

## Problem Set 5

---

This problem set – the last one purely on discrete mathematics – is designed as a cumulative review of the topics we’ve covered so far and a proving ground to try out your newfound skills with mathematical induction. The problems here span all sorts of topics – parallel processing, the nature of infinity, tiling problems, and social networks – and we hope that it serves as a fitting coda to our whirlwind tour of discrete math!

We recommend that you *read Handout #26, “Guide to Induction,” before starting this problem set*. It contains a lot of useful advice about how to approach problems inductively, how to structure inductive proofs, and how to not fall into common inductive traps. Additionally, before submitting, be sure to *read over Handout #27, the “Induction Proofwriting Checklist,”* for a list of specific things to watch for in your solutions before submitting.

As a note on this problem set – normally, you're welcome to use any proof technique you'd like to prove results in this course. On this problem set, we've specifically requested on some problems that you prove a result inductively. For those problems, you should prove those results using induction or complete induction, even if there is another way to prove the result. (If you'd like to use induction in conjunction with other techniques like proof by contradiction or proof by contrapositive, that's perfectly fine.)

As always, please feel free to drop by office hours, visit Piazza, or send us emails if you have any questions. We'd be happy to help out.

Good luck, and have fun!

**Due Friday, November 3<sup>rd</sup> at 2:30PM.  
There is no checkpoint problem.**

## Problem One: Chains and Antichains

Let  $A$  be an arbitrary set and  $<_A$  be some strict order over  $A$ . A **chain in**  $<_A$  is a series  $x_1, \dots, x_k$  of elements drawn from  $A$  such that

$$x_1 <_A x_2 <_A \dots <_A x_k.$$

Intuitively, a chain is a series of values in ascending order according to the strict order  $<_A$ . The **length** of a chain is the number of elements in that chain.

- i. Consider the  $\subsetneq$  relation over the set  $\wp(\{a, b, c\})$ , where  $A \subsetneq B$  means that  $A \subseteq B$  but  $A \neq B$ . What is the length of the longest chain in this strict order? Give an example of a chain with that length. No justification is necessary. (*Hint: Draw the Hasse diagram and see if you can find a visual intuition for the definition of a chain.*)

Now, let's cover a new definition. An **antichain in**  $<_A$  is a set  $X \subseteq A$  such any two elements in  $X$  are incomparable by the  $<_A$  relation. In other words, a set  $X \subseteq A$  is an antichain if

$$\forall a \in X. \forall b \in X. (a \not<_A b \wedge b \not<_A a)$$

The **size** of an antichain  $X$  is the number of elements in  $X$ .

- ii. Consider the  $\subsetneq$  relation over the set  $\wp(\{a, b, c\})$ . What is the size of the largest antichain in this strict order? Give an example of an antichain with that size. No justification is necessary. (*Hint: Draw the Hasse diagram and see if you can find a visual intuition for the definition of an antichain.*)

Given an arbitrary strictly ordered set, you can't say anything *a priori* about the size of the largest chain or antichain in that strict order. However, you can say that at least one of them must be relatively large relative to the strictly ordered set.

Let  $<_A$  be an arbitrary strict order over an arbitrary set  $A$  containing exactly  $n^2+1$  elements for some natural number  $n \geq 1$ . We're going to ask you to prove the following result: either  $A$  contains a chain of length  $n+1$  or an antichain of size  $n+1$  (or both). Following the advice from Handout 17, we'll prove this by instead proving that if  $A$  does *not* contain a chain of length  $n+1$  or greater, then  $A$  must contain an antichain of size  $n+1$  or greater.

- iii. For each element  $a \in A$ , we'll say that the **height** of  $a$  is the length of the longest chain whose final element is  $a$ . Prove that if  $A$  does not contain a chain of length  $n+1$  or greater, then there must be at least  $n+1$  elements of  $A$  at the same height. (*Something to think about: what's the smallest possible height of an element of  $A$ ?*)
- iv. Your result from part (iii) establishes that if  $A$  does not contain a chain of length  $n+1$  or greater, there must be a collection of  $n+1$  elements of  $A$  at the same height as one another. Prove that if  $A$  does not contain a chain of length  $n+1$  or greater, then it contains an antichain of size  $n+1$  or greater.

Intuitively speaking, if  $<_A$  is a strict order over  $A$  that represents some prerequisite structure on a group of tasks, a chain represents a series of tasks that have to be performed one after the other, and an antichain represents a group of tasks that can all be performed in parallel (do you see why?) In the context of parallel computing, the result you've proved states that if a group of tasks doesn't contain long dependency chains, that group must have a good degree of parallelism.

## Problem Two: Recurrence Relations

A *recurrence relation* is a recursive definition of the terms in a sequence. Typically, a recurrence relation specifies the value of the first few terms in a sequence, then defines the remaining terms from the previous terms. For example, the *Fibonacci sequence* can be defined by the following recurrence relation:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_{n+2} &= F_n + F_{n+1} \end{aligned}$$

The first terms of this sequence are  $F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, F_6 = 8$ , etc.

Some recurrence relations define well-known sequences. For example, consider the following recurrence relation:

$$\begin{aligned} a_0 &= 1 \\ a_{n+1} &= 2a_n \end{aligned}$$

The first few terms of this sequence are 1, 2, 4, 8, 16, 32, ..., which happen to be powers of two. It turns out that this isn't a coincidence – this recurrence relation perfectly describes the powers of two.

- i. Prove by induction that for any  $n \in \mathbb{N}$ , we have  $a_n = 2^n$ .

Minor changes to the recursive step in a recurrence relation can lead to enormous changes in what numbers are generated. Consider the following two recurrence relations, which are similar to the  $a_n$  sequence defined above but with slight changes to the recursive step:

$$\begin{array}{ll} b_0 = 1 & c_0 = 1 \\ b_{n+1} = 2b_n - 1 & c_{n+1} = 2c_n + 1 \end{array}$$

- ii. Find non-recursive definitions for  $b_n$  and  $c_n$ , then prove by induction that your definitions are correct.

Finding non-recursive definitions for recurrences (often called “solving” the recurrence) is useful in the design and analysis of algorithms. Commonly, when trying to analyze the runtime of an algorithm, you will arrive at a recurrence relation describing the runtime on an input of size  $n$  in terms of the runtime on inputs of smaller sizes. Solving the recurrence then lets you precisely determine the runtime. To learn more, take CS161, Math 108, or consider reading through the excellent textbook *Concrete Mathematics* by Graham, Knuth, and Patashnik.

### Problem Three: It'll All Even Out

Our very first proof by induction was the proof that for any natural number  $n$ , we have that

$$2^0 + 2^1 + 2^2 + \dots + 2^{n-1} = 2^n - 1.$$

This result is still true for the case where  $n = 0$ , since in that case the sum on the left-hand side of the equation is the *empty sum* of zero numbers, which is by definition equal to zero. It's also true for the case where  $n = 1$ ; in that case, the sum on the left-hand side of the equality just has a single term in it ( $2^0$ ) and the right-hand side has the same value.

Below is a proof by complete induction of an incorrect statement about what happens when you sum up zero or more real numbers:

**Theorem:** The sum of any number of real numbers is even.

**Proof:** Let  $P(n)$  be the statement “the sum of any  $n$  real numbers is even.” We will prove by complete induction that  $P(n)$  holds for all  $n \in \mathbb{N}$ , from which the theorem follows.

As a base case, we prove  $P(0)$ , that the sum of any 0 real numbers is even. The sum of any zero numbers is the empty sum and is by definition equal to 0, which is even. Thus  $P(0)$  holds.

For our inductive step, assume for some arbitrary  $k \in \mathbb{N}$  that  $P(0), \dots$ , and  $P(k)$  are true. We will prove that  $P(k+1)$  is true, meaning that the sum of any  $k+1$  real numbers is even. To do so, let  $x_1, x_2, \dots, x_k$ , and  $x_{k+1}$  be arbitrary real numbers and consider the sum

$$x_1 + x_2 + \dots + x_k + x_{k+1}.$$

We can group the first  $k$  terms and the last term independently to see that

$$x_1 + x_2 + \dots + x_k + x_{k+1} = (x_1 + x_2 + \dots + x_k) + (x_{k+1}).$$

Now, consider the sum  $x_1 + x_2 + \dots + x_k$  of the first  $k$  terms. This is the sum of  $k$  real numbers, so by our inductive hypothesis that  $P(k)$  is true we know that this sum must be even. Similarly, consider the sum  $x_{k+1}$  consisting of just the single term  $x_{k+1}$ . By our inductive hypothesis that  $P(1)$  is true, we know that this sum must be even.

Overall, we have shown that  $x_1 + x_2 + \dots + x_k + x_{k+1}$  can be written as the sum of two even numbers (namely,  $x_1 + x_2 + \dots + x_k$  and  $x_{k+1}$ ), so  $x_1 + x_2 + \dots + x_k + x_{k+1}$  is even. Thus  $P(k+1)$  is true, completing the induction. ■

Of course, this result has to be incorrect, since there are many sums of real numbers that don't evaluate to an even number. The sum  $2 + 3 + 4$ , for example, works out to 9, and the sum  $\pi + 1$  doesn't even work out to an integer!

What's wrong with this proof? Be as specific as possible. For full credit, you should be able to identify a specific claim made in the proof that is not correct, along with an explanation as to why it's incorrect.

## Problem Four: Cartesian Products and Cardinalities

The cardinality of the Cartesian product of two sets depends purely on the cardinalities of those sets, not on what the elements of those sets actually are. This question will ask you to prove this.

Let  $A$ ,  $B$ ,  $C$ , and  $D$  be sets where  $|A| = |C|$  and  $|B| = |D|$ . Our goal is to prove  $|A \times B| = |C \times D|$ . Since we know  $|A| = |C|$ , there has to be some bijection  $g : A \rightarrow C$ . Since we know  $|B| = |D|$ , there has to be some bijection  $h : B \rightarrow D$ . Now, consider the function  $f : A \times B \rightarrow C \times D$  defined as follows:

$$f(a, b) = (g(a), h(b))$$

That is, the output of  $f$  when applied to the pair  $(a, b)$  is an ordered pair whose first element is  $g(a)$  and whose second element is  $h(b)$ .

- i. Using the function  $f$  defined above, prove that if  $A$ ,  $B$ ,  $C$ , and  $D$  are sets where  $|A| = |C|$  and  $|B| = |D|$ , then we have  $|A \times B| = |C \times D|$ . Specifically, prove that  $f$  is a bijection between  $A \times B$  and  $C \times D$ . (By definition, two ordered pairs are equal if their first components are equal and their second components are equal.)

We can define the “Cartesian power” of a set as follows. For any set  $A$  and any positive natural number  $n$ , we define  $A^n$  inductively:

$$\begin{aligned} A^1 &= A \\ A^{n+1} &= A \times A^n \text{ (for } n \geq 1) \end{aligned}$$

Amazingly, we know that  $|\mathbb{N}| = |\mathbb{N}^2|$ , meaning that there’s the same number of pairs of natural numbers as there are natural numbers themselves. Feel free to use this fact in the following problem.

- ii. Using your result from (i), the above definition, and the fact that  $|\mathbb{N}| = |\mathbb{N}^2|$ , prove by induction that  $|\mathbb{N}^k| = |\mathbb{N}|$  for all nonzero  $k \in \mathbb{N}$ . This result means that for any nonzero finite  $k$ , there are the same number of  $k$ -tuples of natural numbers as natural numbers.

(Here’s some justification for the problem you just did. You don’t need to read this if you don’t want to, but we think you might find it interesting. ☺)

We’ve defined cardinality on a relative basis, defining  $|A| = |B|$ ,  $|A| \neq |B|$ ,  $|A| \leq |B|$ , and  $|A| < |B|$  in terms of functions between sets. That said, it is possible to treat set cardinalities as actual objects. Intuitively speaking, the cardinality of a set is a measure of how large that set is. For finite sets, the cardinality of that set will be a natural number, and for infinite sets the cardinality of that set will be an infinite cardinality, a generalization of the natural numbers that measure the sizes of infinite sets. For example,  $\aleph_0$ , which we introduced in our first lecture as the cardinality of  $\mathbb{N}$ , is an infinite cardinality.

Given cardinalities  $\kappa_1$  and  $\kappa_2$ , we define the product of those two cardinalities,  $\kappa_1 \cdot \kappa_2$ , to be  $|A \times B|$ , where  $A$  and  $B$  are any sets where  $|A| = \kappa_1$  and  $|B| = \kappa_2$ . For example,  $4 \cdot 3$  is *by definition* the cardinality  $|A \times B|$  for any set  $A$  of cardinality 4 and any set  $B$  of cardinality 3. Similarly, *by definition* the value of  $\aleph_0 \cdot \aleph_0$  is the cardinality  $|A \times B|$  for any sets  $A$  and  $B$  of cardinality  $\aleph_0$ .

To make sure that this definition is legal, we have to make sure that the cardinality of the Cartesian product depends purely on the cardinalities of the two sets, not their contents. For example, this definition wouldn’t give us a way to compute  $4 \cdot 3$  if the cardinality of the Cartesian product of a set of four apples and three oranges was different than the cardinality of the Cartesian product of a set of four unicorns and three ponies. We need to show that for any sets  $A$ ,  $B$ ,  $C$ , and  $D$ , that if  $|A| = |C|$  and  $|B| = |D|$ , then  $|A \times B| = |C \times D|$ . That way, when determining the value of  $\kappa_1 \cdot \kappa_2$ , it doesn’t matter which sets of cardinality  $\kappa_1$  and  $\kappa_2$  we pick; any choice works. Your proof from (i) filled in this step.

Your result from part (ii) shows that  $\aleph_0^n = \aleph_0$  for any positive natural number  $n$ . Isn’t infinity weird?

## Problem Five: Induction and Recursion

There's a close connection between mathematical induction and recursion, and many of the proofs by induction that we did in class can be thought of as claims about how specific recursive functions work.

One of the first proofs by induction that we did was to prove that, given a collection of  $3^n$  coins containing a single counterfeit coin that's heavier than the rest, it is always possible to discover which coin is fake using exactly  $n$  weighings on a balance. The key idea behind the proof was, essentially, a recursive algorithm that can be used to actually go and find which of the coins is counterfeit!

- i. Implement a recursive function

```
Coin counterfeitIn(std::vector<Coin> coins, Balance balance);
```

that takes as input a set of *exactly  $3^n$  coins* for some natural number  $n$ , one of which is counterfeit and weighs more than the rest, and returns which one that is. You're provided a balance you can use to weigh groups of coins and can make at most  $n$  weighings on that balance. Check the header `CounterfeitCoins.h` for a description of the relevant types here.

Our provided starter files provide an interface you can use to test out your function on a number of different inputs. It will show you which coins actually get weighed against one another. This is a great way to test your code and to see your algorithm in action!

Your code from part (i) shows that the *inductive* argument we made in class can be converted into a *recursive* function that actually finds the coin!

Now, here's a fun little variant on the counterfeit coin problem. Imagine that you're given a collection of coins. You're told that there *might* be a counterfeit in it, but then again, there might not be. If there is a counterfeit coin, it's guaranteed to be heavier than the rest. Your job is to determine whether there even is a counterfeit coin at all and, if so, to return which one it is.

- ii. Implement a recursive function

```
Coin maybeCounterfeitIn(std::vector<Coin> coins, Balance balance)
```

that takes as input a set of *exactly  $3^n - 1$  coins* for some natural number  $n$ , which *might* contain a counterfeit that weighs more than the rest. The function should either return the counterfeit coin if one exists, or return the special constant `None` if none of the coins are counterfeit. You're provided a balance you can use to weigh groups of coins and can make at most  $n$  weighings on that balance.

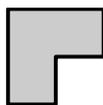
As with part (i) of this problem, you can test your code locally using our provided starter files, and it's pretty cool to see it run!

- iii. Using the recursive intuition that you developed in the course of solving part (ii) of this problem, prove that given any collection of exactly  $3^n - 1$  coins, of which at most one is a counterfeit that weighs more than the rest, it is always possible to identify which coin that is using at most  $n$  weighings on a balance (or to report that all coins are genuine). Your proof should have a similar structure to the one about counterfeit coins from lecture. While you should not explicitly reference the code you wrote in part (ii) of this problem, you may want to use the same recursive insight from that problem to guide the structure of your proof.

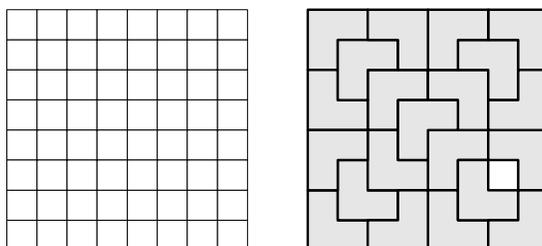
We hope that this exercise gives you a better sense for the interplay between theory (proof by induction) and practice (recursive problem-solving). If you're interested in this sort of thing, we strongly recommend checking out CS161, where you'll alternate between designing clever algorithms and using induction to prove that they work correctly.

## Problem Six: Tiling with Triominoes

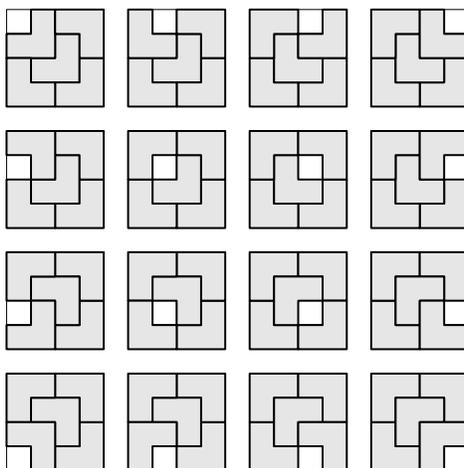
Recall from Problem Set One that a *right triomino* is an L-shaped tile that looks like this:



Suppose that you are also given a square grid of size  $2^n \times 2^n$  and want to *tile* it with right triominoes by covering the grid with triominoes such that all triominoes are completely on the grid and no triominoes overlap. Here's an attempt to cover an  $8 \times 8$  grid with triominoes, which fails because not all squares in the grid are covered:



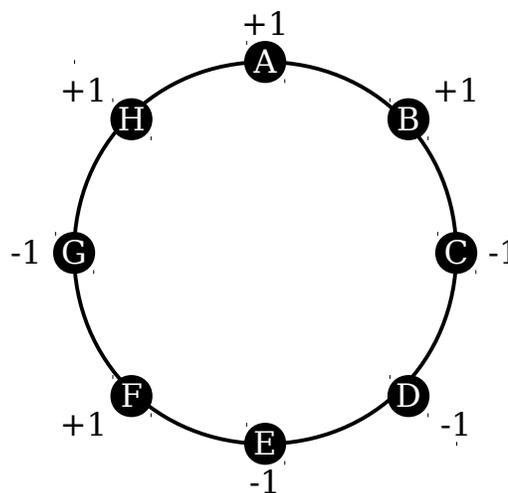
Amazingly, it turns out that it is always possible to tile any  $2^n \times 2^n$  grid that's missing exactly one square with right triominoes. It doesn't matter what  $n$  is or which square is removed; there is always a solution to the problem. For example, here are all the ways to tile a  $4 \times 4$  grid that has a square missing:



- i. Prove, by induction, that  $4^n - 1$  is a multiple of three for any  $n \in \mathbb{N}$ .
- ii. Draw a picture of a figure made of squares where the number of squares is a multiple of three, yet the figure cannot be tiled with right triominoes. Briefly justify your answer; no formal proof is necessary. This shows that your result from part (i) isn't sufficient to prove that any  $2^n \times 2^n$  grid missing a square can be tiled with right triominoes, since just knowing how many total squares there are isn't sufficient to show a tiling exists.
- iii. Prove by induction that for any natural number  $n$ , any  $2^n \times 2^n$  grid with any one square removed can be tiled by right triominoes.

## Problem Seven: The Circle Game

Here's a game you can play. Suppose that you have a circle with  $2n$  arbitrarily-chosen points on its circumference.  $n$  of these points are labeled  $+1$ , and the remaining  $n$  are labeled  $-1$ . One sample circle with eight points, of which four are labeled  $+1$  and four are labeled  $-1$ , is shown to the right.



Here's the rules of the game. First, choose one of the  $2n$  points as your starting point. Then, start moving clockwise around the circle. As you go, you'll pass through some number of  $+1$  points and some number of  $-1$  points. You lose the game if at any point on your journey you pass through more  $-1$  points than  $+1$  points. You win the game if you get all the way back around to your starting point without losing.

For example, if you started at point A, the game would go like this:

Start at A:  $+1$ .  
 Pass through B:  $+2$ .  
 Pass through C:  $+1$ .  
 Pass through D:  $0$ .  
 Pass through E:  $-1$ . (*You lose.*)

If you started at point G, the game would go like this:

Start at G:  $-1$  (*You lose.*)

However, if you started at point F, the game would go like this:

Start at F:  $+1$ .  
 Pass through G:  $0$ .  
 Pass through H:  $+1$ .  
 Pass through A:  $+2$ .  
 Pass through B:  $+3$ .  
 Pass through C:  $+2$ .  
 Pass through D:  $+1$ .  
 Pass through E:  $+0$ .  
 Return to F. (*You win!*)

Interestingly, it turns out that no matter which  $n$  points are labeled  $+1$  and which  $n$  points are labeled  $-1$ , there is always at least one point you can start at to win the game.

Prove, by induction, that the above fact is true for any  $n \geq 1$ .

## Problem Eight: Nim

*Nim* is a family of games played by two players. The game begins with several piles of stones, each of which has zero or more stones in it, that are shared between the two players. Players alternate taking turns removing any nonzero number of stones from any single pile of their choice. If at the start of a player's turn all the piles are empty, then that player loses the game.

Prove, by induction, that if the game is played with just two piles of stones, each of which begins with exactly the same number of stones, then the second player can always win the game if she plays correctly.

Before trying to write up your answer to this question, we recommend playing this game with a partner until you can find a winning strategy.

## Problem Nine: Independent Sets Revisited

Recall from Problem Set Four that an *independent set* in a graph  $G = (V, E)$  is a set  $I \subseteq V$  where no two nodes in  $I$  are adjacent to one another. As you saw on that problem set, many problems in graph theory (finding a large group of mutual strangers to survey, finding a small set of places to set up information kiosks in a city, etc.) boil down to finding large independent sets in a graph. The question then arises: how big of an independent set can you reasonably expect to find?

- i. Let  $G$  be a graph where every node in  $G$  has degree at most  $d$ . (As a reminder, the *degree* of a node in a graph is the number of nodes that it's adjacent to). Prove, by *complete* induction on  $n$ , that if  $G$  is a graph with  $n \geq 0$  nodes, then  $G$  has an independent set of size at least  $\frac{n}{d+1}$ .

Graphs are often used to model social networks: each person is a node, and friendships are represented by edges. The sorts of graphs you tend to find this way in the real world have all sorts of interesting properties. In particular, graphs like these often tend to have a lot of *triangles*, collections of three nodes that are all mutually adjacent. If you're interested in learning more about the properties of these sorts of graphs, take CS224W (Social and Information Network Analysis) or CS267 (Graph Algorithms).

This question explores a class of graphs that are quite different from the graphs that typically arise in social networks: *triangle-free graphs*. A triangle-free graph is one that contains no triangles. That is, if you pick three distinct nodes in the graph, some two of them will not be adjacent.

- ii. Prove that if  $G$  is a triangle-free graph with  $n^2$  nodes, then  $G$  contains an independent set of size at least  $n$ . As a hint, note that we didn't say to use induction here. As another hint, you may want to find a way to work in the result from part (i) of this problem.

## Optional Fun Problem: Synchronicity (1 Point Extra Credit)

Let's say that an *era* is a historical time period with a definitive start date and definitive end date. For example, the Meiji Era ran from October 23, 1868 to July 30, 1912, and the Cuban Missile Crisis ran from October 16, 1962 to October 28, 1962. For simplicity, we'll assume that these time ranges include the entirety of their start and end dates.

Prove that no matter how you choose any fifty eras from history, you can either (1) find a date that's contained in at least eight of those eras, or (2) find eight eras of which no two have any days in common.