CSC165 Week 12

Larry Yueli Zhang, November 25, 2014

Today’s outline

- ➔ countability
- ➔ Induction
- ➔ review for final exam

countability

compare the size of two sets

X = {1, 2, 3}

Y = {“one”, “two”, “three”}

how do their sizes compare?

$|X| \qquad |Y|$

How about these two?

X = {natural numbers} # 0, 1, 2, 3, 4, 5, ...

Y = {even natural numbers} # 0, 2, 4, 6, ...

$|X| < |Y|$?

$|X| > |Y|$?

$|X| = |Y|$?

an real-life example

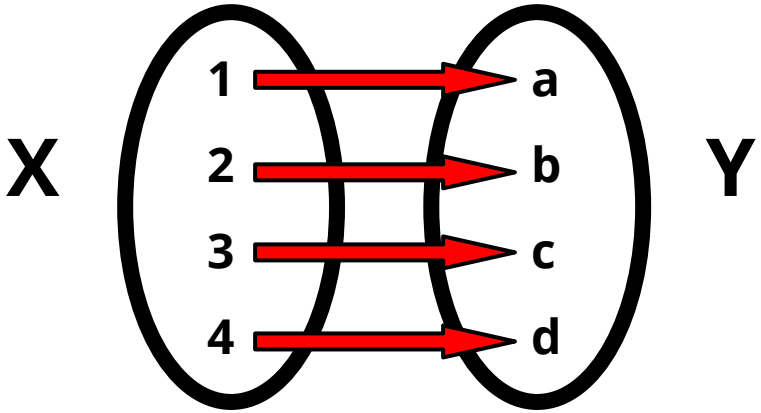
|X|: number of **coins**

|Y|: number of **coin tails**

they're the **same**, because

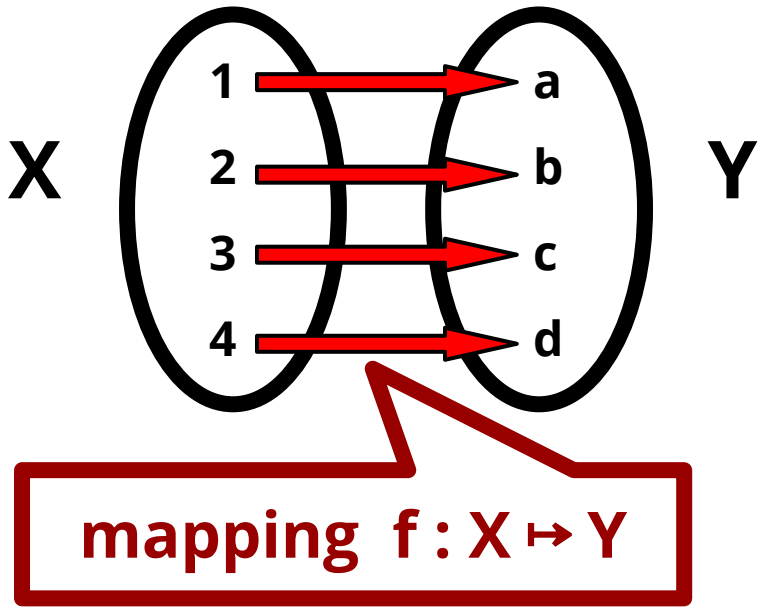
each coin has **one and only one** tail,

and each tail belongs to **one and only one** coin



If there is a mapping from X to Y like this, then we have |X| = |Y|

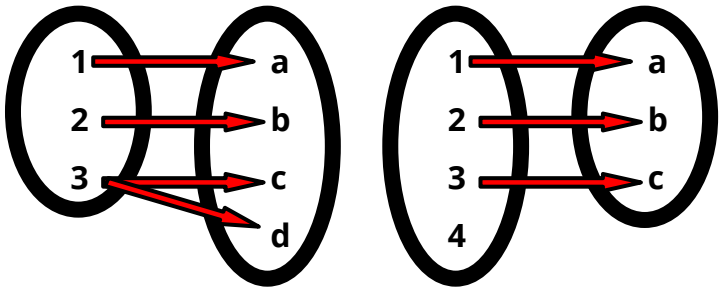
"like this", more precisely



What does $f : X \mapsto Y$ need to satisfy, mathematically?

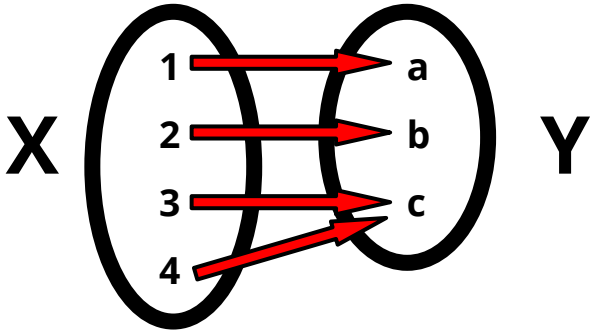
f is a **well defined function**, i.e., each **x** maps to a unique **y**.

So these don't happen:



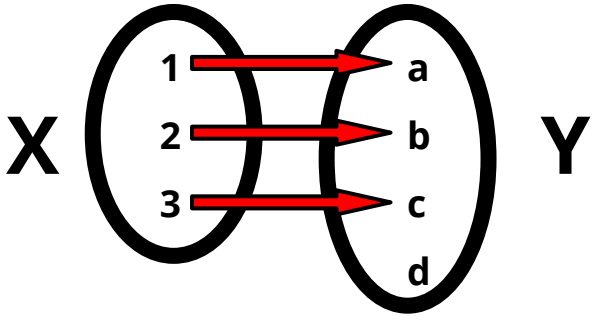
f is **"1-1"**, i.e., each **y** mapped to by at most one **x**.

So this doesn't happen



f is **"onto"**, i.e., every **y** has some **x** that maps to it.

So this doesn't happen:



put these together...

Given two sets **X** and **Y**, if there exists a **well defined** function $\mathbf{f : X \mapsto Y}$ that is **1-1** and **onto**, then $\mathbf{|X| = |Y|}$.

We can use this to compare the sizes of sets with infinitely many elements.

Key is: find an $\mathbf{f : X \mapsto Y}$

X = {natural numbers} # 0, 1, 2, 3, 4, 5, ...

Y = {even natural numbers} # 0, 2, 4, 6, ...

Prove: |X| = |Y|

Proof:

“countable”

when we **count**, we do: 0, 1, 2, 3, 4, ...
that’s basically **enumerating natural numbers**

so, we say the set $\mathbb{N}$ is countable

and, any set A that satisfies $|A| \leq |\mathbb{N}|$ is countable

$\mathbb{Z}$ (integers) is countable

$\mathbb{Q}$ (rational numbers) is countable

$\mathbb{R}$ (real numbers) is uncountable

countability and computability

How many Python functions could be written?

- ➔ the code of each Python function is basically a string of characters
- ➔ each (UTF-8) character corresponds a number between 0~255
- ➔ if we just concatenate all the numbers corresponding to all characters in the Python function, we get a unique natural number
- ➔ that is a 1-1 mapping: **Python functions ↦ N**

```
def hello(s):
    print(s)
    return
```

```
237624362342346234762346234274262342342344
```

so, we can enumerate all Python functions like:

$f_0, f_1, f_2, f_3, \ldots$

so, we can create the **table of function halting behaviours**, i.e., the table of return values of $Halt(f_i, f_j)$

f_i	$H(f_i, f_0)$	$H(f_i, f_1)$	$H(f_i, f_2)$	$H(f_i, f_3)$	...
f_0	T	F	T	F	
f_1	T	T	F	F	
f_2	T	F	T	T	
f_3	F	T	F	F	

⋮

this row describes the "**behaviour**" of f_3

all functions' behaviours are **enumerated** in this table

Now, construct a function f_x with "weird" behaviour

f_i	$H(f_i, f_0)$	$H(f_i, f_1)$	$H(f_i, f_2)$	$H(f_i, f_3)$
f_0	T	F	T	F
f_1	T	T	F	F
f_2	T	F	T	T
f_3	F	T	F	F

Take the diagonal and flip T/F $H(f_x, f_i) = \neg H(f_i, f_i) \quad i = 0, 1, 2, \ldots$

f_x	F	F	F	T

This row must be the same as one of the rows in the table, because f_x is just one of Python functions being enumerated.

Which row?

$$\forall i \in \mathbb{N}, \mathrm{Entry}(x, i) = \neg \mathrm{Entry}(i, i)$$

f_x does NOT exist!!

$$f_x : H(f_x, f_i) = \neg H(f_i, f_i) \quad i = 0, 1, 2, \ldots$$

i.e., $f_x(f_i)$ halts if $H(f_i, f_i)$ is False

$f_x(f_i)$ does not halt if $H(f_i, f_i)$ is True

f_x cannot be programmed

if it could be programmed, the code would look like...

```
def fx(fi):

    while halt(fi, fi):
        pass
    return 42
```

summary

- ➔ The argument we made has a name, it's called "**diagonalization**".
- ➔ Given any list of countably many of functions, we can always construct an **f_x** that is **outside** that list.
- ➔ that means the set of all possible functions is **uncountable**.
- ➔ Since we can only program **countably many** Python functions
- ➔ there are **uncountably many** functions that we **cannot program** in Python, or in any computer language