Recap

- Discussed sets, set operations, set equality proofs
- Discussed functions (injections, surjections, bijections)
  one-to-one, onto
- Found bijection $f: \mathbb{N} \to 2\mathbb{N}$ ↪ evens

But $2\mathbb{N} \subsetneq \mathbb{N}$! Weird...

Today: Define cardinality for infinite sets.

Infinite Cardinality

Let's recall some stuff from last time.

Def: Let $f: X \to Y$ be a function.

- f is an injection if for all $x_1, x_2 \in X$,
  $f(x_1) = f(x_2) \Rightarrow x_1 = x_2$.
- f is a surjection if for all $y \in Y$,
  there is $x \in X$ such that $f(x) = y$.
- f is a bijection if it is injective and surjective.

Theorem: If X, Y are finite, there is a injection/surjection/bijection $f: X \to Y$ iff $|X| \leq |Y|$ / $|X| \geq |Y|$ / $|X| = |Y|$.

Go back to bijection $f: \mathbb{N} \to 2\mathbb{N}$.

Moral: Can't assign absolute size to infinite sets!

Trick: Define relative size for infinite sets using functions.

Def: If X, Y are sets, we write "$|X| = |Y|$" if there is a bijection $f: X \to Y$.

Subtle point: For finite sets, $|X| = |Y|$ already had a meaning, and the above was a property. Now, we extend the meaning of $|X| = |Y|$ to infinite sets using the property as a definition.

Warning: $|X|$ has no meaning if X is infinite. We only put $|X| = |Y|$.

We do the same for $|X| \leq |Y|$.

Def: If X, Y are sets, we write $|X| \leq |Y|$ if there is an injection $f: X \to Y$.

$|X|=|Y|$ and $|X|\le|Y|$ is just notation.

But the notation suggests certain properties.

Property	Proof
$\lvert X\rvert=\lvert Y\rvert \Rightarrow \lvert X\rvert\le\lvert Y\rvert$	Every bijection is an injection.
$\lvert X\rvert=\lvert X\rvert$	$id_X : X\to X$ is a bijection.
$\lvert X\rvert=\lvert Y\rvert \Rightarrow \lvert Y\rvert=\lvert X\rvert$	If $f:X\to Y$ is bijection, then $f^{-1}:Y\to X$ is bijection
$\lvert X\rvert=\lvert Y\rvert$, $\lvert Y\rvert=\lvert Z\rvert$ $\Rightarrow \lvert X\rvert=\lvert Z\rvert$	If $f:X\to Y$ is bijection, $g:Y\to Z$ is bijection, then $g\circ f: X\to Z$ is bijection (Check this!)

So, $|X|=|Y|$ has many expected properties of equality!

Countable Sets

Idea: $\mathbb{N}=\{0,1,2,\dots\}$ is the "simplest" infinite set.

So, look for infinite sets same size as $\mathbb{N}$.

Def: • X is <u>countably infinite</u> if $|X|=|\mathbb{N}|$.

• X is <u>countable</u> if it is countably infinite or finite.

Why countable? Say X is countably infinite.

Then, there is a bijection $f: \mathbb{N} \to X$.

We can use f to list elements of X:

$f(0), f(1), f(2), \ldots \in X$

Since f is bijective, each $x \in X$ is in the list exactly once.

Conversely: Listing elements of X gives a bijection $X \to \mathbb{N}$. How?

List: $x_0, x_1, x_2, \ldots \in X$

$$\begin{array}{cccccc} f: & \downarrow & \downarrow & \downarrow & \\ & 0, & 1, & 2, & \ldots \end{array}$$

$\longrightarrow$ If the list has each element of X exactly once, $f: X \to \mathbb{N}$ is a bijection!

Takeaway: Countable $\Leftrightarrow$ Listable elements

Ex: $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ is countable.

Proof: We can list $\mathbb{Z}$: $0, -1, 1, -2, 2, -3, 3, \ldots$

The list gives a bijection

$$\begin{array}{cccccc} 0, & -1, & 1, & -2, & 2, & -3, \ldots \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 2 & 3 & 4 & \ldots \end{array}$$

$\square$

Warning: Hidden list rules.

Consider listing $\mathbb{Z}$: $0, 1, 2, 3, \dots, -1, -2, \dots$

Q: What position does -1 have? A: "∞"!

"∞" is <u>not</u> in $\mathbb{N}$, so this list does <u>not</u> give a bijection $\mathbb{Z} \to \mathbb{N}$.

Moral: When listing all elements must have <u>finite</u> position.

Prop: Say $A_1, \dots, A_n$ are countably infinite sets with no overlap. Then $\bigcup_{i=1}^{n} A_i$ is countably infinite.

Proof: For each i, A_i is countably infinite.

So, we can list its elements.

Consider writing out lists.

$$\begin{array}{lllll}
A_1 : & a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\
A_2 : & a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\
A_3 : & a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\
\vdots & & & & \\
A_n : & a_{n,1} & a_{n,2} & a_{n,3} & \cdots
\end{array}$$

By zig-zagging through the lists like in blue, we get a list of elements of $\bigcup_{i=1}^{n} A_i$. Done! $\square$

"such that"

Ex: $\mathbb{Q} = \{ \frac{a}{b} \mid a, b \in \mathbb{Z},\ b \neq 0 \}$ is countable.

For simplicity, let's do $\mathbb{Q}^+$, the positive rationals.

To list positive rationals, make a grid:

$$
\begin{array}{ccccc}
\text{0th}\ 1/1 & 1/2 & 1/3 & 1/4 & \cdots \\
2/1 & \text{4th}\ 2/2 & 2/3 & 2/4 & \\
3/1 & 3/2 & 3/3 & & \\
4/1 & 4/2 & & & \\
\vdots & & & &
\end{array}
$$

Problem! Our list has repeats!

For instance: $1/1 = 1$ and $2/2 = 1$.

So, the "function" $\mathbb{Q}^+ \to \mathbb{N}$ we get maps $1 \mapsto 0$ and $1 \mapsto 4$.

Not even a real "well-defined" function!

"Fix": Map each element of $\mathbb{Q}^+$ to its first position in list, e.g. $1 \to 0$, $1/2 \to 1$, $2 \to 2$, $3 \to 3$, $1/3 \to 5, \ldots$

Problem: Nothing maps to 4! Not bijection!

But this is an injection $\mathbb{Q}^+ \to \mathbb{N}$. So, we showed $|\mathbb{Q}^+| \le |\mathbb{N}|$.

What about the other way? Consider $f: \mathbb{N} \to \mathbb{Q}^+$ given by $f(n) = \frac{n+1}{1}$. This is an injection!

So, $|\mathbb{N}| \le |\mathbb{Q}^+|$!

Can we conclude $|\mathbb{Q}^+| = |\mathbb{N}|$? Not right away!

Theorem (Cantor-Schöder-Bernstein): If there exist injections $X \to Y$ and $Y \to X$ then there is a bijection $X \to Y$.

(i.e. $|X| \le |Y|$ and $|Y| \le |X| \Rightarrow |X| = |Y|$)

Proof: See typed notes. Harder than most 70 proofs, but understandable. $\square$

Using CSB: $\mathbb{Q}^+$ is countable!

Prop: $\mathbb{Q}$ is countable.

Proof: Can show $\mathbb{Q}^{\leq 0}$ is countable → neg rationals and 0

using same method as $\mathbb{Q}^+$.

Then, $\mathbb{Q} = \mathbb{Q}^+ \cup \mathbb{Q}^{\leq 0}$ is countable

by our earlier result. $\square$

Moral: If we want to show $|X| = |Y|$,

sometimes finding two injections is easier.

Uncountable Sets

Let's consider the set of reals $\mathbb{R}$.

We will view $\mathbb{R}$ as the set of all numbers

with an infinite decimal expansion.

$\frac{1}{2} = 0.5000000\ldots$

$\frac{4}{3} = 1.333333\ldots$

$\pi = 3.141592653\ldots$

Theorem: $\mathbb{R}$ is uncountable (no bijection $\mathbb{N} \to \mathbb{R}$).

Proof: For contradiction, say $\mathbb{R}$ is countable.

Then, we can list its elements

In particular, can list elements of

$$(0,1) = \{x \in \mathbb{R} \mid 0 < x < 1\}.$$

Consider <u>any</u> list of $(0,1)$. For instance,

$$
\begin{array}{l}
0.18442\cdots \\
0.30664\cdots \\
0.01847\cdots \\
0.01101\cdots
\end{array}
$$

Make a new number: Take circled digits and change nonzero digits to 0, change zeros to 1.

So, get $\underline{0.0101\cdots}$.

Notice: This new number is different than 1st num in 1st decimal, 2nd num in 2nd decimal, ...

So, it is <u>missing</u> from the list!

Contradiction! $\square$

This is called a <u>diagonalization argument</u>.

Countability Summary

Prove countability: • List elements
Can use CSB → • Find bijection to countable set

Prove uncountability: • Diagonalization
Can use CSB → • Find bijection to uncountable set

Uncountability heuristic? 1. Set is infinite
2. Each element is infinitely complex

Final Ex: $\{f : \mathbb{N} \to \mathbb{N}\}$ is uncountable.

Proof: By diagonalization. Suppose we list all functions.

f_0 : $\boxed{f_0(0)}$ $f_0(1)$ $f_0(2)$ . . .

f_1 : $f_1(0)$ $\boxed{f_1(1)}$ $f_1(2)$ . . .

⋮

Make a new function f so that

$f(0) \neq f_0(0)$, $f(1) \neq f_1(1)$, . . .

Missing from our list! $\square$

Computability

What is a computer program?

A piece of finite text that can be encoded in binary. So...

$\boxed{\text{A computer program is a finite binary string}}$

The set of finite binary strings is countable (list by length) so set of computer programs is countable.

What functions might we want to implement as computer programs?

The input/output of a computer program can be viewed as a finite binary string....

There is a countably infinite set of possible inputs/outputs for a program!

So, $|\{f : \{\text{inputs}\} \to \{\text{outputs}\}\}| = |\{f : \mathbb{N} \to \mathbb{N}\}|$

↳ uncountable

Upshot: The set of functions which we might want to implement as programs is uncountable, but the set of programs is countable! Some functions cannot be implemented!

We call such functions <u>uncomputable</u>.

Can we find one?

Halting Problem

Consider the following function:

$$\text{TestHalt}(P, x) = \begin{cases} 1, & P(x) \text{ halts in finite time} \\ 0, & P(x) \text{ loops forever} \end{cases}$$

binary rep of a program

binary rep of an input to P

Idea: Checking if P(x) halts is hard... How do we know if it is stuck in a loop or about to finish?

Theorem (Turing): TestHalt is uncomputable.

Proof: Suppose we have written a computer program which implements TestHalt.

Consider defining another function.

binary string

interp. P as program and input

```
def Turing(P):
    if TestHalt(P, P) == 1 :  # P(P) halts
        enter an infinite loop
    else:  # P(P) loops forever
        return
```

Consider Turing(Turing). binary rep of Turing

1. If Turing(Turing) halts, then TestHalt(Turing, Turing) == 0.
   But this means Turing(Turing) loops forever! Makes no sense.
2. If Turing(Turing) loops forever, TestHalt(Turing, Turing) == 1.
   But this means Turing(Turing) halts.
   Makes no sense.

So, Turing(Turing) cannot halt or loop forever. Contradiction! $\square$

We found an uncomputable function!

But the proof was weird...

Good news: We can use TestHalt to show other things are uncomputable!

Ex: $\text{HaltsOnZero}(P) = \begin{cases} 1, & P(0) \text{ halts} \\ 0, & P(0) \text{ loops forever} \end{cases}$

binary rep of program

Prop: HaltsOnZero is uncomputable.

Strategy: Assume we implemented HaltsOnZero, and show we can implement TestHalt.

We know implementing TestHalt is impossible, so we get a contradiction!

This is called a <u>reduction</u> from <u>TestHalt</u> to <u>HaltsOnZero</u>.

Proof: For contradiction, suppose we implemented HaltsOnZero. We implement TestHalt.

```
def TestHalt(P, x):
    def Helper(n):   # Helper(0) halts
                     # iff P(x) halts
        if n == 0:
            run P(x)
        return
    return HaltsOnZero(Helper)
```

If P(x) halts, Helper(0) halts, so HaltsOnZero(Helper) = 1. So, TestHalt(P, x) returns 1.

If $P(x)$ loops, $\mathrm{Helper}(0)$ loops, so $\mathrm{HaltsOnZero}(\mathrm{Helper}) = 0$. So, $\mathrm{TestHalt}(P, x)$ returns 0.

We implemented TestHalt, a contradiction! $\square$