A$ to different elements of $B$.

This conceptual idea motivates the following definition:

> If $A$ and $B$ are sets, then $|A| \le |B|$ iff there exists an injection $f : A \to B$.

As with equality of cardinality, the notation $|A| \le |B|$ suggests that $|A|$ and $|B|$ are real objects and that we are making a judgment about them. However, the formal definition does not talk about $|A|$ and $|B|$ as actual objects and instead defines $|A| \le |B|$ purely in terms of functions.

As an example, suppose we let $A = \{1, 2, 3\}$ and $B = \{a, b, c, d\}$. Then $|A| = 3$ and $|B| = 4$, so using our naïve conception of cardinality we'd expect $|A| \le |B|$. To formally prove this, though, we'd need to find an injection $f : A \to B$. Below is one such function that does this:

$$f(x) = \begin{cases} a & \text{if } x = 1 \\ b & \text{if } x = 2 \\ c & \text{if } x = 3 \end{cases}$$

As another example, suppose we want to prove that $|\mathbb{N}| \le |\mathbb{R}|$. Intuitively, this should be obvious because every element of $\mathbb{N}$ is also an element of $\mathbb{R}$, so "clearly" $|\mathbb{R}|$ should be at least as big as $|\mathbb{N}|$. To formalize this, let's see if we can find an injective function $f : \mathbb{N} \to \mathbb{R}$. Well, since every natural number is also a real number, one possible function that does the trick is $f(n) = n$, where $f : \mathbb{N} \to \mathbb{R}$. Showing this is injective isn't too hard, and we'll leave it as an exercise to the reader. (I know we've been doing that a lot recently... sorry about that! I just want to let us get to more exciting results without getting bogged down in the details.)

The notation we've chosen for $|A| \le |B|$ suggests that certain mathematical properties should hold for the relation "$A$'s cardinality is no larger than $B$'s cardinality." For example, we'd expect that

- **Reflexivity:** For any set $|A|$, we have $|A| \le |A|$.
- **Antisymmetry:** If $|A| \le |B|$ and $|B| \le |A|$, then $|A| = |B|$.
- **Transitivity:** If $|A| \le |B|$ and $|B| \le |C|$, then $|A| \le |C|$.

All of these properties turn out to be true, but as with the relation "$A$ has the same cardinality as $B$," they're by no means obvious from the definitions. Let's take a few minutes to see why these results are true.

First, let's prove reflexivity, that for any set $A$, we have $|A| \le |A|$. Calling back to the definition, we need to show that for every set $A$, there exists some injective function $f : A \to A$. If you'll recall, we proved earlier that $\text{id}_A$ is a bijection from $A$ to $A$ for any set $A$, and since all bijections are injections, this claim follows immediately. Formalizing this as a proof is not really a challenge once we have this insight:

***Theorem:*** For any set $A$, we have $|A| \le |A|$.

***Proof:*** For any set $A$, the function $\text{id}_A : A \to A$ is a bijection. Therefore, $\text{id}_A$ is an injection from $A$ to $A$. Therefore, by definition, $|A| \le |A|$. ■

Next, let's look at transitivity: if $|A| \le |B|$ and $|B| \le |C|$, then $|A| \le |C|$. Translating back to definitions, this means that if there's an injection $f : A \to B$ and an injection $g : B \to C$, then there's an injection $h : A \to C$. The proof of this result is similar to our earlier proof that equality of cardinality is transitive: since $f$ and $g$ are injections, their composition is also an injection, and it goes from $A$ to $C$. This is formalized here:

***Theorem:*** If $|A| \le |B|$ and $|B| \le |C|$, then $|A| \le |C|$.

***Proof:*** Since $|A| \le |B|$ and $|B| \le |C|$, there exist injections $f : A \to B$ and $g : B \to C$. Then their composition $g \circ f : A \to C$ is also an injection by our earlier theorem. Since an injection exists from $A$ to $C$ (namely, $g \circ f$), we have $|A| \le |C|$. ■

Reflexivity and transitivity have been relatively straightforward to prove. What about antisymmetry? Let's suppose $|A| \le |B|$ and $|B| \le |A|$; can we prove $|A| = |B|$? Take a minute to think about what this means: if there is an injection $f : A \to B$ and an injection $g : B \to A$, could we prove there has to be a ***bijection*** $h : A \to B$? It's not at all obvious why this would be true. Take, for example, $\mathbb{N}$ and $\mathbb{Z}$. We can define injections from both sets into the other: for an injection $f : \mathbb{N} \to \mathbb{Z}$, we can use the function $f(n) = n$. For an injection $g : \mathbb{Z} \to \mathbb{N}$, we can use this function, which is indeed an injection (we'll let you verify that one on your own):

$$g(x) = \begin{cases} 4x & \text{if } x \ge 0 \\ -4x - 2 & \text{if } x < 0 \end{cases}$$

Both $f$ and $g$ are injections and neither $f$ nor $g$ is a bijection. If all we knew about were the functions $f$ and $g$, how would we use them to find a bijection from $\mathbb{N}$ to $\mathbb{Z}$? It's not at all clear that one exists in the first place (we had to do a bit of mental gymnastics to find one earlier one!), and if we didn't already know that one existed the existence of injections $f$ and $g$ doesn't seem to help much.

It turns out that by using some decidedly tricky math that it's possible to start with *any* two injective functions $f : A \to B$ and $g : B \to A$ and build a bijection $h : A \to B$ from them. In fact, this math is so tricky that it's often presented as a theorem in its own right and the proof is by no means obvious.

***Theorem (Cantor-Bernstein-Schroeder):*** If $|A| \le |B|$ and $|B| \le |A|$, then $|A| = |B|$.

The proof of this result is too advanced to include here and relies on certain tricky operations (namely, infinite unions of sets and chains of images and preimages) that are beyond the scope of what we're planning on covering. If you are interested in seeing a proof of this result, we suggest picking up a textbook on ax-

iomatic set theory. You might want to have a lot of scratch paper with you when you start reading through the proof.

The following results also hold for cardinalities and are similarly difficult to establish. In fact, these results are often used as building blocks in the proof of the Cantor-Bernstein-Schroeder Theorem:

> ***Theorem:*** If $|A| \leq |B|$, $|B| \leq |C|$, and $|A| = |C|$, then $|A| = |B| = |C|$.

# 6.5 Diagonalization

Up to this point, our discussion of cardinalities has considered two cases: the case where two sets have exactly the same size, and the case where one set is no larger than another. There is one final case to consider: the case where one set actually is larger than another set. In other words, we want to explore under what circumstances we would have $|A| < |B|$.

As with $|A| = |B|$ and $|A| \leq |B|$, we need to provide a rigorous definition of what it means for $|A| < |B|$. Intuitively, we'd expect that $|A| < |B|$ means that there are strictly more elements in $B$ than in $A$. For example, a set with three elements is strictly smaller than a set with five elements. To turn this intuition into a formal definition, we will need to be a bit more precise than this. The definition that's conventionally used for $|A| < |B|$ is the following, which is deceptively simple:

> If $A$ and $B$ are sets, then $|A| < |B|$ iff $|A| \leq |B|$ and $|A| \neq |B|$.

Restated in plain English, this says that $A$ is strictly smaller than $B$ if $A$ is no larger than $B$ and $A$ is not the same size as $B$. This makes intuitive sense if you consider the analogous case for real numbers: you can prove $x < y$ by showing $x \leq y$ and that $x \neq y$.

However, if we look more closely at this definition, we'll see that it's a bit more nuanced. Remember that $|A| \leq |B|$ and $|A| \neq |B|$ have very precise mathematical definitions. We just saw $|A| \leq |B|$, which means that there is some injection $f : A \to B$. But what does $|A| \neq |B|$ mean? Well, according to the definition, we have $|A| = |B|$ iff there exists a bijection $f : A \to B$. By negating both sides, we get that $|A| \neq |B|$ iff *there does not exist* a bijection $f : A \to B$. In other words, in order to prove that $|A| \neq |B|$, we have to show that there cannot exist a function $f : A \to B$ that's both injective and surjective.

Putting everything together, we get the following: $|A| < |B|$ iff

- There is an injection $f : A \to B$.
- There is no bijection $f : A \to B$.

Typically, when trying to prove that $|A| < |B|$, the easy part is finding an injection $f : A \to B$, and the hard part is showing that there are no bijections $f : A \to B$. To give an example about why this is challenging, think back to $\mathbb{N}$ and $\mathbb{Z}$. Until we actually found the bijection between these two sets, it was not at all obvious that $|\mathbb{N}| = |\mathbb{Z}|$. Lots of reasonable-looking functions from $\mathbb{N}$ to $\mathbb{Z}$ aren't bijections. However, the fact that there is even a single function from $\mathbb{N}$ to $\mathbb{Z}$ that is a bijection means that the sets must have the same cardinality. Trying to prove that two sets don't have the same cardinality can be extremely difficult because the proof has to rule out all functions as possible bijections between the two sets, including bizarre piecewise

functions and functions that are so strange-looking that it would be impossible to write them down in finite space.

In this section, we will introduce the technique of *diagonalization* as a tool for showing that two sets do not have the same cardinality as one another. We'll start off with the example of proving $|\mathbb{N}| < |\mathbb{R}|$, then will use a similar technique to write a formal, rigorous mathematical proof of Cantor's Theorem, the result we saw in Chapter 1 that states $|S| < |\wp(S)|$ for all sets $S$. Later on, we will use diagonalization as a technique in other proofs, such as finding concrete examples of problems that can't be solved by computers or exploring why all interesting programming languages make it possible to have infinite loops.

## 6.5.1 $|\mathbb{N}| < |\mathbb{R}|$

We've seen many infinite sets so far that appear to have completely different sizes but which end up having exactly the same size. For example, $\mathbb{N}$ and $\mathbb{Z}$ have the same cardinality as one another, even though it seems like there's twice as many integers as natural numbers, and $\mathbb{N}$ and $\mathbb{N}^2$ have the same cardinality, even though it just "feels" like there are infinitely many copies of $\mathbb{N}$ inside of $\mathbb{N}^2$. However, that doesn't mean that all infinite sets have the same cardinality as one another; you saw a few glimpses of this in Chapter One.

One of the most famous pairs of sets known to have different cardinalities is the set of natural numbers $\mathbb{N}$ and the set of all real numbers $\mathbb{R}$. The real numbers encompass all natural numbers, plus all integers, all rational numbers, and all irrational numbers. There are infinitely many real numbers between 0 and 1, and more generally between any pair of natural numbers. That on its own doesn't immediately mean that there should be more real numbers than natural numbers, though. We've seen other oddities when it comes to infinite cardinalities, so we shouldn't let that influence our thought process. Yet indeed $|\mathbb{N}| < |\mathbb{R}|$, and we're going to prove exactly why this is.

To prove that $|\mathbb{N}| < |\mathbb{R}|$, we need to prove two results. First, we need to prove that $|\mathbb{N}| \leq |\mathbb{R}|$; second, we have to prove $|\mathbb{N}| \neq |\mathbb{R}|$. Earlier on, we briefly sketched a proof that $|\mathbb{N}| \leq |\mathbb{R}|$ by noting that the function $f : \mathbb{N} \to \mathbb{R}$ defined as $f(n) = n$ is an injection from $\mathbb{N}$ to $\mathbb{R}$. However, we haven't yet tried to prove that $|\mathbb{N}| \neq |\mathbb{R}|$. The entire remainder of this section will be dedicated to this result.

Our proof that $|\mathbb{N}| \neq |\mathbb{R}|$ will proceed by contradiction. Let's suppose, hypothetically speaking, that indeed $|\mathbb{N}| = |\mathbb{R}|$. That means that there is some bijection $f : \mathbb{N} \to \mathbb{R}$. This bijection tells us something very interesting about the real numbers: if indeed $\mathbb{R}$ is countably infinite, then we should be able to "count off" all the real numbers by looking at the sequence $f(0)$, $f(1)$, $f(2)$, $f(3)$, …, $f(n)$, … . This list contains every real number exactly once, since $f$ is injective and surjective. To make the notation a bit easier, let's define a sequence $r_0$, $r_1$, $r_2$, …, $r_n$, … of real numbers by setting $r_k = f(k)$. This means that $r_0$, $r_1$, $r_2$, … and $f(0)$, $f(1)$, $f(2)$, … are the same sequences, just with slightly different notation.

The key step in the proof that $|\mathbb{N}| \neq |\mathbb{R}|$ is showing that even if we thought we've counted off all the real numbers, there has to be at least one real number $d$ that isn't anywhere in this sequence. If we can find even one real number $d$ such that $d$ doesn't appear in the sequence $r_0$, $r_1$, $r_2$, …, then we've contradicted the fact that every real number appears somewhere in this sequence. That contradiction in turn will mean that our initial assumption must have been wrong, from which we can conclude $|\mathbb{N}| \neq |\mathbb{R}|$.

We can now state our objective. Given a sequence $r_0$, $r_1$, $r_2$, …, of real numbers, we want to find a real number $d$ such that

$$\text{For any } n \in \mathbb{N}, r_n \neq d.$$

Intuitively, this says that the real number $d$ has to be different from all of the real numbers in the sequence $r_0$, $r_1$, $r_2$, … . It doesn't matter *how* $d$ is different from them; as long as $d$ is never equal to any of the terms in the sequence, we know that we have found a real number that we didn't list.

Our objective, as stated above, is to find a real number $d$ such that for any $n \in \mathbb{N}$, $r_n \neq d$. Although we're writing this as one single constraint, you can think of this as *infinitely many* smaller constraints:

$$r_0 \neq d$$

$$r_1 \neq d$$

$$r_2 \neq d$$

$$r_3 \neq d$$

$$\ldots$$

This gives us a different perspective on how to approach finding our choice of $d$. Rather than starting with the broader claim "$d$ must not appear anywhere in the series," we can think of trying to solve the problem of finding some choice of $d$ that is different from all the terms in the series. This is just a rephrasing of our original goal, but it might make it a bit clearer what we're trying to do.

So how exactly do we find such a $d$? The key insight necessary for this proof is that we can build a "Frankenstein" real number that is built out of tiny "pieces" of real numbers. The important property we will use in the course of building this "Frankenreal" is that each piece is chosen such that the real number $d$ will be different from some particular term in the sequence $r_0$, $r_1$, $r_2$, …, . Specifically, one piece of $d$ will be picked so that $d \neq r_0$, another piece will be picked so that $d \neq r_1$, another so that $d \neq r_2$, etc. That way, there can't be some natural number $n$ such that $d = r_n$, since there is some piece of $d$ specifically built to be different from $r_n$.

All that's left to do now is figure out a way to construct this Frankenreal. This is where it helps to start looking at the structure of real numbers. The major insight we need to have here is that every real number has an infinite decimal representation. For example:

$$2 = 2.000000000000000...$$

$$^1/_7 = 0.142857142857142...$$

$$\pi = 3.141592653589793...$$

$$e = 2.718281828459045...$$

These infinite representations might repeat the same pattern forever (as in the case of 2 or $^1/_7$), or they might have no repeated pattern (as is the case for $\pi$ or $e$). We don't really care about this. All that matters is that we can write out the real numbers as infinite sequences of natural numbers.

Let's introduce some new notation to make it easier to talk about the infinite decimal representations. Specifically, for any real number $r$, let's say that $r[0]$ is the integer part of the real number. For example, $2[0] = 2$, $^1/_7[0] = 0$, $\pi[0] = 3$, $-15.122[0] = -15$, etc. We'll then say that $r[n]$, for $n > 0$, is the $n$th decimal digit of the real number $r$. For example, $\pi[1] = 1$, $\pi[2] = 4$, $\pi[3] = 1$, $\pi[4] = 5$, etc.

Once we have this notation, we can use a cute trick in order to build a Frankenreal that is different from every term of the sequence $r_0$, $r_1$, $r_2$, … . Every real number can be written with an infinite decimal representation, so one way that we could build this Frankenreal $d$ would be to define the values of $d[0]$, $d[1]$, $d[2]$, … etc. on to infinity. That in turn will define the real number $d$.

So how do we build $d$ so that $d \neq r_0$, $d \neq r_1$, $d \neq r_2$, etc.? Here's the key idea. We'll construct this number $d$ such that $d[0] \neq r_0[0]$, and $d[1] \neq r_1[1]$, and $d[2] \neq r_2[2]$, etc. More generally, the real number $d$ will be constructed such that $d[n] \neq r_n[n]$ for any choice of $n$. In this way, we end up with a real number $d$ that can't be equal to any of the numbers in the sequence $r_0$, $r_1$, $r_2$, …, because it was specifically constructed to be different from each of the terms.

All that's left to do now is choosing some way to force $d[n] \neq r_n[n]$ for any choice of $n$. There are many ways that we can do this, and here's one simple option. Define the real number $d$ as follows:

$$d[n] = \begin{cases} 1 & \text{if } r_n[n] = 0 \\ 0 & \text{otherwise} \end{cases}$$

In other words, we'll make $d[n] = 0$ if $r_n[n] \neq 0$, and will make $d[n] \neq 0$ if $r_n[n] = 0$. In other words, if the $n$th digit of the $n$th number is 0, we make the $n$th digit of $d$ nonzero. If the $n$th digit of the $n$th number is nonzero, we make the $n$th digit of $d$ zero. That guarantees us that $d[n] \neq r_n[n]$ for any choice of $n$. It's a pretty mischievous trick, but it gets the job done beautifully. We now have a real number that cannot be equal to any of the terms in the sequence, guaranteeing us that the sequence can't contain all the real numbers. All that's left to do now is to formalize the reasoning in a proof.

To see an example of this construction in action, suppose that we have the following proposed bijection between $\mathbb{N}$ and $\mathbb{R}$:

$r_0 =$ 3.1415926535...

$r_1 =$ 2.7182818284...

$r_2 =$ 1.0000000000...

$r_3 =$ 1.0203040506...

$r_4 =$ -4.1111111111...

...

In this case, we would build our Frankenreal $r$ as follows. Since $r_0[0] \neq 0$, we make $d[0] = 0$. Since $r_1[1] \neq 0$, we make $d[1] = 0$. Since $r_2[2] = 0$, we make $d[2] = 1$. Since $r_3[3] = 0$, we make $d[3] = 1$. Since $r_4[4] = 1$, we make $d[4] = 0$. As a result, the first digits of $d$ will be 0.0110... .

If you look at the above picture of how we built the number $d$, you'll see that we didn't need to look at all of the digits of each of the numbers $r_0$, $r_1$, $r_2$, … . Instead, we just looked at the 0th digit of $r_0$, the 1st digit of $r_1$, the second digit of $r_2$, etc. If we highlight these numbers, we get the following:

$r_0 =$ **3**.1 4 1 5 9 2 6 5 3 5...

$r_1 =$ 2.**7** 1 8 2 8 1 8 2 8 4...

$r_2 =$ 1.0 **0** 0 0 0 0 0 0 0 0...

$r_3 =$ 1.0 2 **0** 3 0 4 0 5 0 6...

$r_4 =$ -4.1 1 1 **1** 1 1 1 1 1 1...

Notice that this sequence of numbers goes down the diagonal of this sequence. This is why this type of proof is called "diagonalization;" one way to think of the proof is by taking this diagonal, then building a number that is different from every term in it.

Before going into the formal proof, we need one quick clarification. Note that this logic works in response to a proposed one-to-one correspondence between $\mathbb{N}$ and $\mathbb{R}$. What we've shown is that no matter how you try to pick $r_0$, $r_1$, $r_2$, …, there has to be at least one real number $d$ that isn't in the list. However, this real number $d$ isn't going to be the same every time. Specifically, if you choose a difference sequence $r_0$, $r_1$, $r_2$, …, then $d$ might come out to a different number. In this way, we can show that *no possible sequence of real numbers contains all the real numbers*. No matter how you choose the sequence, there will always be some number that's missing. Consequently, we can conclude that $|\mathbb{N}| \neq |\mathbb{R}|$.

The formal version of the proof will work along the lines we just stated. We'll start off by assuming there's a bijection $f : \mathbb{N} \to \mathbb{R}$. From there, we'll define a sequence $r_0$, $r_1$, $r_2$, … of all the real numbers. We'll then define $d$ in terms of the series using the above trick. Finally, we'll conclude that it's impossible for $d$ to appear in the series, giving us our contradiction.

***Theorem:*** $|\mathbb{N}| \neq |\mathbb{R}|$.

***Proof:*** By contradiction; assume $|\mathbb{N}| = |\mathbb{R}|$. This means there is some bijection $f : \mathbb{N} \to \mathbb{R}$. For notational simplicity, define $r_n = f(n)$ for all $n \in \mathbb{N}$.

Next, for any real number $r$, define $r[0]$ to be the integer part of $r$ and $r[n]$, for any $n > 0$, to be the $n$th decimal digit of $r$. Now, consider the real number $d$ defined as follows:

$$d[n] = \begin{cases} 1 & \text{if } r_n[n] = 0 \\ 0 & \text{otherwise} \end{cases}$$

Since $f$ is a bijection, it is a surjection. Therefore, there must be some $n \in \mathbb{N}$ for which $f(n) = d$. This in turn means there is some $n \in \mathbb{N}$ for which $r_n = d$. Now, consider $r_n[n]$ and how it relates to $d[n]$. We consider two cases:

*Case 1:* $r_n[n] = 0$. Then by construction, $d[n] = 1$, so $d[n] \neq r_n[n]$.
*Case 2:* $r_n[n] \neq 0$. Then by construction, $d[n] = 0$, so $d[n] \neq r_n[n]$.

In both cases we find $d[n] \neq r_n[n]$. But this is impossible, since $d = r_n$. We have reached a contradiction, so our assumption must have been wrong. Thus $|\mathbb{N}| \neq |\mathbb{R}|$. ■

This proof was a landmark in the history of mathematics. When presented, it gave a clear, simple, and cogent argument that showed that not all infinite sets have the same size, and in fact the infinite number of real numbers is not the same as the infinite number of natural numbers.

Previously, we talked about countably infinite sets, sets whose cardinality is equal to that of the natural numbers. We've just shown that the real numbers have a cardinality strictly greater than that of the natural numbers. To describe infinite sets like $\mathbb{R}$ that are bigger than the natural numbers, we'll introduce a new piece of terminology.

A set $S$ is called ***uncountable*** iff $|\mathbb{N}| < |S|$.

We have just proven that $\mathbb{R}$ is uncountable, but it is by no means the only uncountable set. Many other infinite sets are uncountable, and in the next section we'll see how to construct infinitely many of them.

## 6.5.2 Cantor's Theorem

In Chapter One, we briefly explored *Cantor's Theorem*, the result that every set is strictly smaller than its power set. Formally, Cantor's Theorem states that $|S| < |\wp(S)|$ for any set $S$. Our initial "proof" of Cantor's theorem was not at all rigorous; at the time that we set out to prove it, we hadn't actually even seen how to

write a proof yet! To conclude our treatment of cardinalities, let's revisit this proof and see exactly how it works and what makes it tick.

To show for every set $S$ that $|S| < |\wp(S)|$, we need to prove two things. First, we need to show that $S$ is no bigger than $\wp(S)$ by proving $|S| \leq |\wp(S)|$. This isn't too challenging, and it's left as an exercise at the end of the chapter. Next, we need to show $|S| \neq |\wp(S)|$, which is decidedly more involved.

In Chapter One, we saw a sketch of a diagonal argument that we could use to prove that $|S| \neq |\wp(S)|$ for every set $S$. We drew out a two-dimensional grid where each row corresponded to the set paired with some element of $S$, then flipped the elements along the diagonal and concluded that we had a set that couldn't appear in the table. But where did this idea come from? Could we have figured this out without having to draw out a 2D grid and take the diagonal? Using the same line of reasoning we tried out in the case of $\mathbb{N}$ and $\mathbb{R}$, it's absolutely possible to arrive at this conclusion independently. This section explores exactly how we might do that.

As with before, the intuition is the following. Let's suppose, for the sake of contradiction, that there is some set $S$ where $|S| = |\wp(S)|$. That means there's some bijection $f : S \to \wp(S)$. Since $f$ is a bijection, $f$ is surjective, so for every set $T \in \wp(S)$, there is some $x \in S$ for which $f(x) = S$. If we can find even one set $D \in \wp(S)$ such that no element of $S$ maps to $D$, then we'll have shown $f$ isn't a bijection, contradicting our initial assumption. Therefore, in our proof, we'll try to find a set $D$ for which no element of $S$ maps to $D$. Formalizing this mathematically, we want to find a set $D \in \wp(S)$ such that

$$\text{For any } x \in S, \text{ we have } f(x) \neq D.$$

(A quick note on the notation: $f(x)$ means "the set produced when you apply function $f$ to the object $x$. Saying $f(x) \neq D$ means "$x$ doesn't map to $D$ when you apply function $f$ to it," rather than "the function $f$ is not defined by the rule $f(x) = D$")

The question now, of course, is how we're supposed to come up with $D$. As before, our goal will be to construct some sort of "Frankenset" $D$ that is built out of lots of smaller pieces, each of which prevents some specific $x$ from satisfying $f(x) = D$.

In the previous proof that $|\mathbb{N}| < |\mathbb{R}|$, we constructed our "Frankenreal" by building a real number $d$ where the $n$th digit of $d$ conflicted with the $n$th digit of $r_n$. Abstracting away from the particular details of how we built $d$, we can think of this construction as follows: we built $d$ from different pieces, one for each possible value of $n$, so that the $n$th piece of $d$ conflicts with the $n$th piece of the $n$th element in the sequence. That guarantees that $d$ can't be equal to any of the elements in the sequence.

There are two key properties at play that made this previous proof work. First, when pairing up natural numbers with real numbers, it was possible to talk about the "$n$th piece" of some particular real number. We made this work by looking at the decimal representation of the real numbers; the $n$th "piece" of a real number is its $n$th decimal digit. Second, it was possible to define some real number by specifying each of its pieces individually. When defining the real number $d$, we ended up constructing $d$ by specifying each of its digits individually.

Let's see if we can make this argument work on elements of a set $S$ and elements of its power set $\wp(S)$. Specifically, we will want to have some way of talking about the "$x$th piece" of some set $X$, where $x \in S$ and $X \in \wp(S)$. Once we can come up with a way of doing this, we can try to define our set $D$ such that for any $x \in S$, the "$x$th piece" of $D$ disagrees with the "$x$th piece" of $f(x)$. If we can do this, we can guarantee that for any $x \in S$, that $f(x) \neq D$, since the "$x$th piece" of $f(x)$ isn't the same as the "$x$th piece" of $D$.

The key creative insight we need to come up with is how we define the "$x$th piece" of a set $X \in \wp(S)$. For starters, notice that any $X \in \wp(S)$ has to be a subset of $S$. Consequently, for any $x \in S$, either $x \in X$ or $x \notin X$. Let's therefore define the "$x$th piece" of a set $X$ to be whether or not $x \in X$. This step is the major creative

step in the proof – if it seems non-obvious, that's okay. Lots of diagonalization proofs have a non-obvious step like this one in them, but with practice you'll get quite good at building an intuition.

Given this insight as to how we can define the "$x$th piece" of a set $X$, we can formalize how we're going to construct our Frankenset $D$. We want to build $D$ such that the "$x$th piece" of $D$ disagrees with the "$x$th piece" of $f(x)$. Using the fact that the "$x$th piece" of a set is whether or not it contains the element $x$, this means that we want to define our set $D$ such that if $x \in f(x)$, then $x \notin D$, and if $x \notin f(x)$, then $x \in D$.*

Equivalently, this means that

$$\text{For any } x \in S\text{: } x \in D \text{ iff } x \notin f(x)$$

Since a set is uniquely defined by what elements it contains, this means that the set $D$ is the set

$$D = \{\ x \in S \mid x \notin f(x)\ \}$$

*Et voilà*. We have constructed a set $D$ that can't possibly be equal to $f(x)$ for any $x$, since it won't contain $x$ if $f(x)$ contains $x$ and will contain $x$ if $f(x)$ doesn't contain $x$. All that's left to do now is formalize this as a proof.

***Theorem:*** For all sets $S$, we have $|S| \neq |\wp(S)|$.

***Proof:*** By contradiction; assume there is some set $S$ for which $|S| = |\wp(S)|$. This means that there is a bijection $f : S \to \wp(S)$.

Consider the set $D = \{\ x \in S \mid x \notin f(x)\ \}$. Note that $D \subseteq S$, so $D \in \wp(S)$. Since $f$ is a bijection, it is surjective, so there must be some $x \in S$ such that $f(x) = D$. We now consider two possibilities:

*Case 1:* $x \in D$. Since $D = f(x)$, this means $x \in f(x)$. But then by definition of $D$, we have $x \notin D$, contradicting that $x \in D$.

*Case 2:* $x \notin D$. Since $D = f(x)$, this means $x \notin f(x)$. But then by definition of $D$, we have $x \in D$, contradicting that $x \notin D$.

In either case, we reach a contradiction, so our assumption must have been wrong. Therefore, for every set $S$, we have $|S| \neq |\wp(S)|$. ■

This proof is deceptive in its complexity. It isn't very long, and there aren't that many difficult steps (except for the weird logical dances inside the two cases). However, coming up with this proof in the first place was challenging. This diagonalization proof is substantially more involved than the diagonalization involving real numbers. It wasn't as clear how to use the elements of $S$ to "index" into sets of elements of $S$. We had to recognize that the "$x$th piece" of a set of elements of $S$ could be defined as whether or not the set $S$ contained $x$.

If this proof does not seem at all obvious to you, *don't worry!* This proof rocked the mathematical community when it was first introduced and requires some decidedly non-obvious leaps. Our hope is that you can at

* A quick aside on notation: $x \notin f(x)$ means "$x$ is not an element of the set that it maps to." It doesn't mean "$x$ isn't mapped by the function $f$."

least follow what the proof is doing and that you're comfortable with the intuition behind the proof: choosing $D$ to be different from $f(x)$ for all $x \in S$. The rest is just the mechanics necessary to get everything to work.

A consequence of Cantor's theorem is that there cannot be any "largest" infinite set. Given any set that you'd like, you can always take its power set to produce a set larger than it. For example, we can form sets that are much, much larger than $\mathbb{R}$ by considering the sequence $\wp(\mathbb{R})$, $\wp(\wp(\mathbb{R}))$, $\wp(\wp(\wp(\mathbb{R})))$, etc.

# 6.6 Chapter Summary

- A *function* is a means of associating each element of a one set (called the *domain*) with another set (called the *codomain*). We write $f : A \to B$ if $f$ is a function with domain $A$ and codomain $B$.
- Functions can be defined with a picture, by writing a rule associating one element of the codomain with each element of the domain, or by transforming existing functions (perhaps by composing them, for example).
- An *injection* is a function where no two elements of the domain map to the same element of the codomain.
- A *surjection* is a function where every element of the codomain is covered by at least one element of the domain.
- The *image* of a set $X$ under a function $f$, denoted $f[X]$, is the set of all elements of the codomain mapped to by $f$ by some element of $X$.
- The *preimage* of a set $Y$ under a function $f$, denoted $f^{-1}[Y]$, is the set of all elements in the domain that map by $f$ to some element in $Y$.
- The *range* of a function $f : A \to B$ is $f[A]$.
- A *bijection* is a function that is both injective and surjective.
- The *composition* of $f$ and $g$, denoted $g \circ f$, is the function formed by first applying $f$ to the input, then applying $g$ to the result.
- The composition of two injective functions is injective and the composition of two surjective functions is surjective. Consequently, the composition of two bijections is a bijection.
- If f is a bijection, then $f^{-1}$ is the inverse function of $f$. The inverse of $f$ obeys the rule $f(x) = y$ iff $f^{-1}(y) = x$.
- For any set $A$, the *identity function* over $A$ is the function $\mathrm{id}_A : A \to A$ defined as $\mathrm{id}_A(x) = x$.
- $f$ and $g$ are inverses of one another iff $f \circ g$ and $g \circ f$ are the identity functions over the appropriate domains.
- $A$ has the same cardinality as $B$ iff there is a bijection $f : A \to B$. We denote this as $|A| = |B|$.
- Equality of cardinality is reflexive, symmetric, and transitive.
- A set is called countably infinite iff its cardinality is the cardinality of $\mathbb{N}$. The set of all integers $\mathbb{Z}$ and set of all pairs of naturals $\mathbb{N}^2$ are countably infinite.
- $A$ has cardinality no greater than $B$'s cardinality iff there is an injection $f : A \to B$. We denote this by writing $|A| \leq |B|$. This relation is reflexive, antisymmetric, and transitive.
- $|A| < |B|$ iff $|A| \leq |B|$ and $|A| \neq |B|$.

- Diagonalization is a proof technique for showing, among other things, that two sets cannot have the same cardinality.
- A set is called uncountable iff its cardinality is greater than that of the natural numbers. $\mathbb{R}$ is an uncountable set.
- Cantor's theorem states that $|S| < |\wp(S)|$ for any set $S$.

# 6.7 Chapter Exercises

1. Early in the chapter we came up with two formal definitions for functions, one involving binary relations and one involving subsets of $A \times B$. Using the fact that a binary relation over sets $A$ and $B$ is defined as a subset of $A \times B$, show that these two definitions are in fact completely equivalent to one another.
2. Find a function $f : \mathbb{R} \to \mathbb{R}$ such that $f[\mathbb{N}] = \mathbb{N}$ but $f[\mathbb{Z}] \neq \mathbb{Z}$.
3. Let $A$ and $B$ be nonempty sets. In terms of $|A|$ and $|B|$, what is the maximum possible cardinality of $f[A]$? What is the minimum possible cardinality of $f[A]$? How about $f^{-1}[B]$?
4. Prove or disprove: if $f : A \to B$ is a function and $f^{-1}[B] = A$, then $f$ is surjective.
5. Prove or disprove: if $f : A \to B$ is a function and $f^{-1}[B] = A$, then $f$ is injective.
6. Find an example of a function that is neither injective nor surjective.
7. Let $A$ and $B$ be arbitrary sets where $\leq_B$ is a partial order over $B$. Suppose that we pick an injective function $f : A \to B$. We can then define a relation $\leq_A$ over $A$ as follows: for any $x$, $y \in A$, we have $x \leq_A y$ iff $f(x) \leq_B f(y)$. Prove that $\leq_A$ is a partial order over $A$.
8. Prove that function composition is associative: that is, $h \circ (g \circ f) = (h \circ g) \circ f$.
9. Let $f : A \to B$ be a function. Prove that $\text{id}_B \circ f = f \circ \text{id}_A = f$.
10. Prove that $\text{id}_A$ is its own inverse for any set $A$.
11. Recall from the start of the chapter that any function $f : A \to B$ is actually a binary relation over $A \times B$; specifically, $f(a) = b$ is just shorthand for $afb$. Give a simple description of $\text{id}_A$ as a relation over $A \times A$.
12. Find an example of a function $f : A \to A$ other than $\text{id}_A$ such that $f$ is its own inverse. Functions that are their own inversions are called *involutions*.
13. Show that every invertible function has exactly one inverse. That is, if $f : A \to B$ is invertible and there are two functions $g_1 : B \to A$ and $g_2 : B \to A$ such that $g_1$ is the inverse of $f$ and $g_2$ is the inverse of $f$, then $g_1 = g_2$.
14. Suppose we find functions $f : A \to B$ and $g : B \to A$ where $g \circ f = \text{id}_A$. Does this necessarily mean that $g = f^{-1}$? If so, prove it. If not, find a counterexample.
15. Suppose we find bijections $f : A \to B$ and $g : B \to A$ where $g \circ f = \text{id}_A$. Does this necessarily mean that $g = f^{-1}$? If so, prove it. If not, find a counterexample.
16. In this chapter, we proved that if $f : A \to B$ and $g : B \to C$ are bijections, then $g \circ f$ is also a bijection. Is the converse true? That is, if $g \circ f$ is a bijection, does it necessarily mean that $f$ and $g$ are bijections? If so, prove it. If not, find a counterexample.
17. Suppose $f : A \to B$ and $g : B \to C$ are invertible. Prove that $g \circ f$ is invertible. What is its inverse?

18. Let $<_A$ and $<_B$ be strict total orders over sets $A$ and $B$, respectively. A function $f : A \to B$ is called *monotonically increasing* iff for any $a_0$, $a_1 \in A$ where $a_0 <_A a_1$, we have $f(a_0) <_B f(a_1)$. Prove that if $f$ is surjective and monotonically increasing, then $f$ is a bijection.

19. Prove that if $A$ and $B$ are countably infinite sets, then $A \times B$ is countably infinite.

20. Prove that if $A$ is countably infinite, then $A \times \{0, 1\}$ is countably infinite.

21. Prove that if $A$ and $B$ are countably infinite sets, then $A \cup B$ is countably infinite. *(Hint: Use the Cantor-Bernstein-Schroeder theorem.)*

22. Prove that if $|A| = |C|$ and $|B| = |D|$, that $|A \times B| = |C \times D|$. Remember – to prove this, you need to find a bijection from $A \times B$ to $C \times D$.

23. Using your result from the previous problem, prove that $|\mathbb{N}^k| = |\mathbb{N}|$ for all $k \geq 1$. Recall that $\mathbb{N}^k$ is recursively defined such that $\mathbb{N}^1 = \mathbb{N}$ and $\mathbb{N}^{k+1} = \mathbb{N} \times \mathbb{N}^k$.

24. Prove that if $|A| = |C|$, $|B| = |D|$, $A \cap B = \varnothing$, and $C \cap D = \varnothing$, then $|A \cup B| = |C \cup D|$.

25. Given the intuition about where the Cantor pairing function $\pi$ comes from, give a justification as to why $\pi(a, b) = (a + b)(a + b + 1) / 2 + a$. *(Hint: What diagonal does (a, b) belong to? How many pairs are on earlier diagonals? What position is (a, b) in on its diagonal?)*

26. Prove that the Cantor pairing function is a bijection between $\mathbb{N}^2$ and $\mathbb{N}$. ★

27. Prove or disprove: If $A \subseteq B$, then $|A| \leq |B|$.

28. Prove or disprove: If $A \subset B$, then $|A| < |B|$.

29. The function $f(x) = 0$ is a function $f : \mathbb{R} \to \mathbb{N}$ that is not a surjection. Why doesn't that contradict the fact that $|\mathbb{N}| < |\mathbb{R}|$?

30. Prove that for every set $S$, we have $|S| \leq |\wp(S)|$.

31. One edge case of Cantor's Theorem that's worth exploring is when $S = \varnothing$. In that case, the only function from $S$ to $\wp(S)$ is the empty function. Trace through Cantor's theorem in this case. What is the set $D$? Why does the theorem still hold in this case?

32. Consider the set $\mathbb{N}^{\mathbb{N}} = \{ f \mid f : \mathbb{N} \to \mathbb{N} \}$, the set of all functions from natural numbers to natural numbers. Prove that $|\mathbb{N}| < |\mathbb{N}^{\mathbb{N}}|$.

33. Prove that if $|A| < |B|$, then there exists some $x \in B$ such that $x \notin A$. *(This may seem obvious, but your proof should use the formal definition of $|A| < |B|$ to prove its result.)*

34. One of the major results from modern set theory is that not all collections can be gathered together to form a set. One such example is the "set of all sets."

    Suppose hypothetically that the universal set $\square = \{ x \mid x \text{ exists} \}$ exists. This set contains *everything*: it contains all natural numbers, all sets, all people in the world, all thoughts and hopes and dreams, and even itself! It turns out, however, that $\square$'s existence leads to a contradiction.

    Using Cantor's theorem and the result from the previous problem, prove that $\square$ does not exist.

# Alphabetical Index