Problem Set 2

This second problem set is all about induction and its sheer breadth of applications. By the time you're done with this problem set, you will have a much deeper understanding of how to think inductively. Plus, you'll learn a bit about data structures, recurrence relations, puzzles, games, and tournament structures!

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.

This problem set has 27 possible points. It is weighted at 5% of your total grade.

Good luck, and have fun!

Checkpoint Questions Due Monday, April 13 at the start of class
Remaining Questions Due Friday, April 17 at the start of class

Write your solutions to the following problems and submit them electronically on Scoryst by this Monday, April 13th at the start of lecture. As before, these problems are graded on a 0/1/2 scale based on whether or not you have made a good, honest effort to complete the problems. We will try to get these problems returned to you with feedback on your proof style this Wednesday, April 15th.

Please make the best effort you can when solving this problem. We want the feedback we give you on your solutions to be as useful as possible, so the more time and effort you put into them, the better we'll be able to comment on your proof style and technique.

Checkpoint Question: Recurrence Relations (2 Points if Submitted)

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:

$$F_0 = 0$$
$$F_1 = 1$$
$$F_{n+2} = F_n + F_{n+1}$$

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:

$$a_0 = 1$$
$$a_{n+1} = 2a_n$$

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:

$$b_0 = 1 \qquad\qquad c_0 = 1$$
$$b_{n+1} = 2b_n - 1 \qquad\qquad c_{n+1} = 2c_n + 1$$

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 *Concrete Mathematics* by Graham, Knuth, and Patashnik.

The remainder of these problems should be completed and submitted online by Friday, April 17th at the start of class.

Problem One: Fibonacci Numbers (4 Points)

In many cases, it's possible to simplify a complicated summation by replacing it with a simple expression. For example, in lecture we proved that $2^0 + 2^1 + 2^2 + \ldots + 2^{n-1} = 2^n - 1$. The course notes give a proof showing that $1 + 2 + 3 + \ldots + n = n(n+1) / 2$. This question explores a different summation.

The *Fibonacci sequence* is a famous sequence defined by the following recurrence relation:

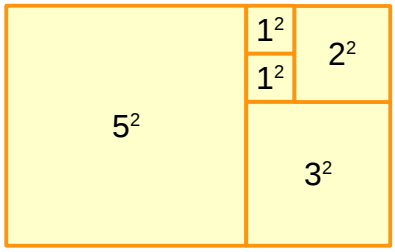
$$F_0 = 0 \qquad F_1 = 1 \qquad F_{n+2} = F_n + F_{n+1}$$

The first few Fibonacci numbers are

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

Now, think about the sum of the squares of Fibonacci numbers. Specifically, for $n \geq 1$, think about the sum $F_0^2 + F_1^2 + F_2^2 + \ldots + F_n^2$. Find a closed-form formula for this summation (that is, a simple expression that has the same value as $F_0^2 + F_1^2 + F_2^2 + \ldots + F_n^2$, but which doesn't involve a long summation), then prove by induction that your formula is correct.

As a hint, you might find the following picture useful:



This particular picture corresponds to the sum $0^2 + 1^2 + 1^2 + 2^2 + 3^3 + 5^2$. Try thinking about the dimensions of the rectangle and seeing if you spot a pattern.

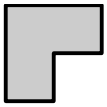
Problem Two: A Lot of Rocks? (2 Points)

Let's say that *a lot of rocks* is a number of rocks that isn't zero (surely, zero rocks is not a lot of rocks) and where if you remove one of the rocks, you're left with a lot of rocks (if there really are a lot of rocks, removing one rock shouldn't make a difference).

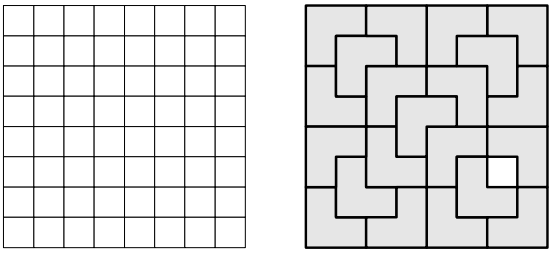
Prove by induction that no natural number of rocks is a lot of rocks. This hopefully teaches you to be careful with new definitions – just because we can define something doesn't mean it actually exists!

Problem Three: Tiling with Triominoes (3 Points)*

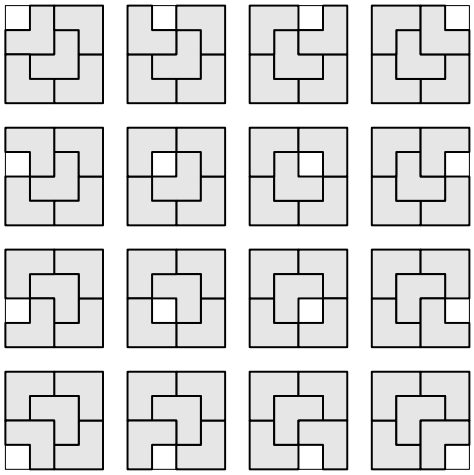
Recall from Problem Set One that a *right triomino* is an L-shaped tile that looks like this:



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 8×8 grid with triominoes, which fails because not all squares in the grid are covered:



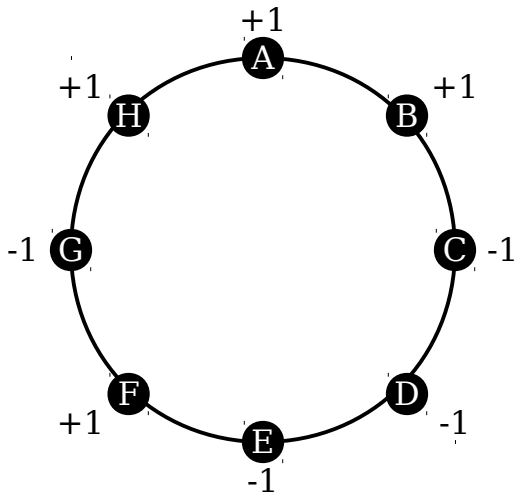
Amazingly, it turns out that it is always possible to tile any $2^n \times 2^n$ grid that's missing exactly one square with right triominoes. It doesn't matter what n is or which square is removed; there is always a solution to the problem. For example, here are all the ways to tile a 4×4 grid that has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



Prove by induction that any $2^n \times 2^n$ grid with any one square removed can be tiled by right triominoes. (*Hint: Split the board up into four smaller boards.)*

* I originally heard this problem from David Gries of Cornell University.

Problem Four: The Circle Game (4 Points)



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!)*

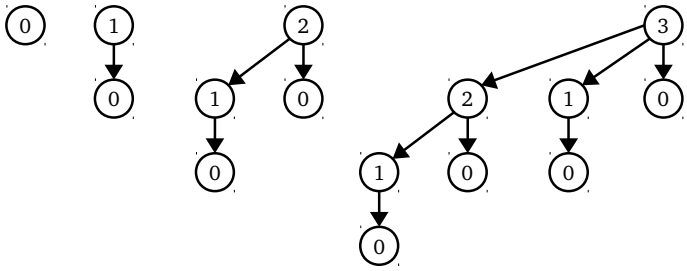
Interestingly, it turns out that no matter which n points are labeled +1 and which n points are labeled -1, there is always at least one point you can start at to win the game.

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

Problem Five: Binomial Trees (3 Points)

In computer science, a ***directed tree*** is a structure used to represent hierarchical information. A directed tree consists of a set of ***nodes***. Each node can have zero or more ***child nodes***. Every node has exactly one parent except for a special node called the ***root node***, which has no parent.

Binomial trees are a specific family of directed trees defined as follows: a binomial tree of order n is a single node with n children, which are binomial trees of order 0, 1, 2, …, $n - 1$. For example, here are pictures of binomial trees of orders 0, 1, 2, and 3:



Prove by induction that a binomial tree of order n has exactly 2^n nodes. Your proof should use to the formal definition of binomial trees given above – if you want to use any other facts about binomial trees in your proof of the main result, you will need to prove those other facts first.

Binomial trees are used as a building block in the *binomial heap* data structure, which can be used to efficiently determine the smallest element out of a group of values. Binomial heaps form the basis for a huge number of different data structures, such as the soft heap (used in a fast algorithm for the minimum spanning tree problem), the Fibonacci heap (used to implement fast versions of Dijkstra's shortest paths algorithm), and several others. If you're interested in learning more about the binomial heap and its applications, consider picking up a copy of *Introduction to Algorithms, Second Edition* by Cormen, Leiserson, Rivest, and Stein or taking CS166.

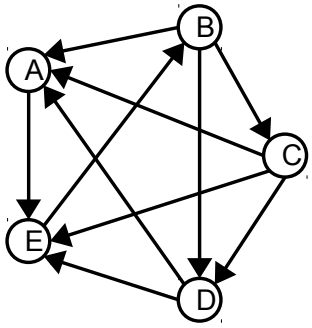
Problem Six: Nim (4 points)

Nim is a family of games played by two players. The game begins with several piles of stones 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 Seven: Tournament Winners (5 Points)



In Problem Seven, you explored elimination tournaments – tournaments in which players are constantly eliminated until only one player remains.

A different way to structure a tournament would be to have each player play everyone else. We'll say that a *tournament* is a contest among n players. Each player plays a game against each other player, and either wins or loses the game (let's assume that there are no draws). We can visually represent a tournament as a graph where each player is a node and each edge points from the winner of a game to the loser.

A *tournament winner* is a player in a tournament who, for each other player, either won her game against that player, or won a game against a player who in turn won his game against that player. For example, in the graph on the left, players *B*, *C*, and *E* are tournament winners. However, player *D* is ***not*** a tournament winner, because he neither beat player *C*, nor beat anyone who in turn beat player *C*. Although player *D* won against player *E*, who in turn won against player *B*, who then won against player *C*, under our definition player *D* is ***not*** a tournament winner. *(Make sure you understand why!)*

Prove, by induction, that every tournament with at least one player has a tournament winner.

Extra Credit Problem: Pie Fights! (1 Point Extra Credit)

Here's how a pie fight works: $n \geq 2$ people, each of whom has a pie in their hands, stands in a big open field. Everyone then throws their pie at the person closest to them. (To make sure that "the person closest to them" is well-defined, let's assume that no two pairs of people are standing at the same distance from one another). In the end, lots of people will end up covered in pie.

Here's a surprising result: in any pie fight with an odd number of people, at least one person won't have a pie thrown at them. Prove this result by induction. *(As a note: it is **not** the case that if person A throws a pie at person B, then person B will throw a pie at person A. Make sure you see why this is before you try to prove this result!)*