Exercise 1: SOLUTIONS

HONOR CODE NOTICE: SOLUTION SET, for use by CS3102 at UVA students in Spring 2020 only. Anyone reading this outside of that course for that semester is in violation of the honor code.

Exercise 1-1: Induction Practice

Prove that for any natural number $n \geq 2$, $n! < n^n$.

To show:

$$n! < n^n$$

We will show this using a proof by induction.

Base case: $n = 2$

$$2! < 2^2$$
$$2 < 4$$

So the base holds

Inductive Hypothesis:

Assume $k! < k^k$ for arbitrary $k \geq 2$

Inductive Step: Show $k! < k^k \rightarrow (k+1)! < (k+1)^{k+1}$

To begin, consider $(k+1)!$:

$$\begin{aligned}(k+1)! &= (k+1) \cdot k! \\ (k+1) \cdot k! &< (k+1)k^k \\ (k+1)k^k &< (k+1)(k+1)^k \\ (k+1)(k+1)^k &= (k+1)^{k+1}\end{aligned}$$

and so $(k+1)! < (k+1)^{k+1}$

Exercise 1-2: LED Bus Displays

The UTS buses have LED displays that give information about route, service, etc. These displays are 97 pixels wide and 17 pixels tall. Each pixel could have 1 of 2 colors: orange (on) or black (off). Answer the following questions of this display. Support your answer with a proof involving bijections between sets.

1. We will store the contents of display in binary. Assuming all configurations are represented with the same number of bits, what is the minimum number of bits required to do so? Justify your answer by demonstrating a bijection between all strings of the length you indicate, and all possible configurations of the display.
2. Suppose, to limit the energy needed to power the display, we required that no more than half of the pixels could be on at a time. Assuming all configurations are represented with the same number of bits, what is the minimum number of bits required to represent the contents of the display, now? Justify your answer by demonstrating a bijection between configurations of the display which are majority orange vs. those that are majority black.

Part 1) We will show that $97 \cdot 17 = 1649$ bits are required. To do this, we will find a bijection between the images displayable on the bus display, and binary strings of length 1649.

Mapping from images to $\{0,1\}^{1649}$: We begin by describing how to map images to binary strings.

Consider that we have a 97×17 image p. I will first number the pixels of p row-by-row (top-to-bottom, left-to-right) so that the top-left most pixel is labelled with 0, and the bottom-right most pixel is labelled with 1648. I will use p_i to refer to the pixel labelled with natural number i.

The image p will map to a bitstring b by assigning $b_i = 1$ if p_i is on (orange), and $b_i = 0$ otherwise (i.e. if p_i is black). Next we will show that this mapping is a bijection.

The mapping is one-to-one:

Consider that we have two different images p' and p''. We will show that the binary strings they map to (call them b' and b'' respectively) must be different as well.

For images p' and p'' to be different, they must differ by at least one pixel. Let i represent the index of this pixel. Since $p'_i \neq p''_i$ it must be the case according to our mapping that $b'_i \neq b''_i$, and so $b' \neq b''$.

The mapping is onto:

Consider an arbitrary binary string $b \in \{0,1\}^{1649}$, we will show that some image p maps to it. Essentially, this will be done with the "inverse" mapping of the above. The pixel p_i in p be on (orange) if $b_i = 1$ and black if $b_i = 0$. The produced image will map to b.

Conclusion:

Since there is a bijection between images and binary strings of length 1649, it must be that those two sets have the same cardinality. If we were use binary strings shorter than 1649, the set of binary strings must be smaller than the set of images, and so no onto mapping could exist. Thus we conclude that 1649 is the smallest number of bits which allow us to encode all binary strings.

Part 2) For this section we will show that we need 1648 (i.e. 1 fewer) bits to represent all images if we require the majority be off. We will do this by showing that the number of majority-on images is equal to the number of majority-off images. Since there are an odd number of pixels, every image must be either majority-on or majority-off, and so each of those sets must have half the cardinality of the set of all images. Since $2^{1648} = 2^{1649} \cdot \frac{1}{2}$, we can conclude that 1648 bits suffice.

The Mapping: To be able to conclude the above, we will show that there is a bijection between the set of majority-on images and the set of majority-off images. To do this, we will take a majority-on image and invert all of its pixels. Since that image was majority-on, and all orange pixels become black (and vice-versa), the new image is majority black.

Demonstrating it's a bijection: This mapping is one-to-one because if we have majority-orange images that differ by some pixel, those pixels will still be different in the majority-black images they map two (since both were inverted, they remain different). This mapping is onto because any majority-black image can be converted to a majority-orange image which maps to it by inverting the bits (the inverse of the inverse will get you back to the original image).

Exercise 1-3: Countable Graphs

The book defines an *undirected graph* (Definition 1.3). We modify this by adding one word to define a *undirected finite graph*:

Definition 1 (Undirected finite graph) An undirected finite graph $G = (V, E)$ consists of a *finite* set V of vertices and a set E of edges. Every edge is a size-two subset of V.

Prove that the set of all undirected finite graphs is *countably infinite*.

To demonstrate that the number of finite graphs is countable, we will show a surjection from the Natural Numbers onto the set of finite graphs. Since a surjection from $\mathbb{N}$ to finite graphs (call the set of finite graphs G_f) means $|\mathbb{N}| \geq |G_f|$, and $\mathbb{N}$ is countable, we can conclude that G_f is countable as well. To then show that G_f is countably infinite, we will show that G_f is infinite.

G_f is countable: Consider that we have a graph of n vertices. For an undirected graph, any (unordered) pair of vertices defines a unique edge. We next derive an upper bound on the number of distinct graphs of n vertices. The number of unordered pairs of n vertices is given by $\binom{n}{2} = \frac{n(n-1)}{2}$. Any graph of n nodes can be defined by a subset of the edges (identifying which edges belong to that graph), and therefore the number of unique graphs is upper-bounded by the cardinality of the powerset of the set of all unordered pairs of vertices, i.e. $2^{\frac{n(n-1)}{2}}$. Most importantly, this implies there are finitely many graphs of fixed size n.

Since for any n, there are only finitely many graphs, we can define a surjection from $\mathbb{N}$ to G_f by mapping 0 to the graph of 0 nodes, then 1 to the graph of 1 node, then $2, 3$ to each of the graphs with 2 nodes (one where the edge is present, the other where the edge is not), and for each n we will map the next $2^{\frac{n(n-1)}{2}}$ natural numbers to all the graphs of n nodes. Since every graph will appear on this list, by construction, this defines a surjection from the natural numbers to the finite graphs, and so the number of finite graphs is countable.

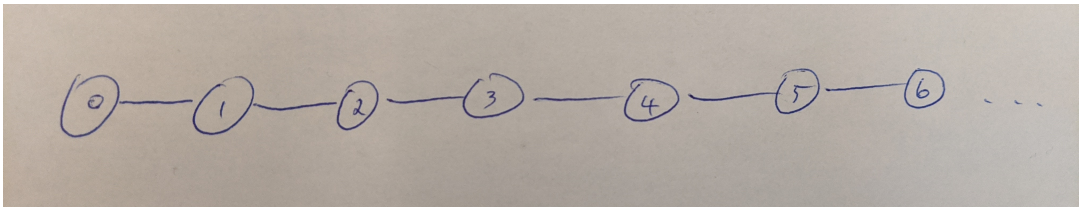
G_f is infinite: Since for any natural number n we can find a graph with n nodes (e.g. the graph with n nodes and 0 edges), and there are an infinite number of natural numbers, the number of graphs is also infinite.

Exercise 1-4: Uncountable Sets

Prove that the set of all undirected graphs (using the book's Definition 1.3, without the constraint that V is finite that was added for the previous problem) is not countable. (Note: be careful in any argument that you make that the graphs you are counting as actually *different*. The nodes on the graph have no labels, so the only way for two graphs to be different is if they have different structure.)

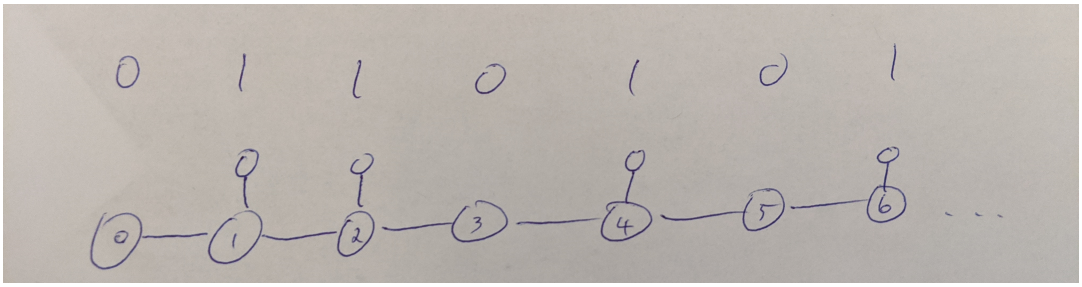
To show that the number of graphs (with a potentially infinite number of vertices) is uncountable, we will provide an injective (one-to-one) mapping from the inifinite binary strings (denote with $\{0,1\}^\infty$) to the infinite graphs (denote with G_∞).

To define this one-to-one mapping, the idea is to have a graph "gadget" to use as the basis to represent a string. We will consider a graph with an infinite number of vertices arranged in a "chain" like this:



In this case we add labels to each vertex for convenience. We will say that vertex v_0 is the (unique) node with degree 1 (i.e. only has one edge connected to it). The vertex adjacent to vertex v_0 we will call v_1, the vertex adjacent to v_1 that is not v_0 we ill call v_2, and in general will will label the vertices adjacent to v_i as v_{i+1} and v_{i-1} (with v_{i-1} as that vertex that is closer to v_0). In this way, we have defined an ordering of the vertices.

We can then map a binary string b a graph like the one above by adding additional vertices for each 1 that appears in b. In particular, if bit b_i in b is a 1, we will add an additional vertex to the graph that has an edge only to v_i. The following image illustrates an example of this mapping.



The mapping is one-to-one: To show that the mapping is one-to-one, we must show that two different strings map to different graphs. In this case, we know that if string $b' \neq b''$, then they must differ by at least one bit. Let's say that they differ at bit i, and without loss of generality we can say that $b' = 1$ and $b'' = 0$. In this case, if b' maps to graph g' and b'' maps to graph g'', we know that $g' \neq g''$ because vertex i in g' will have degree 3 whereas vertex i in graph g'' will have degree 2.