Problem Set 4

This fourth problem set explores set cardinality and graph theory. It serves as tour of the infinite (through set theory) and the finite (through graphs and their properties) and will give you a better sense for how discrete mathematical structures connect across these domains. Plus, you'll get to see some pretty pictures and learn about why all this matters in the first place. ☺

We know you have the midterm coming up on Monday, so we've tried to make this checkpoint problem a lot shorter than the one for PS3. We hope it solidifies the relevant concepts without diverting your attention from the exam.

Good luck, and have fun!

Checkpoint due Monday, October 23rd at 2:30PM

Remaining problems due Friday, October 27th at 2:30PM

Checkpoint Problem: A Really Simple Bijection? (2 Points if Submitted)

Consider the function $f : \mathbb{N} \to \mathbb{Z}$ defined as $f(n) = n$.

i. Prove that f is not a bijection.

Below is a purported proof that f is a bijection:

> ***Theorem:*** Let $f : \mathbb{N} \to \mathbb{Z}$ be defined as $f(n) = n$. Then f is a bijection.
>
> ***Proof:*** In lecture, we proved that $|\mathbb{N}| = |\mathbb{Z}|$. Since the sets $\mathbb{N}$ and $\mathbb{Z}$ have the same cardinality, we know that every function between them must be a bijection. In particular, this means that f must be a bijection, as required. ■

This proof has to be incorrect, since, as you proved in part (i), f isn't a bijection.

ii. What's wrong with this proof? Justify your answer.

Problem One: Set Cardinalities

Let's begin with a new definition. If A and B are sets, the ***Cartesian product*** of A and B, denoted $\boldsymbol{A \times B}$, is the set

$$\{ (x, y) \mid x \in A \wedge y \in B \}.$$

Intuitively, $A \times B$ is the set of all ordered pairs you can make by taking one element from A and one element from B, in that order. For example, the set $\{1, 2\} \times \{u, v, w\}$ is

$$\{ (1, u), (1, v), (1, w), (2, u), (2, v), (2, w) \}.$$

For the purposes of this problem, let's let ★ and ☺ denote two arbitrary objects where ★ ≠ ☺. Over the course of this problem, we're going to ask you to prove that $|\mathbb{N} \times \{★, ☺\}| = |\mathbb{N}|$.

Before starting this problem, you might want to play around with the set $\mathbb{N} \times \{★, ☺\}$ so that you can get a better feel for what it looks like.

i. Draw a picture showing a way to pair off the elements of $\mathbb{N} \times \{★, ☺\}$ with the elements of $\mathbb{N}$ so that no elements of either set are uncovered or paired with multiple elements.

ii. Based on the picture you came up with in part (i), define a bijection $f : \mathbb{N} \times \{★, ☺\} \to \mathbb{N}$. The inputs to this function will be elements of $\mathbb{N} \times \{★, ☺\}$, so you can define your function by writing

$$f(n, x) = \_\_\_\_\_$$

where $n \in \mathbb{N}$ and $x \in \{★, ☺\}$. (You cannot assume ★ or ☺ are numbers. They're arbitrary values out of your control.)

iii. Prove that the function you came up with in part (ii) is a bijection.

The result you've proved here essentially shows that $2\aleph_0 = \aleph_0$. Isn't infinity weird?

Problem Two: Understanding Diagonalization

Proofs by diagonalization are tricky and rely on nuanced arguments. In this problem, we'll ask you to review the formal proof of Cantor's theorem to help you better understand how it works. (Please read the Guide to Cantor's Theorem before attempting this problem.)

i. Consider the function $f : \mathbb{N} \to \wp(\mathbb{N})$ defined as $f(n) = \varnothing$. Trace through our formal proof of Cantor's theorem with this choice of f in mind. In the middle of the argument, the proof defines some set D in terms of f. Given that $f(n) = \varnothing$, what is that set D? Provide your answer without using set-builder notation. Is it clear why $f(n) \neq D$ for any $n \in \mathbb{N}$?

ii. Let f be the function from part (i). Find a set $S \subseteq \mathbb{N}$ such that $S \neq D$, but $f(n) \neq S$ for any $n \in \mathbb{N}$. Justify your answer. This shows that while the diagonalization proof will always find *some* set D that isn't covered by f, it won't find *every* set with this property.

iii. Repeat part (i) of this problem using the function $f : \mathbb{N} \to \wp(\mathbb{N})$ defined as

$$f(n) = \{ m \in \mathbb{N} \mid m \geq n \}$$

Now what do you get for the set D? Is it clear why $f(n) \neq D$ for any $n \in \mathbb{N}$?

iv. Repeat part (ii) of this problem using the function f from part (iii).

Problem Three: Simplifying Cantor's Theorem?

In our proof of Cantor's theorem, we proved that $|S| \neq |\wp(S)|$ by using a diagonal argument. Below is a purported proof that $|S| \neq |\wp(S)|$ that doesn't use a diagonal argument:

> ***Theorem:*** If S is a set, then $|S| \neq |\wp(S)|$.
>
> ***Proof:*** Let S be any set and consider the function $f : S \to \wp(S)$ defined as $f(x) = \{x\}$. To see that this is a valid function from S to $\wp(S)$, note that for any $x \in S$, we have $\{x\} \subseteq S$. Therefore, $\{x\} \in \wp(S)$ for any $x \in S$, so f is a legal function from S to $\wp(S)$.
>
> Let's now prove that f is injective. Consider any $x_1, x_2 \in S$ where $f(x_1) = f(x_2)$. We'll prove that $x_1 = x_2$. Because $f(x_1) = f(x_2)$, we have $\{x_1\} = \{x_2\}$. Since two sets are equal if and only if their elements are the same, this means that $x_1 = x_2$, as required.
>
> However, f is not surjective. Notice that $\varnothing \in \wp(S)$, since $\varnothing \subseteq S$ for any set S, but that there is no x such that $f(x) = \varnothing$; this is because $\varnothing$ contains no elements and $f(x)$ always contains one element. Since f is not surjective, it is not a bijection. Thus $|S| \neq |\wp(S)|$. ■

Unfortunately, this argument is incorrect. What's wrong with this proof? Justify your answer, and be as specific as possible.

Problem Four: Paradoxical Sets

What happens if we take *absolutely everything* and throw it into a set? If we do, we would get a set called the ***universal set***, which we denote $\mathscr{U}$:

$$\mathscr{U} = \{\ x \mid x \text{ exists}\ \}$$

Absolutely everything would belong to this set:

$$1 \in \mathscr{U} \qquad \mathbb{N} \in \mathscr{U} \qquad \text{CS103} \in \mathscr{U} \qquad \text{whimsy} \in \mathscr{U}$$

In fact, we'd even have $\mathscr{U} \in \mathscr{U}$, which is strange but not immediately a problem.

Unfortunately, the set $\mathscr{U}$ doesn't actually exist, as its existence would break mathematics.

i. Prove that if A and B are arbitrary sets where $A \subseteq B$, then $|A| \leq |B|$. *(Hint: Look at the Guide to Cantor's theorem. Formally speaking, if you want to prove that $|A| \leq |B|$, what do you need to prove? Your answer should involve defining some sort of function between A and B and proving that function has some specific property or properties.)*

ii. Using your result from (i), prove that if $\mathscr{U}$ exists, then $|\wp(\mathscr{U})| \leq |\mathscr{U}|$.

iii. Using your result from (ii) and Cantor's Theorem, prove that $\mathscr{U}$ does not exist. Feel free to use the following fact: for any sets A and B, if $|A| \leq |B|$ and $|B| \leq |A|$, then $|A| = |B|$.

The result you've proven shows that there is a collection of objects (namely, the collection of everything that exists) that cannot be put into a set. When this was discovered at the start of the twentieth century, it caused quite a lot of chaos in the math world and led to a reexamination of logical reasoning itself and a more formal definition of what objects can and cannot be gathered into a set. If you're curious to learn more about what came out of that, take Math 161 (Set Theory) or Phil 159 (Non-Classical Logic).

Problem Five: Independent Sets

An ***independent set*** in a graph $G = (V, E)$ is a set $I \subseteq V$ with the following property:

$$\forall u \in I.\ \forall v \in I.\ \{u, v\} \notin E.$$

This question explores independent sets and their properties.

i. Translate the definition of an independent set into plain English. No justification is necessary.

ii. Download the starter files for Problem Set Four from the website, then open the file `GraphTheory.cpp` and implement a function

```
bool isIndependentSet(Graph G, std::set<Node> I)
```

that takes as input a graph G and a set I, then returns whether I is an independent set in G. The definition of the `Graph` type is provided in `GraphTheory.h`.

Our provided starter code contains logic that, given a graph G, finds the largest independent set in G by making a lot of repeated calls to `isIndependentSet`. You might want to look over some of the sample graphs to get a feel for what large independent sets look like.

iii. You want to conduct a poll for an election and would like to survey people where no two people in the group know each other so that you can get a good representative sample of the population. Ideally, you'd find a large group of mutual strangers so that your poll has good statistical power. Explain how you might model this problem in terms of building some sort of graph, then finding a large independent set in it. Briefly justify your answer; no formal proof is necessary. *(We don't want you to design an algorithm for finding large independent sets; instead, imagine you have a program that finds large independent sets and explain how you'd use it to choose who to survey.)*

The size of an independent set is the number of nodes in it. Formally speaking, if I is an independent set, then the size of I is given by $|I|$. The ***independence number*** of a graph G, denoted $\boldsymbol{\alpha(G)}$, is the size of the largest independent set in G.

iv. Edit the file `PartA.graph` in the `res/` directory to define a graph G where G has exactly five nodes and $\alpha(G) = 5$.

v. Edit the file `PartB.graph` in the `res/` directory to define a graph G where G has exactly five nodes and $\alpha(G) = 1$.

Problem Six: Vertex Covers

A ***vertex cover*** in a graph $G = (V, E)$ is a set $C \subseteq V$ with the following property:

$$\forall u \in V.\ \forall v \in V.\ (\{u, v\} \in E \to u \in C \vee v \in C).$$

This question explores vertex covers and their properties.

i. Translate the definition of a vertex cover into plain English. No justification is necessary.

ii. Implement a function

```
bool isVertexCover(Graph G, std::set<Node> C)
```

that takes as input a graph G and a set C, then returns whether C is a vertex cover of G.

Our provided starter code contains logic that, given a graph G, finds the smallest vertex cover in G by making a lot of repeated calls to `isVertexCover`. You might want to explore some of the sample graphs to get a feel for what vertex covers look like.

iii. Suppose that you have a map of a city's roads and streets (assume that the roads are set up so that it's possible to walk between any two points in the city). You want to set up information kiosks so that no matter where you are, you never need to walk more than a block to get to a kiosk. You also want to use as few information kiosks as possible to accomplish this. Explain how you might model this problem by building some sort of graph and looking for a small vertex cover in that graph. Briefly justify your answer; no formal proof is necessary. *(Along the lines of part (iii) of the previous problem, assume you have a black box for finding small vertex covers and don't try to come up with an algorithm on your own.)*

The size of a vertex cover is the number of nodes in it. Formally speaking, if C is a vertex cover, then the size of C is given by $|C|$. The ***vertex cover number*** of a graph G, denoted $\boldsymbol{\tau(G)}$, is the size of the smallest vertex cover of G.

iv. Edit the file `PartC.graph` in the `res/` directory to define a graph G with exactly five nodes where $\tau(G) = 0$.

v. Edit the file `PartD.graph` in the `res/` directory to define a graph G with exactly five nodes where $\tau(G) = 4$.

Problem Seven: Independent Sets and Vertex Covers

(This problem builds on Problem Five and Problem Six, so you may want to complete those problems before starting this one.)

There's a close connection between independent sets and vertex covers.

i. Prove that if $G = (V, E)$ is a graph and I is an independent set in G, then $V - I$ is a vertex cover of G.

ii. Prove that if $G = (V, E)$ is a graph and C is a vertex cover of G, then $V - C$ is an independent set in G.

The connection you explored here shows that the problem of finding a maximum independent set in a graph is essentially the same as finding a minimum vertex cover in that graph. It's suspected that both of these problems are computationally infeasible for large graphs. We'll talk about this more when we discuss **NP**-completeness later in the quarter.

This result also shows, for any graph G, that $\alpha(G) + \tau(G) = n$.

Problem Eight: Chromatic and Clique Numbers

Recall from lecture that a ***k-vertex coloring*** of a graph is a way of coloring each node in the graph one of up to k different colors such that no two adjacent nodes are the same color. The ***chromatic number*** of a graph, denoted $\chi(G)$, is the minimum value of k for which a k-vertex coloring exists.

i. Implement a function

```
bool isKVertexColoring(Graph G,
                       std::map<Node, Color> colors,
                       std::size_t k);
```

that takes as input a graph, a mapping from nodes in the graph to colors, and a number k, then returns whether the indicated coloring is a k-vertex coloring. You can assume that the map has one key for each node in the graph and that the only keys in the map are nodes in G. (The type `std::size_t` represents a natural number.)

Our provided starter code contains some logic that, given a graph G, finds a minimum k-vertex-coloring of G by making a lot of calls to your `isKVertexColoring` function. We recommend taking some time to look at a few sample graphs and their minimum colorings – they're quite pretty!

Here's a new definition. A ***clique*** in a graph $G = (V, E)$ is a set $K \subseteq V$ with the following property:

$$\forall u \in K.\ \forall v \in K.\ (u \neq v \to \{u, v\} \in E).$$

This question explores the connection between cliques and chromatic numbers.

ii. Translate the definition of a clique into plain English. No justification is necessary.

iii. Implement a function

```
bool isClique(Graph G, std::set<Node> K)
```

that takes as input a graph G and a set K, then returns whether K is a clique in G.

Our provided starter code contains some logic that, given a graph G, finds the largest clique in G by making a lot of calls to your `isClique` function. You may want to take a look at some of the provided sample graphs to see what large cliques look like.

The size of a clique K, denoted $|K|$, is the number of nodes in K. The ***clique number*** of a graph, denoted $\omega(G)$, is the size of the largest clique in G.

iv. Prove that if G is a graph, then $\chi(G) \geq \omega(G)$.

v. Edit the file `PartE.graph` in the `res/` directory to contain a graph G where $\chi(G) \neq \omega(G)$. This shows that, in general, the chromatic and clique numbers of a graph don't have to equal one another.

Problem Nine: Bipartite Graphs

The ***bipartite graphs*** are a special class of graphs with applications throughout computer science. An undirected graph $G = (V, E)$ is called ***bipartite*** if there exist two sets V_1 and V_2 such that

- every node $v \in V$ belongs to exactly one of V_1 and V_2, and
- every edge $e \in E$ has one endpoint in V_1 and the other in V_2.

The sets V_1 and V_2 here are called the ***bipartite classes*** of G.

To help you get a better intuition for bipartite graphs, let's consider an example. Suppose that you have a group of people and a list of restaurants. You can illustrate which people like which restaurants by constructing a bipartite graph where V_1 is the set of people, V_2 is the set of restaurants, and there's an edge from a person p to a restaurant r if person p likes restaurant r.

Bipartite graphs have many interesting properties. One of the most fundamental is this one:

An undirected graph is bipartite if and only if it contains no cycles of odd length.

Intuitively, a bipartite graph contains no odd-length cycles because cycles alternate between the two groups V_1 and V_2, so any cycle has to have even length.

The trickier step is proving that if G is a graph that contains no cycles of odd length, then G has to be bipartite. For now, assume that G has just one connected component; if G has multiple connected components, we can treat each one as a separate graph for the purposes of determining whether G is bipartite. (You don't need to prove this, but I'd recommend taking a minute to check why this is the case.)

Suppose G is a connected, undirected graph with no cycles of odd length. Choose any node $v \in V$. Let V_1 be the set of all nodes that are connected to v by a path of odd length and V_2 be the set of all nodes connected to v by a path of even length. (Note that these paths do not have to be simple paths.) Formally:

$$V_1 = \{ x \in V \mid \text{there is an odd-length path from } v \text{ to } x \}$$

$$V_2 = \{ x \in V \mid \text{there is an even-length path from } v \text{ to } x \}$$

i. Prove that V_1 and V_2 have no nodes in common.

ii. Using your result from part (i), prove that if G is connected and has no cycles of odd length, then G is bipartite. *(Hint: look back at the definition of a bipartite graph. What exactly do you need to prove to show that a graph is bipartite?)*

Problem Ten: Chromatic and Independence Numbers

(This problem builds on Problem Five, so you may want to complete that problem before starting this one.)

In Problem Eight, you explored the connection between clique numbers and chromatic numbers. This problem explores the connection between independence numbers and chromatic numbers.

Let n be an arbitrary positive natural number. Prove that if G is an arbitrary undirected graph with exactly n^2+1 nodes, then $\chi(G) \geq n+1$ or $\alpha(G) \geq n+1$ (or both). As a hint, look at Handout 17's advice about how to prove a statement of the form $P \vee Q$.

Optional Fun Problem 1: Hugs All Around! (1 Point Extra Credit)

There's a party with 137 attendees. Each person is either ***honest***, meaning that they *always* tell the truth, or ***mischievous***, meaning that they *never* tell the truth.

After everything winds down, everyone is asked how many ***honest*** people they hugged at the party. Surprisingly, each of the numbers 0, 1, 2, 3..., and 136 was given as an answer exactly once.

How many honest people were at the party? Prove that your answer is correct and that no other answer could be correct.

Optional Fun Problem 2: How Many Functions Are There? (1 Point Extra Credit)

If A and B are sets, we can define the set $\boldsymbol{B^A}$ to be the set of all functions from A to B. Formally speaking:

$$B^A = \{ f \mid f : A \to B \}$$

Using the formal definition of the less-than relation over cardinalities, prove that $|\mathbb{N}| < |\mathbb{N}^{\mathbb{N}}|$. This shows that $\aleph_0 < \aleph_0^{\aleph_0}$. Isn't infinity weird?