CS 103: Mathematical Foundations of Computing
Problem Set 3

July 12, 2019

Checkpoint due Monday, July 15th at 3:00PM.
Remaining problems due Friday, July 19th at 3:00PM.

This third problem set explores binary relations, functions, and their properties. We've chosen these problems to help you learn how to reason about these structures, how to write proofs using formal mathematical definitions, and why all this matters in practice.

Before beginning this problem set, we ***strongly*** recommend reading over the following handouts:

- The "Guide to Proofs on Discrete Structures," which explores how to write proofs when definitions are rigorously specified in first-order logic. This handout contains both general guiding principles to follow and some sample proof templates that you're welcome to use here.
- The "Discrete Structures Proofwriting Checklist," which contains some specific items to look for when proofreading your work.

We recommend that you take a look at the proofs from this week's lectures to get a sense of what this looks like. The proofs on cyclic relations from Wednesday, or the proofs about injectivity and surjectivity from Friday, are great examples of the style we're looking for.

Good luck, and have fun!

These three checkpoint problems on this problem set are due on Monday at 3:00PM.

Checkpoint Problem One: The Coprime Relation

If a and b are integers, we say that a ***divides*** b if there is an integer q such that $b = aq$. We'll say that a and b are ***coprime*** if there are no integers that divide both a and b besides ± 1. As an example, 7 and 2 are coprime (± 1 are the only integers that divide both 7 and 2) but 6 and 2 are not (the integer 2 divides both 6 and 2).

We define a relation $\perp$ (note we are reusing our First-Order Logic symbol for "false," but that meaning of this symbol is unrelated to this new meaning we are giving it just for this problem) over $\mathbb{N}$ as:

$$x \perp y \text{ if } x \text{ and } y \text{ are coprime}$$

Note that this specifically is a property that holds between *pairs* of numbers. For example, you can say "8 is coprime with 15" or "10 is coprime with 21," but not that "45 is coprime." This follows from the definition, which only defines what coprime means in terms of pairs of integers.

For each of the properties below, determine whether or not the $\perp$ relation has that property. If it does, give a brief (one-sentence) justification why. For each property it doesn't have, write a short disproof.

i. Is the relation $\perp$ reflexive?

ii. Is the relation $\perp$ irreflexive?

iii. Is the relation $\perp$ symmetric?

iv. Is the relation $\perp$ asymmetric?

v. Is the relation $\perp$ transitive?

As a total aside, using the notation $\perp$ for expressing coprimality comes from the text *Concrete Mathematics* by Donald Knuth. The rationale given is that $\perp$ represents a pair of perpendicular lines and, "Like perpendicular lines don't have a common direction, perpendicular numbers don't have common factors."

Checkpoint Problem Two: Binary Relations IRL

The first part of this problem revolves around a mathematical construct called ***homogeneous coordinates*** that shows up in computer graphics. If you take CS148, you'll get to see how they're used to quickly determine where to display three-dimensional objects on screen.

Let $\mathbb{R}^2$ denote the set of all ordered pairs of real numbers. For example $(137, 42) \in \mathbb{R}^2$, $(\pi, e) \in \mathbb{R}^2$, and $(-2.71, 103) \in \mathbb{R}^2$. Two ordered pairs are equal if and only if each of their components are equal. That is, we have $(a, b) = (c, d)$ if and only if $a = c$ and $b = d$.

Consider the relation E defined over $\mathbb{R}^2$ as follows:

$$(x_1, y_1) E (x_2, y_2) \text{ if } \exists k \in \mathbb{R} . (k \neq 0 \wedge (kx_1, ky_1) = (x_2, y_2)).$$

For example, $(3, 4) E (6, 8)$ because $(2 \cdot 3, 2 \cdot 4) = (6, 8)$.

i. Prove that E is an equivalence relation over $\mathbb{R}^2$.

Remember that the "if" in the definition of the relation E means "is defined as" and isn't an implication.

Follow the advice of the Guide to Proofs on Discrete Structures and the Discrete Structures Proofwriting Checklist: don't use quantifiers or connectives in your written proof. You may want to start by taking the first-order statement in the definition here and determining what it says in plain English.

Once you've written a draft of your proof of this result, take a few minutes to read over them and apply both the standard Proofwriting Checklist (the one you've used on the first two problem sets) and the new Discrete Structures Proofwriting Checklist. Here are a few specific things to keep an eye on:

- *The key terms in binary relations (reflexivity, symmetry, transitivity, irreflexivity, and asymmetry) are defined in first-order logic and proofs of those properties depend on those first-order definitions. However, as a reminder, you should* ***not*** *include first-order logic notation (quantifiers, connectives, etc.) anywhere in your proofs. Look at the proofs from the Guide to Binary Relations and last week's lectures for some examples of what we're expecting.*
- *Make sure that you've set all of your proofs up properly. For example, what should a proof that a relation is symmetric look like? What should you assume, and what you should prove? Does your proof match this pattern?*
- *You don't need to – and in fact, shouldn't – repeat the definitions of the E or R relations in your proofs. You can assume that the reader knows how they're defined.*

Checkpoint Problem Three: Redefining Equivalence Relations?

Below is a purported proof that every relation that is both symmetric and transitive is also reflexive.

(False!) Theorem: If R is a symmetric and transitive binary relation over a set A, then R is also reflexive.
(Incorrect!) Proof: Let R be an arbitrary binary relation over a set A such that R is both symmetric and transitive. We need to show that R is reflexive. To do so, consider an arbitrary $x, y \in A$ where xRy. Since R is symmetric and xRy, we know that yRx. Then, since R is transitive, from xRy and yRx we learn that xRx is true. Therefore, R is reflexive, as required.

This proof, unfortunately, is incorrect.

i. Draw a picture of a binary relation that is symmetric and transitive but not an equivalence relation. Briefly justify your answer, though no formal proof is required. This shows that the "theorem" here isn't even true to begin with.

What has to happen for a binary relation to not be an equivalence relation? What, specifically, has to happen if you know that that relation is symmetric and transitive?

ii. What's wrong with this proof? Justify your answer. Be as specific as possible.

We've given this proof as an example because the error it contains is one that we see a lot of people make consistently throughout the problem set. That often leads to folks having errors in every single one of the proofs they submit. Read over the Guide to Proofs on Discrete Structures and Discrete Structures Proofwriting Checklist and make sure you're confident that you see what the issue is. If you aren't sure, feel free to ask on Piazza or to stop by office hours. Regardless of what you write, be sure to read the checkpoint solutions as soon as they come out! It would be a shame if you missed the issue here and then made this exact mistake later on in this problem set.

These remaining problems are due on Friday at 3:00PM.

1 So What Exactly is a Binary Relation, Anyway?

When we described binary relations in lecture, we gave an *operational definition* of a binary relation by saying what binary relations do, but we never actually said *what binary relations are.*

Let's begin with a new definition. Given a set A, the Cartesian square of A, denoted A^2, is the set of all ordered pairs that can be formed from elements of A. Formally speaking, we define A^2 as

$$A^2 = \{(a_1, a_2) \mid a_1, a_2 \in A\}$$

For example, if $A = \{1, 3, 7\}$, then

$$A^2 = \{(1,1), (1,3), (1,7), (3,1), (3,3), (3,7), (7,1), (7,3), (7,7)\}.$$

We can use the Cartesian square of a set to rigorously define binary relations. Formally speaking, a binary relation R over a set A is a set $R \subseteq A^2$. The ordered pairs in R correspond to pairs of elements where the relation holds. For example, the $<$ relation over the set $\mathbb{N}$ would formally be defined as

$$< = \{(0,1), (0,2), (0,3), \ldots, (1,2), (1,3), (1,4), \ldots, (2,3), (2,4), (2,5), \ldots\}$$

When we've talked about relations, we've used the notation xRy to denote that x relates to y by relation R. Formally speaking, the notation xRy is just a shorthand for $(x, y) \in R$. This means that if you happen to stumble across a random set of pairs of things, you could interpret it as a binary relation.

Visit the CS103 website and download the starter project files for Problem Set Three. In BinaryRelations.h, there's a definition of a Relation type that represents a binary relation expressed as a set of ordered pairs. We'd like you to write some C++ code in BinaryRelations.cpp to analyze those relations.

i. Implement a function

```
bool isReflexive(Relation R);
```

that takes as input a binary relation R and returns whether R is reflexive.

You already have practice translating first-order logic statements into code. Aside from the new syntax for working with pairs, this is no different.

ii. Implement a function

```
bool isSymmetric(Relation R);
```

that takes as input a binary relation R and returns whether R is symmetric.

iii. Implement a function

```
bool isTransitive(Relation R);
```

that takes as input a binary relation R and returns whether R is transitive.

iv. Implement a function

```
bool isIrreflexive(Relation R);
```

that takes as input a binary relation R and returns whether R is irreflexive.

v. Implement a function

```
bool isAsymmetric(Relation R);
```

that takes as input a binary relation R and returns whether R is asymmetric.

vi. Implement a function

```
bool isEquivalenceRelation(Relation R);
```

that takes as input a binary relation R and returns whether R is an equivalence relation.

vii. Implement a function

```
bool isStrictOrder(Relation R);
```

that takes as input a binary relation R and returns whether R is a strict order.

viii. Implement a function

```
std::vector<std::set<int>> equivalenceClassesOf(Relation R);
```

that takes as input a binary relation R, which you can assume is an equivalence relation, and returns a list of all equivalence classes of R. The return type is a `std::vector` (essentially, a list) of sets of integers; there's information in the starter files about how to work with `std::vector`. You should return exactly one copy of each equivalence class.

Our starter code will automatically call your `isEquivalenceRelation` predicate function and, for each relation you flag as an equivalence relation, will call `equivalenceClassesOf` to color code the different equivalence classes of each equivalence relation.

ix. Edit the file PartA.relation in the res/ directory to define a binary relation that is neither symmetric nor asymmetric. This shows that the terms "symmetric" and "asymmetric" are not negations of one another. There's a description of the expected file format in this file.

x. Edit the file PartB.relation in the res/ directory to define a binary relation that is both symmetric and asymmetric. *(Yes, this is possible!)*

xi. Edit the file PartC.relation in the res/ directory to define a binary relation that is both reflexive and irreflexive. *(Yes, this is possible!)*

You're welcome to submit your answers as many times as you'd like; our autograder will provide feedback on how you're doing.

2 Fun with Equivalence Relations

Throughout mathematics, it's helpful to switch back and forth between the rigorous definitions we've developed for various terms and the intuitions that got us to those terms in the first place.

i. Let $A = \{1, 2, 3\}$. How many different equivalence relations are there over the set A? Explain how you arrived at your answer and why you know there aren't any more. No formal proof is required.

Two binary S and T over the same set are the same if, for any x and y, xSy holds if and only if xTy holds. For example, The relation xSy over $\mathbb{R}$ defined as $|x| = |y|$ is exactly the same relation as xTy over $\mathbb{R}$ defined as $x^2 = y^2$, since xSy and xTy are always either both true or both false.

Let's suppose you have an equivalence relation R over a set A. The Fundamental Theorem of Equivalence Relations says that R partitions the elements of A into different equivalence classes. If you pick exactly one element from each equivalence class and gather the result into a set, you get what's called a ***system of representatives*** for R. Stated differently, a system of representatives for an equivalence relation R over a set A is a set $X \subseteq A$ such that X contains exactly one element of each equivalence class of R.

ii. Consider the equivalence relation $\equiv_5$ over the set $\mathbb{Z}$. Give two different systems of representatives for $\equiv_5$, and briefly justify your answer.

iii. Consider the equivalence relation $=$ over the set $\mathbb{Z}$. List all the systems of representatives for that equivalence relation. Briefly justify your answer; in particular, explain how you know you have all possible systems of representatives.

Systems of representatives are a powerful idea, and we'll return to them later in the quarter.

3 Redefining Strict Orders

In Wednesday's lecture, we defined strict orders as binary relations that are irreflexive, asymmetric, and transitive. Interestingly, it turns out that we could have left asymmetry out of our definition and just gone with irreflexivity and transitivity.

i. Prove that a binary relation R over a set A is a strict order if and only if the relation R is irreflexive and transitive.

Once you've written up your proof, take a minute to critique your proof by applying the Proofwriting Checklist and the Discrete Structures Proofwriting Checklist. Also, think about the following:

- *The statement you need to prove here is a biconditional. How do you prove a biconditional statement? What should you be assuming in each part of the proof? What do you need to prove?*
- *At some point, you'll need to prove that R is asymmetric under some set of assumptions. What does a proof of asymmetry look like? What should you be assuming? What should you be proving?*
- *The proof you will be doing here will involve reasoning about arbitrarily-chosen binary relations where you have no idea what the relation is or what set it's over and only know some properties that it must have (say, that it's irreflexive). In cases like these, be very careful not to make claims about how the relation works that don't immediately follow from your assumptions. After all, your relation could be something like the $<$ relation over $\mathbb{N}$, or the $\subsetneq$ relation over $\wp(\mathbb{R})$, or the "runs strictly faster than" relation over people.*

Going forward, it turns out that one of the easiest ways to prove that a relation is a strict order is to prove that it's irreflexive and transitive.

While we could have left asymmetry out of the definition of a strict order, we could *not* have left out transitivity.

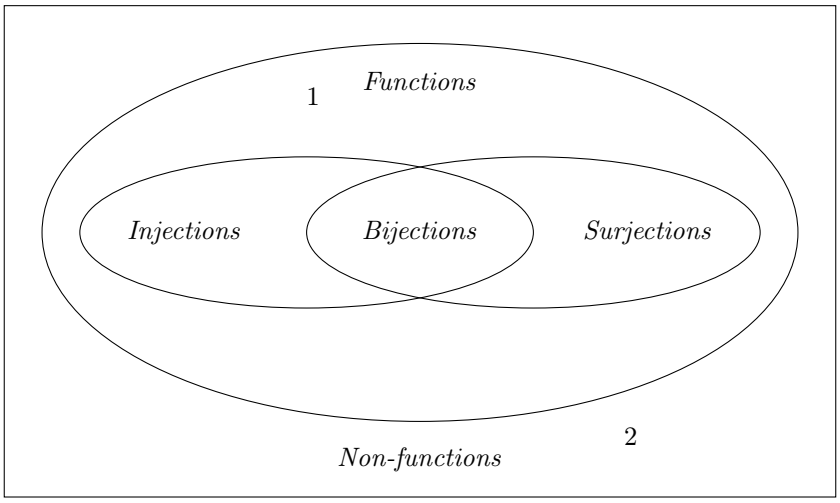
ii. Edit the file PartD.relation in the res/ directory of the starter files to define a binary relation that is irreflexive and asymmetric, but not transitive. This shows that a relation that's asymmetric and irreflexive isn't necessarily a strict order.

It turns out that we *also* could have equivalently defined strict orders to be binary relations that are asymmetric and transitive. We're not going to ask you to prove this, but it's a great exercise if you want to give it a try!

A note going forward: this result about strict orders is so fundamental that we'll commonly use it without citation. Feel free, in your proofs later in this course, to prove that a binary relation is a strict order by just proving that it's irreflexive and transitive. No further elaboration is needed!

4 Properties of Functions

Consider the following Venn diagram:

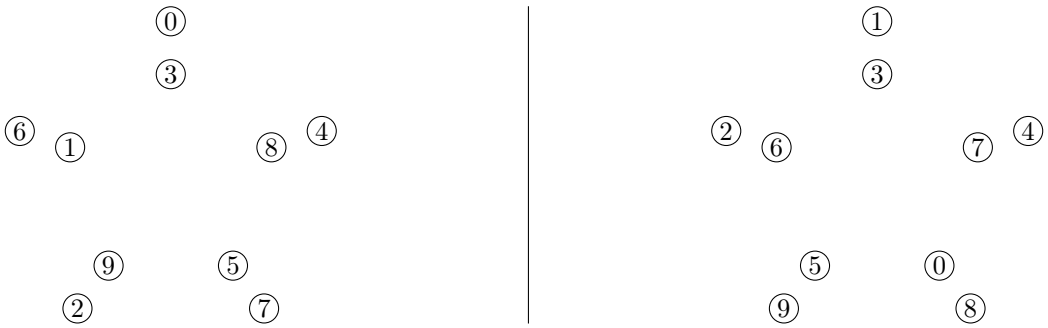


Below is a list of purported functions. For each of those purported functions, determine where in this Venn diagram that object goes. No justification is necessary. To submit your answers, edit the file FunctionsVennDiagram.h in the src/ directory of the starter files for this problem set. For simplicity, we've shown you where functions 1 and 2 go in this Venn diagram.

1. $f : \mathbb{N} \to \mathbb{N}$ defined as $f(n) = 137$
2. $f : \mathbb{N} \to \mathbb{N}$ defined as $f(n) = -137$
3. $f : \mathbb{N} \to \mathbb{N}$ defined as $f(n) = n^2$
4. $f : \mathbb{Z} \to \mathbb{N}$ defined as $f(n) = n^2$
5. $f : \mathbb{N} \to \mathbb{Z}$ defined as $f(n) = n^2$
6. $f : \mathbb{Z} \to \mathbb{Z}$ defined as $f(n) = n^2$
7. $f : \mathbb{R} \to \mathbb{N}$ defined as $f(n) = n^2$
8. $f : \mathbb{N} \to \mathbb{R}$ defined as $f(n) = n^2$
9. $f : \mathbb{N} \to \mathbb{N}$ defined as $f(n) = \sqrt{n}$. ($\sqrt{n}$ is the ***principle square root*** of n, the nonnegative one.)
10. $f : \mathbb{R} \to \mathbb{R}$ defined as $f(n) = \sqrt{n}$.
11. $f : \mathbb{R} \to \{x \in \mathbb{R} \mid x \geq 0\}$ defined as $f(n) = \sqrt{n}$.
12. $f : \{x \in \mathbb{R} \mid x \geq 0\} \to \{x \in \mathbb{R} \mid x \geq 0\}$ defined as $f(n) = \sqrt{n}$.
13. $f : \{x \in \mathbb{R} \mid x \geq 0\} \to \mathbb{R}$ defined as $f(n) = \sqrt{n}$.
14. $f : \mathbb{N} \to \wp(\mathbb{N})$, where f is some injective function.
15. $f : \{0, 1, 2\} \to \{3, 4\}$, where f is some surjective function.
16. $f : \{\textit{breakfast}, \textit{lunch}, \textit{dinner}\} \to \{\textit{shakshuka}, \textit{soondubu}, \textit{maafe}\}$, where f is some injection.

5 Permutation Dances

There's a dance in which each dancer has an assigned position. In the first dance, the dancers begin in the positions indicated on the left (it's a top-down view, and we've numbered the dancers 0, 1, ..., 9). In the second dance, some dancers have moved to new starting positions, and the overall arrangement is what's shown on the right.



How can we model how the dancers' positions changed from the first dance to the second? For now, focus on Dancer 0. Notice that, in the second dance, Dancer 0 has moved to the inner position in the bottom-right pair. That's the spot that was occupied by Dancer 5 in the first dance. In that sense, from Dancer 0's perspective, she starts off the second dance at "the spot that Dancer 5 used to occupy."

We can do this for other people as well. Look at Dancer 8, for example. Dancer 8 ended up in the outer position in the bottom-right pair, which is where Dancer 7 used to be. So we could instruct Dancer 8 to get to his new location by saying "go to where Dancer 7 was in the previous dance."

How about Dancer 3? Notice that Dancer 3 started in the inner pair of the top-center pair, and that's where she ended up as well. If we wanted to instruct Dancer 3 how to prepare for the second dance, we could tell her "go to the spot that Dancer 3 was in in the first dance." It's a little verbose, but it works!

More generally, we can move from the first dance to the second by telling each dancer whose spot they should take. This method of rearranging a group of things (here, people, but in principle they could be anything) by indicating how each item takes on a position previously held by some item is called a permutation, which is at the heart of this problem.

To make this rigorous, let's introduce some notation. First, for any natural number k, let's have

$$[\![k]\!] = \{n \in \mathbb{N} \mid n < k\}$$

In other words, $[\![k]\!]$ is the set of all natural numbers less than k. The people in our dance can be represented as $[\![10]\!]$. Next, let's think of how everyone swaps around. This is something we can model as a function that takes as input a person, then outputs which person's position they should move to. In our case, we could represent this as a function $f : [\![10]\!] \to [\![10]\!]$. For example, we'd have $f(0) = 5$.

i. Just to make sure everything makes sense at this point, what is $f(6)$?

Formally speaking, a ***permutation*** of a collection of items A is a bijection $\sigma : A \to A$ from A to itself. (That's a lower-case Greek letter sigma, by the way, in case you haven't encountered it before.) There's a good reason this definition says a permutation is a *bijection*, rather than just a plain old *function*.

ii. Let's go back to our example of dancers changing places. Imagine that there's a dance where the dancers start off in some initial configuration. To set up for the next dance, each dancer moves to the spot previously occupied by one of the dancers, and after everyone has set up all positions are filled. If we model that change as a function as we did here, explain why that function must be a bijection. No formal proof is necessary, but you should address the rigorous definition of a bijection in your answer.

It might help to think of things this way: what happens if that function isn't a bijection?

There's a nice notation that's often used to describe permutations called ***two-line notation***. In the top line, we list the objects being permuted in some nice, human-readable order. Then, below each object, we write the object whose position it ends up in after the objects move. For example, the two-line notation

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 3 & 5 & 7 & 9 & 2 & 4 & 6 & 8 & 0 \end{pmatrix}$$

could be read as "Dancer 0 moves to the position previously held by Dancer 1, Dancer 1 moves to the position previously held by Dancer 3, Dancer 2 moves to the position previously held by Dancer 5, etc." This isn't the permutation described in the previous picture; it's just an example of the notation.

iii. Look back at the dances from the previous page. The function f we described earlier tells each dancer whose position to take when setting up for the second dance. Express the function f using two-line notation by filling in the following blanks:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \end{pmatrix}$$

Now, let's imagine that there's a third dance scheduled and the dancers yet again need to change places. At the end of the first dance, Dancer 0 moved to the spot that Dancer 5 started off in. To make things easier for Dancer 0, imagine that she adopts the following strategy: starting with the second dance, she always lines up for the next dance in by moving to the position Dancer 5 held in the prior dance.

iv. Where will Dancer 0 be at the start of the third dance? Provide your answer in the following way: determine which position Dancer 0 will be in, then look back at the original configuration of the dancers and tell us whose position Dancer 0 would be standing in. For example, if Dancer 0 would end up in the outer position in the bottom-right pair, you'd say that Dancer 0 ends in the position originally held by Dancer 7.

Now, imagine that *every* dancer adopts a strategy similar to Dancer 0. Each dancer n is assigned some dancer $f(n)$ that they're tasked with following. For each dance after the first, Dancer n then sets up at the spot where Dancer $f(n)$ was standing at the start of the previous dance.

v. If all the dancers adopt this strategy, where will Dancer 0 end up at the start of the fourth dance? Again, express your answer by looking back at the original configuration of dancers from the first dance and telling us whose position Dancer 0 will be occupying.

In parts (iv) and (v) of this problem, you've explored an important idea. We can use the original positions of the dancers as a way of identifying each location. That is, rather than saying "the dancer in the top center position," we can say "the position that Dancer 0 occupies in the first dance."

Let's imagine we want to know where some dancer is going to be in the third dance. We have a permutation f that explains how all the dancers change positions from the first dance to the second. Can we somehow manipulate f to see where everyone ends up for the third dance?

vi. Suppose you pick Dancer n and want to figure out where he starts in the third dance. Explain why he will be in the spot originally occupied by person $(f \circ f)(n)$ in the first dance. No formal proof is necessary.

vii. Starting with your answer from part (iii) of this problem, write out the permutation $f \circ f$ using two-line notation by filling in the following blanks:

You can solve this problem either by working this out mathematically, or by using the intuition from part (vi). Before you move on, make sure you can solve it both ways!

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ & \_ \end{pmatrix}$$

viii. Explain why Dancer n's position in the fourth dance is the given by the spot originally occupied by Dancer $(f \circ f \circ f)(n)$ in the first dance. No formal proof is necessary.

Given a permutation $\sigma : A \to A$, the ***kth power*** of σ, for some natural number $k \geq 1$, is defined as

$$\sigma^k = \sigma \circ \sigma \circ \cdots \circ \sigma,$$

where there are k copies of σ composed together. For example, $\sigma^4 = \sigma \circ \sigma \circ \sigma \circ \sigma$. As a useful edge case, we'll define σ^0 to be the function given by the rule $\sigma^0(x) = x$.

Powers of permutations give us a really nice way of figuring out where all the dancers will be at the start of the kth dance. Specifically, the positions are given by the outputs of f^{k-1}, where, as before, positions are given by the number of the dancer who was originally at that position in the first dance.

Imagine that the dancers keep changing their positions using the setup described above. At some point, it's guaranteed that all the dancers will end up returning to their original positions.

ix. How many different configurations will the dancers go through before everyone returns back to their original positions? Justify your answer, but no formal proof is necessary.

One final piece of terminology. If $\sigma : A \to A$ is a permutation, the order of σ is the smallest positive number k for which $\sigma^k = \sigma^0$. With regards to part (ix) of this problem, the order of f happens to be the number of unique configurations the dancers will end up going through.

Permutations are a useful building block in mathematics. We'll see them again in the next problem set, where we'll use them to talk about symmetries.

6 Left, Right, and True Inverses

Let $f : A \to B$ be a function. A function $g : B \to A$ is called a ***left inverse*** of f if the following is true:

$$\forall a \in A.\ g(f(a)) = a.$$

i. Find examples of a function f and two *different* functions g and h such that both g and h are left inverses of f. This shows that left inverses don't have to be unique. (Two functions g and h are different if there is some x where $g(x) \neq h(x)$.) Express your answer by drawing pictures of f, g, and h along the lines of what we did in class.

ii. Prove that if f is a function that has a left inverse, then f is injective.

As a hint on this problem, look back at the proofs we did with injections in lecture. To prove that a function is an injection, what should you assume about that function, and what will you end up proving about it?

Let $f : A \to B$ be a function. A function $g : B \to A$ is called a ***right inverse*** of f if the following is true:

$$\forall b \in B.\ f(g(b)) = b.$$

iii. Find examples of a function f and two different functions g and h such that both g and h are right inverses of f. This shows that right inverses don't have to be unique. As in part (i), express your answer by drawing pictures of f, g, and h along the lines of what we did in lecture.

iv. Prove that if f is a function that has a right inverse, then f is surjective.

If $f : A \to B$ is a function, then a ***true inverse*** (often just called an ***inverse***) of f is a function g that's simultaneously a left and right inverse of f. In parts (i) and (iii) of this problem you saw that functions can have several different left inverses or right inverses. However, a function can only have a single true inverse.

v. Prove that if $f : A \to B$ is a function and both $g_1 : B \to A$ and $g_2 : B \to A$ are inverses of f, then $g_1(b) = g_2(b)$ for all $b \in B$.

vi. Explain why your proof from part (v) doesn't work if g_1 and g_2 are just *left* inverses of f, not full inverses. Be specific – you should point at a specific claim in your proof of part (v) that is no longer true in this case.

vii. Explain why your proof from part (v) doesn't work if g_1 and g_2 are just *right* inverses of f, not full inverses. Be specific – you should point at a specific claim in your proof of part (v) that is no longer true in this case.

Left and right inverses have some surprising applications. We'll see one of them next week!

7 The Star-Drawing Saga, Part Four

On Problem Set 2, you proved that for natural numbers p and s where $p > 0$, the star $\{p \ / \ s\}$ is simple if and only if there is an integer t where $1 \equiv_p s \cdot t$. This theorem is useful for checking whether a particular star is simple, but it's fairly complicated to use in practice. What we'd ultimately like is some criterion we can use to quickly "eyeball" a pair of numbers p and s and to quickly tell whether $\{p \ / \ s\}$ is a simple star.

An observation you may have had so far is that in cases where we do have a simple star (say, $\{7 \ / \ 2\}$, $\{7 \ / \ 3\}$, $\{8 \ / \ 3\}$, $\{10 \ / \ 3\}$, $\{12 \ / \ 5\}$, etc.) the values of p and s have no common factors, whereas in the stars we've seen that aren't simple (say, $\{6 \ / \ 2\}$, $\{8 \ / \ 2\}$, $\{9 \ / \ 3\}$, $\{10 \ / \ 4\}$, etc.) the values of p and s do have a common factor. Perhaps that's important?

As a reminder from the checkpoint problem, if a and b are integers, we say that a ***divides*** b if there is an integer q such that $b = aq$. We'll say that a and b are ***coprime*** if there are no integers that divide both a and b besides ± 1.

With this in mind, we can form a guess of what we think is necessary for a star to work:

> ***Conjecture***: A p-point star with a step size of s is a simple star if and only if p and s are coprime.

At this point, this is just a conjecture – something we *think* is true, but aren't totally sure about. Our goal will be to try to see whether this statement is a ***theorem***, meaning that it's something we have a rigorous proof for, or whether it's actually not true.

We'll need to develop a little bit more mathematical machinery before we can fully take aim at resolving this question, but with the tools you've developed so far, you actually have enough to show that at least one direction of this result is true.

Let p and s be natural numbers where $p > 0$. Prove that if $\{p \ / \ s\}$ is a simple star, then p and s are coprime. You may want to use the following fact, which you can use without proof: if x and y are integers and $xy = 1$, then either $x = y = 1$ or $x = y = -1$. Also, feel free to use the result you proved on Problem Set 2.

You have three proof techniques at your disposal at this point: direct proof, proof by contradiction, and proof by contrapositive. If you're having trouble with one of these approaches, try switching to another.

We've included two optional fun problems for this problem set. Feel free to work through all of them, but please submit at most one of them for credit. If you submit answers to more than one, we won't have the bandwidth to grade all your answers and will just pick one arbitrarily. (Here, by "arbitrarily," we mean "based on a whim and without any deeper reason," as in "the CEO made her decisions arbitrarily and capriciously, much to the chagrin of her underlings.")

Optional Fun Problem One: High-Order Dancing (Extra Credit)

In Problem Five, we modeled ten dancers changing places as a permutation $\sigma : [\![10]\!] \to [\![10]\!]$. The order of that permutation, as you saw, corresponds to the number of unique configurations the dancers will take on before everyone returns to their starting positions.

Find a permutation $\sigma : [\![10]\!] \to [\![10]\!]$ of order 30. Justify your answer *without* actually computing σ^{30}.

Optional Fun Problem Two: Reversing a Fundamental Theorem (Extra Credit)

The Fundamental Theorem of Equivalence Relations (FToER) says that if R is an equivalence relation over a set A, then every $a \in A$ belongs to exactly one equivalence class of R. Here, an equivalence class is a set of the form $[x]_R = \{y \in A \mid xRy\}$.

The general convention is to not speak of the equivalence classes of a binary relation R unless R actually happens to be an equivalence relation. But let's say that we decide to break with that tradition. After all, for any relation R, the set $[x]_R = \{y \in A \mid xRy\}$ is a perfectly well-defined set – it just might not happen to correspond to our intuition about how equivalence classes behave.

For the purposes of this problem, if R is any binary relation over a set A, we'll call a set

$$[x]_R = \{y \in A \mid xRy\}$$

a ***pseudoequivalence class*** of R. If R is indeed an equivalence relation, then the pseudoequivalence classes of R are the equivalence classes of R, and if R isn't then these sets might not have a nice structure.

Prove or disprove: if R is a binary relation over a set A and every element of A belongs to exactly one pseudoequivalence class of R, then R is an equivalence relation. In other words, we'd like you to prove or disprove whether you can flip the direction of implication in the FToER.