Today.

Today.

Farewell to modular arithmetic.

Today.

Farewell to modular arithmetic. Until the midterm.

Today.

Farewell to modular arithmetic. Until the midterm. And final.

Today.

Farewell to modular arithmetic. Until the midterm. And final.

Coutability and Uncountability.

Today.

Farewell to modular arithmetic. Until the midterm. And final.

Coutability and Uncountability.

Undecidability.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative,

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!

Allow for solving linear systems, discussing polynomials...

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!

Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$?

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$?

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$?

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh..

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer?

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure?

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25,

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1,

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1, 3.000001.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1, 3.000001. $\ldots$

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1, 3.000001. $\ldots$

For reals numbers we have the notion of limit, continuity,

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1, 3.000001. $\ldots$

For reals numbers we have the notion of limit, continuity, and derivative.......

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1, 3.000001. $\ldots$

For reals numbers we have the notion of limit, continuity, and derivative.......

....and Calculus.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1, 3.000001. $\ldots$

For reals numbers we have the notion of limit, continuity, and derivative.......

....and Calculus.

For modular arithmetic...

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1, 3.000001. $\dots$

For reals numbers we have the notion of limit, continuity, and derivative.......

....and Calculus.

For modular arithmetic...no Calculus.

Farewell (for now) to modular arithmetic...

Modular arithmetic modulo a prime.

Add, subtract, commutative, associative, inverses!
Allow for solving linear systems, discussing polynomials...

Why not modular arithmetic all the time?

$4 > 3$? Yes!

$4 > 3 \pmod 7$? Yes...maybe?

$-3 > 3 \pmod 7$? Uh oh.. $-3 = 4 \pmod 7$.

Another problem.

4 is close to 3.
But can you get closer? Sure. 3.5. Closer. Sure? 3.25, 3.1, 3.000001. $\ldots$

For reals numbers we have the notion of limit, continuity, and derivative.......

....and Calculus.

For modular arithmetic...no Calculus. Sad face!

Next up: how big is infinity.

Next up: how big is infinity.

- Countable
- Countably infinite.
- Enumeration

How big are the reals or the integers?

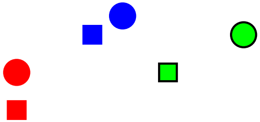
Infinite!

How big are the reals or the integers?

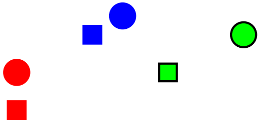
Infinite!

Is one bigger or smaller?

Same size?

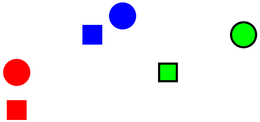


Same size?



Same number?

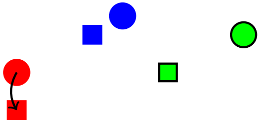
Same size?



Same number?

Make a function f : Circles $\to$ Squares.

Same size?

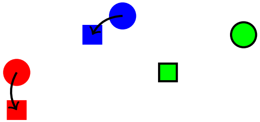


Same number?

Make a function f : Circles $\rightarrow$ Squares.

f(red circle) $=$ red square

Same size?



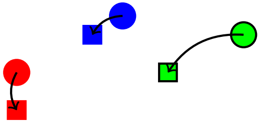
Same number?

Make a function f : Circles $\to$ Squares.

f(red circle) $=$ red square

f(blue circle) $=$ blue square

Same size?



Same number?

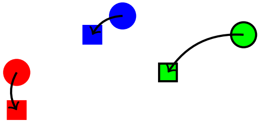
Make a function f : Circles $\to$ Squares.

f(red circle) $=$ red square

f(blue circle) $=$ blue square

f(circle with black border) $=$ square with black border

Same size?



Same number?

Make a function f : Circles $\rightarrow$ Squares.

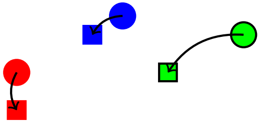
f(red circle) $=$ red square

f(blue circle) $=$ blue square

f(circle with black border) $=$ square with black border

One to one.

Same size?



Same number?

Make a function f : Circles $\rightarrow$ Squares.

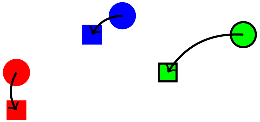
f(red circle) $=$ red square

f(blue circle) $=$ blue square

f(circle with black border) $=$ square with black border

One to one. Each circle mapped to different square.

Same size?



Same number?

Make a function f : Circles $\rightarrow$ Squares.

f(red circle) $=$ red square

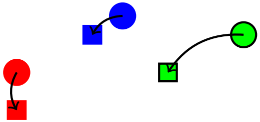
f(blue circle) $=$ blue square

f(circle with black border) $=$ square with black border

One to one. Each circle mapped to different square.

One to One: For all $x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.

Same size?



Same number?

Make a function f : Circles $\rightarrow$ Squares.

f(red circle) $=$ red square

f(blue circle) $=$ blue square

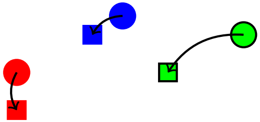
f(circle with black border) $=$ square with black border

One to one. Each circle mapped to different square.

One to One: For all $x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.

Onto.

Same size?



Same number?

Make a function f : Circles $\rightarrow$ Squares.

f(red circle) $=$ red square

f(blue circle) $=$ blue square

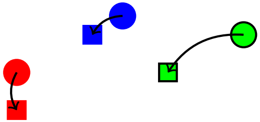
f(circle with black border) $=$ square with black border

One to one. Each circle mapped to different square.

One to One: For all $x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.

Onto. Each square mapped to from some circle .

Same size?



Same number?

Make a function f : Circles $\rightarrow$ Squares.

f(red circle) $=$ red square

f(blue circle) $=$ blue square

f(circle with black border) $=$ square with black border

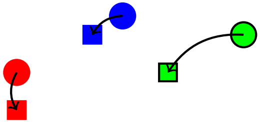
One to one. Each circle mapped to different square.

One to One: For all $x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.

Onto. Each square mapped to from some circle .

Onto: For all $s \in R$, $\exists c \in D, s = f(c)$.

Same size?



Same number?

Make a function f : Circles $\rightarrow$ Squares.

f(red circle) $=$ red square

f(blue circle) $=$ blue square

f(circle with black border) $=$ square with black border

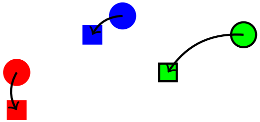
One to one. Each circle mapped to different square.

One to One: For all $x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.

Onto. Each square mapped to from some circle .

Onto: For all $s \in R$, $\exists c \in D, s = f(c)$.

Same size?



Same number?
Make a function f : Circles $\rightarrow$ Squares.
f(red circle) $=$ red square
f(blue circle) $=$ blue square
f(circle with black border) $=$ square with black border
One to one. Each circle mapped to different square.
One to One: For all $x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.
Onto. Each square mapped to from some circle .
Onto: For all $s \in R$, $\exists c \in D, s = f(c)$.

Isomorphism principle: If there is $f : D \rightarrow R$ that is one to one and onto, then, $|D| = |R|$.

Isomorphism principle.

Given a function, $f : D \rightarrow R$.

Isomorphism principle.

Given a function, $f : D \rightarrow R$.
One to One:

Isomorphism principle.

Given a function, $f : D \to R$.
One to One:
For all $\forall x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.

Isomorphism principle.

Given a function, $f : D \rightarrow R$.

One to One:

For all $\forall x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.

or

Isomorphism principle.

Given a function, $f : D \to R$.
One to One:
For all $\forall x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.
or
$\forall x, y \in D$, $f(x) = f(y) \implies x = y$.

Isomorphism principle.

Given a function, $f : D \to R$.
One to One:
For all $\forall x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.
or
$\forall x, y \in D$, $f(x) = f(y) \implies x = y$.

Isomorphism principle.

Given a function, $f : D \to R$.
One to One:
For all $\forall x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.
or
$\forall x, y \in D$, $f(x) = f(y) \implies x = y$.

Onto: For all $y \in R$, $\exists x \in D, y = f(x)$.

Isomorphism principle.

Given a function, $f : D \to R$.
One to One:
For all $\forall x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.
or
$\forall x, y \in D$, $f(x) = f(y) \implies x = y$.

Onto: For all $y \in R$, $\exists x \in D, y = f(x)$.

$f(\cdot)$ is a **bijection** if it is one to one and onto.

Isomorphism principle.

Given a function, $f : D \to R$.

One to One:

For all $\forall x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.

or

$\forall x, y \in D$, $f(x) = f(y) \implies x = y$.

Onto: For all $y \in R$, $\exists x \in D, y = f(x)$.

$f(\cdot)$ is a **bijection** if it is one to one and onto.

Isomorphism principle:

Isomorphism principle.

Given a function, $f : D \to R$.
One to One:
For all $\forall x, y \in D$, $x \neq y \implies f(x) \neq f(y)$.
or
$\forall x, y \in D$, $f(x) = f(y) \implies x = y$.

Onto: For all $y \in R$, $\exists x \in D, y = f(x)$.

$f(\cdot)$ is a **bijection** if it is one to one and onto.

Isomorphism principle:
If there is a bijection $f : D \to R$ then $|D| = |R|$.

Countable.

How to count?

Countable.

How to count?

0,

Countable.

How to count?

0, 1,

Countable.

How to count?

0, 1, 2,

Countable.

How to count?

0, 1, 2, 3,

Countable.

How to count?

$0, 1, 2, 3, \ldots$

Countable.

How to count?

0, 1, 2, 3, $\ldots$

The Counting numbers.

Countable.

How to count?

0, 1, 2, 3, $\ldots$

The Counting numbers.
The natural numbers! N

Countable.

How to count?

0, 1, 2, 3, ...

The Counting numbers.
The natural numbers! N

Definition: S is **countable** if there is a bijection between S and some subset of N.

Countable.

How to count?

0, 1, 2, 3, $\ldots$

The Counting numbers.
The natural numbers! N

Definition: S is **countable** if there is a bijection between S and some subset of N.

If the subset of N is finite, S has finite **cardinality**.

Countable.

How to count?

0, 1, 2, 3, ...

The Counting numbers.
The natural numbers! N

Definition: S is **countable** if there is a bijection between S and some subset of N.

If the subset of N is finite, S has finite **cardinality**.

If the subset of N is infinite, S is **countably infinite**.

Where’s 0?

Which is bigger?

Where’s 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. 0,

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1,$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2,$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3,$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1,$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2,$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3,$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0,1,2,3,\ldots.$

Positive integers. $1,2,3,\ldots.$

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n,

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$,

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z)$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z) = (n + 1) - 1$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z) = (n + 1) - 1 = n$.

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z) = (n + 1) - 1 = n$.
Onto for $\mathbb{N}$

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z) = (n + 1) - 1 = n$.
Onto for $\mathbb{N}$

Bijection!

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z) = (n + 1) - 1 = n$.
Onto for $\mathbb{N}$

Bijection! $\implies |\mathbb{Z}^+| = |\mathbb{N}|$.

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z) = (n+1) - 1 = n$.
Onto for $\mathbb{N}$

Bijection! $\implies |\mathbb{Z}^+| = |\mathbb{N}|$.

But.. but

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z) = (n+1) - 1 = n$.
Onto for $\mathbb{N}$

Bijection! $\implies |\mathbb{Z}^+| = |\mathbb{N}|$.

But.. but Where's zero?

Where's 0?

Which is bigger?
The positive integers, $\mathbb{Z}^+$, or the natural numbers, $\mathbb{N}$.

Natural numbers. $0, 1, 2, 3, \ldots$.

Positive integers. $1, 2, 3, \ldots$.

Where's 0?

More natural numbers!

Consider $f(z) = z - 1$.

For any two $z_1 \neq z_2 \implies z_1 - 1 \neq z_2 - 1 \implies f(z_1) \neq f(z_2)$.
One to one!

For any natural number n, for $z = n + 1$, $f(z) = (n+1) - 1 = n$.
Onto for $\mathbb{N}$

Bijection! $\implies |\mathbb{Z}^+| = |\mathbb{N}|$.

But.. but Where's zero? "Comes from 1."

A bijection is a bijection.

A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.

A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.
$f(n) = n + 1$.

A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.
$f(n) = n+1$. $0 \to 1$,

A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.
$f(n) = n+1$. $0 \to 1, 1 \to 2,$

A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.
$f(n) = n+1$. $0 \rightarrow 1, 1 \rightarrow 2, \ldots$

A bijection is a bijection.

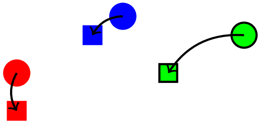
Notice that there is a bijection between N and Z^+ as well.
$f(n) = n+1$. $0 \to 1, 1 \to 2, \ldots$

Bijection from A to $B \implies$ a bijection from B to A.

A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.
$f(n) = n+1$. $0 \to 1, 1 \to 2, \ldots$

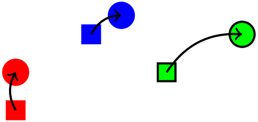
Bijection from A to $B \implies$ a bijection from B to A.



A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.
$f(n) = n + 1$. $0 \to 1, 1 \to 2, \ldots$

Bijection from A to $B \implies$ a bijection from B to A.

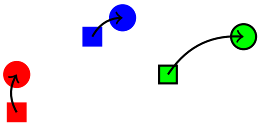


Inverse function!

A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.
$f(n) = n+1$. $0 \to 1, 1 \to 2, \ldots$

Bijection from A to $B \implies$ a bijection from B to A.



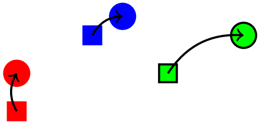
Inverse function!

Can prove equivalence either way.

A bijection is a bijection.

Notice that there is a bijection between N and Z^+ as well.
$f(n) = n+1$. $0 \rightarrow 1, 1 \rightarrow 2, \ldots$

Bijection from A to B $\implies$ a bijection from B to A.



Inverse function!

Can prove equivalence either way.
Bijection to or from natural numbers implies countably infinite.

More large sets.

E - Even natural numbers?

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

Onto:

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

Onto: $\forall e \in E$, $f(e/2) = e$.

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

Onto: $\forall e \in E$, $f(e/2) = e$. $e/2$ is natural since e is even

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

Onto: $\forall e \in E$, $f(e/2) = e$. $e/2$ is natural since e is even
One-to-one:

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

Onto: $\forall e \in E$, $f(e/2) = e$. $e/2$ is natural since e is even
One-to-one: $\forall x, y \in N, x \neq y \implies 2x \neq 2y$.

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

Onto: $\forall e \in E$, $f(e/2) = e$. $e/2$ is natural since e is even
One-to-one: $\forall x, y \in N, x \neq y \implies 2x \neq 2y. \equiv f(x) \neq f(y)$

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

Onto: $\forall e \in E$, $f(e/2) = e$. $e/2$ is natural since e is even
One-to-one: $\forall x, y \in N, x \neq y \implies 2x \neq 2y. \equiv f(x) \neq f(y)$

Evens are countably infinite.

More large sets.

E - Even natural numbers?

$f : \mathbb{N} \to E$.

$f(n) \to 2n$.

Onto: $\forall e \in E$, $f(e/2) = e$. $e/2$ is natural since e is even
One-to-one: $\forall x, y \in N, x \neq y \implies 2x \neq 2y. \equiv f(x) \neq f(y)$

Evens are countably infinite.
Evens are same size as all natural numbers.

All integers?

What about Integers, Z?

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

All integers?

What about Integers, Z?
Define $f : N \rightarrow Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$

All integers?

What about Integers, Z?
Define $f : N \rightarrow Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$

All integers?

What about Integers, Z?
Define $f : N \rightarrow Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,
then $x/2 \neq y/2$

All integers?

What about Integers, Z?
Define $f : N \rightarrow Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,
then $x/2 \neq y/2 \implies f(x) \neq f(y)$

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,
then $x/2 \neq y/2 \implies f(x) \neq f(y)$
$\ldots$.

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,
then $x/2 \neq y/2 \implies f(x) \neq f(y)$
$\ldots$.

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,
then $x/2 \neq y/2 \implies f(x) \neq f(y)$
$\ldots$.

Onto: For any $z \in Z$,

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,
then $x/2 \neq y/2 \implies f(x) \neq f(y)$
$\ldots$.

Onto: For any $z \in Z$,
if $z \geq 0$, $f(2z) = z$ and $2z \in N$.

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,
then $x/2 \neq y/2 \implies f(x) \neq f(y)$
$\ldots$.

Onto: For any $z \in Z$,
if $z \geq 0$, $f(2z) = z$ and $2z \in N$.
if $z < 0$, $f(2|z| - 1) = z$ and $2|z| + 1 \in N$.

All integers?

What about Integers, Z?
Define $f : N \to Z$.

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

One-to-one: For $x \neq y$
if x is even and y is odd,
then $f(x)$ is nonnegative and $f(y)$ is negative $\implies f(x) \neq f(y)$
if x is even and y is even,
then $x/2 \neq y/2 \implies f(x) \neq f(y)$
$\ldots$.

Onto: For any $z \in Z$,
if $z \geq 0$, $f(2z) = z$ and $2z \in N$.
if $z < 0$, $f(2|z|-1) = z$ and $2|z|+1 \in N$.

Integers and naturals have same size!

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1
2	1

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1
2	1
3	-2

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1
2	1
3	-2
4	2

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1
2	1
3	-2
4	2
...	...

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1
2	1
3	-2
4	2
$\dots$	$\dots$

Notice that: A listing “is” a bijection with a subset of natural numbers.

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1
2	1
3	-2
4	2
...	...

Notice that: A listing "is" a bijection with a subset of natural numbers.
Function $\equiv$ "Position in list."

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1
2	1
3	-2
4	2
...	...

Notice that: A listing "is" a bijection with a subset of natural numbers.
Function $\equiv$ "Position in list."
If finite: bijection with $\{0, \ldots, |S|-1\}$

Listings..

$$f(n) = \begin{cases} n/2 & \text{if } n \text{ even} \\ -(n+1)/2 & \text{if n odd.} \end{cases}$$

Another View:

n	$f(n)$
0	0
1	-1
2	1
3	-2
4	2
...	...

Notice that: A listing "is" a bijection with a subset of natural numbers.
Function $\equiv$ "Position in list."
If finite: bijection with $\{0,\ldots,|S|-1\}$
If infinite: bijection with N.

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
...

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
...
Any element x of S has *specific, finite* position in list.

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0,$

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1,$

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1,$

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2,$

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2,$

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
...
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2, \ldots..\}$

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\dots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2, \dots..\}$
$Z = \{\{0, 1, 2, \dots, \}$

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2, \ldots..\}$
$Z = \{\{0, 1, 2, \ldots,\}$ and then $\{-1, -2, \ldots\}\}$

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2, \ldots..\}$
$Z = \{\{0, 1, 2, \ldots,\}$ and then $\{-1, -2, \ldots\}\}$

When do you get to -1?

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2, \ldots..\}$
$Z = \{\{0, 1, 2, \ldots,\}$ and then $\{-1, -2, \ldots\}\}$

When do you get to -1? at infinity?

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of *S*",
"Output next element of *S*"
$\dots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2, \dots..\}$
$Z = \{\{0, 1, 2, \dots,\}$ and then $\{-1, -2, \dots\}\}$

When do you get to -1? at infinity?

Need to be careful.

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2, \ldots..\}$
$Z = \{\{0, 1, 2, \ldots, \}$ and then $\{-1, -2, \ldots\}\}$

When do you get to -1? at infinity?

Need to be careful.

61A

Enumerability $\equiv$ countability.

Enumerating (listing) a set implies that it is countable.

"Output element of S",
"Output next element of S"
$\ldots$
Any element x of S has *specific, finite* position in list.
$Z = \{0, 1, -1, 2, -2, \ldots..\}$
$Z = \{\{0, 1, 2, \ldots,\}$ and then $\{-1, -2, \ldots\}\}$

When do you get to -1? at infinity?

Need to be careful.

61A —- streams!

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,
output only if $x \in T$.

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,
output only if $x \in T$.

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,
output only if $x \in T$.

Implications:

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,
output only if $x \in T$.

Implications:
Z^+ is countable.

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,
output only if $x \in T$.

Implications:
Z^+ is countable.
It is infinite since the list goes on.

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,
output only if $x \in T$.

Implications:
Z^+ is countable.
It is infinite since the list goes on.
There is a bijection with the natural numbers.

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,
output only if $x \in T$.

Implications:
Z^+ is countable.
It is infinite since the list goes on.
There is a bijection with the natural numbers.
So it is countably infinite.

Countably infinite subsets.

Enumerating a set implies countable.
Corollary: Any subset T of a countable set S is countable.

Enumerate T as follows:
Get next element, x, of S,
output only if $x \in T$.

Implications:
Z^+ is countable.
It is infinite since the list goes on.
There is a bijection with the natural numbers.
So it is countably infinite.

All countably infinite sets have the same cardinality.

Enumeration example.

All binary strings.

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi,$

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0,$

Enumeration example.

All binary strings.
$B = \{0, 1\}^*$.

$B = \{\phi, 0, 1,$

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00,$

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00, 01, 10, 11,$

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.
ϕ is empty string.

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.
ϕ is empty string.

For any string, it appears at some position in the list.

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.
ϕ is empty string.

For any string, it appears at some position in the list.
If n bits, it will appear before position 2^{n+1}.

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.
ϕ is empty string.

For any string, it appears at some position in the list.
If n bits, it will appear before position 2^{n+1}.

Should be careful here.

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.
ϕ is empty string.

For any string, it appears at some position in the list.
If n bits, it will appear before position 2^{n+1}.

Should be careful here.

$B = \{\phi; , 0, 00, 000, 0000, ...\}$

Enumeration example.

All binary strings.
$B = \{0,1\}^*$.

$B = \{\phi, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\}$.
ϕ is empty string.

For any string, it appears at some position in the list.
If n bits, it will appear before position 2^{n+1}.

Should be careful here.

$B = \{\phi; , 0, 00, 000, 0000, ...\}$
Never get to 1.

More fractions?

Enumerate the rational numbers in order...

More fractions?

Enumerate the rational numbers in order...

$0, \ldots, 1/2, ..$

More fractions?

Enumerate the rational numbers in order...

$0, \ldots, 1/2, ..$

Where is $1/2$ in list?

More fractions?

Enumerate the rational numbers in order...

$0, \ldots, 1/2, ..$

Where is $1/2$ in list?

After $1/3$, which is after $1/4$, which is after $1/5$...

More fractions?

Enumerate the rational numbers in order...

$0, \ldots, 1/2, ..$

Where is $1/2$ in list?

After $1/3$, which is after $1/4$, which is after $1/5$...

A thing about fractions:

More fractions?

Enumerate the rational numbers in order...

$0, \ldots, 1/2, ..$

Where is $1/2$ in list?

After $1/3$, which is after $1/4$, which is after $1/5$...

A thing about fractions:
any two fractions has another fraction between it.

More fractions?

Enumerate the rational numbers in order...

$0, \ldots, 1/2, ..$

Where is $1/2$ in list?

After $1/3$, which is after $1/4$, which is after $1/5$...

A thing about fractions:
any two fractions has another fraction between it.

Can't even get to "next" fraction!

More fractions?

Enumerate the rational numbers in order...

$0, \ldots, 1/2, ..$

Where is $1/2$ in list?

After $1/3$, which is after $1/4$, which is after $1/5$...

A thing about fractions:
any two fractions has another fraction between it.

Can't even get to "next" fraction!

Can't list in "order".

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets S_1 and S_2,

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets S_1 and S_2,
then $S_1 \times S_2$

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets S_1 and S_2,
then $S_1 \times S_2$
has size $|S_1| \times |S_2|$.

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets S_1 and S_2,
then $S_1 \times S_2$
has size $|S_1| \times |S_2|$.

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets S_1 and S_2,
then $S_1 \times S_2$
has size $|S_1| \times |S_2|$.

So, $N \times N$ is countably infinite

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets S_1 and S_2,
then $S_1 \times S_2$
has size $|S_1| \times |S_2|$.

So, $N \times N$ is countably infinite squared

Pairs of natural numbers.

Consider pairs of natural numbers: $N \times N$
E.g.: $(1,2)$, $(100,30)$, etc.

For finite sets S_1 and S_2,
then $S_1 \times S_2$
has size $|S_1| \times |S_2|$.

So, $N \times N$ is countably infinite squared ???

Pairs of natural numbers.

Enumerate in list:

Pairs of natural numbers.

Enumerate in list:
$(0,0)$,

Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),$

Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),$

Pairs of natural numbers.

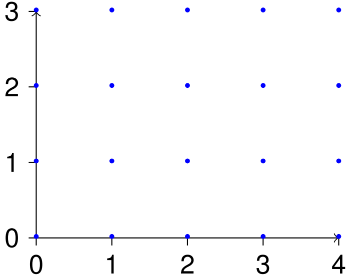
Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),$

Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),$

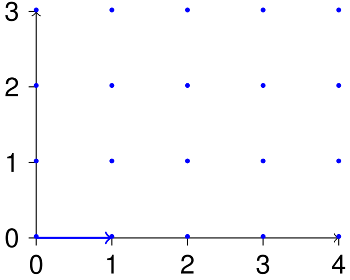
Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



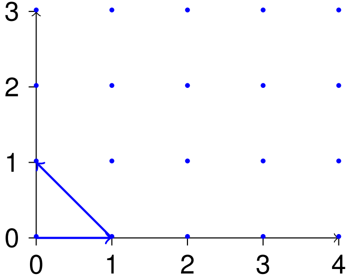
Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



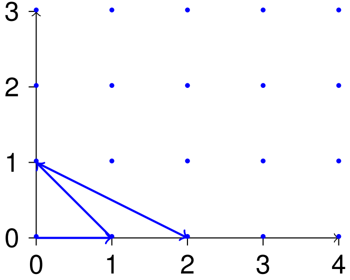
Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



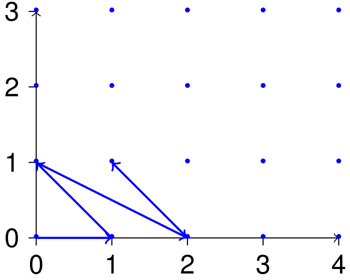
Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



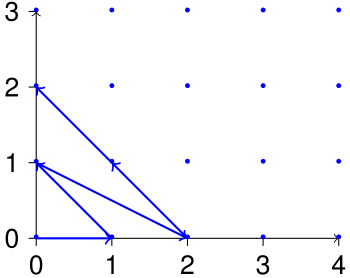
Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



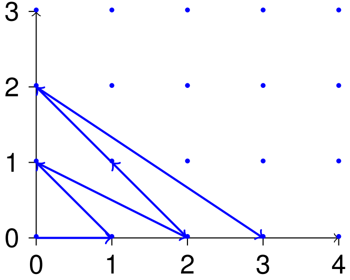
Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



Pairs of natural numbers.

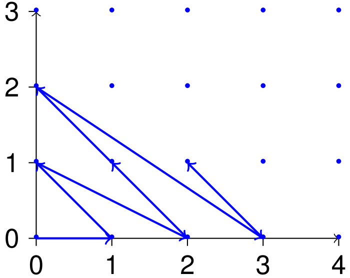
Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



Pairs of natural numbers.

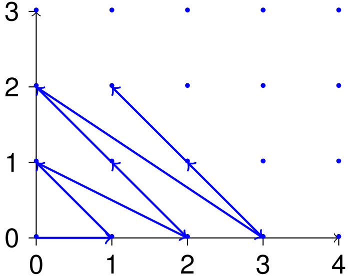
Enumerate in list:

$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



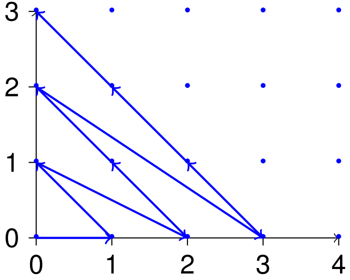
Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



Pairs of natural numbers.

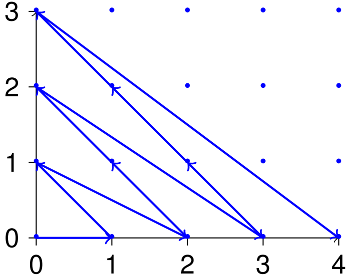
Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



Pairs of natural numbers.

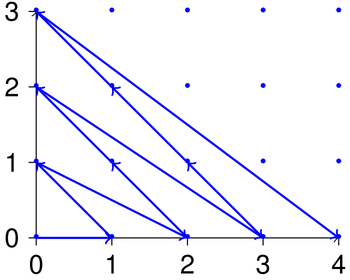
Enumerate in list:

$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



Pairs of natural numbers.

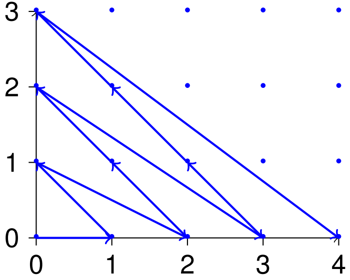
Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



The pair (a,b), is in first $\approx (a+b+1)(a+b)/2$ elements of list!

Pairs of natural numbers.

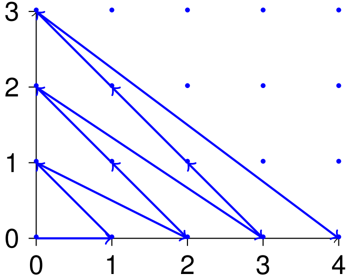
Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



The pair (a,b), is in first $\approx (a+b+1)(a+b)/2$ elements of list!
(i.e., "triangle").

Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$

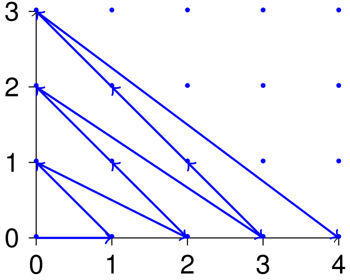


The pair (a,b), is in first $\approx (a+b+1)(a+b)/2$ elements of list!
(i.e., “triangle”).

Countably infinite.

Pairs of natural numbers.

Enumerate in list:
$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),\ldots\ldots$



The pair (a,b), is in first $\approx (a+b+1)(a+b)/2$ elements of list!
(i.e., "triangle").

Countably infinite.

Same size as the natural numbers!!

Rationals?

Positive rational number.

Rationals?

Positive rational number.
Lowest terms: a/b

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a,b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a,b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable.

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a,b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative ???

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative ??? No!

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative ??? No!

Repeatedly and alternatively take one from each list.

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative ??? No!

Repeatedly and alternatively take one from each list.
    Interleave Streams in 61A

Rationals?

Positive rational number.
Lowest terms: a/b
$a, b \in N$
with $gcd(a, b) = 1$.

Infinite subset of $N \times N$.

Countably infinite!

All rational numbers?

Negative rationals are countable. (Same size as positive rationals.)

Put all rational numbers in a list.

First negative, then nonegative ??? No!

Repeatedly and alternatively take one from each list.
Interleave Streams in 61A

The rationals are countably infinite.

Real numbers..

Real numbers are same size as integers?

The reals.

Are the set of reals countable?

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000...

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162...

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441...

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558...

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558... $1-1/e$

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558... $1-1/e$
.345212312...

The reals.

Are the set of reals countable?

Lets consider the reals $[0, 1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558... $1 - 1/e$
.345212312... Some real number

The reals.

Are the set of reals countable?

Lets consider the reals $[0,1]$.

Each real has a decimal representation.
.500000000... $(1/2)$
.785398162... $\pi/4$
.367879441... $1/e$
.632120558... $1-1/e$
.345212312... Some real number

Diagonalization.

If countable, there a listing, L contains all reals.

Diagonalization.

If countable, there a listing, L contains all reals. For example

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number:

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .7

Diagonalization.

If countable, there a listing, *L* contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .77

Diagonalization.

If countable, there a listing, *L* contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct “diagonal” number: .776

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .7767

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .77677

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: .77677 . . .

Diagonal Number:

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
⋮

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!

Diagonalization.

If countable, there a listing, *L* contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit *i* is 7 if number *i*'s *i*th digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonal number is real.

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonal number is real.

Contradiction!

Diagonalization.

If countable, there a listing, L contains all reals. For example

0: .500000000...
1: .785398162...
2: .367879441...
3: .632120558...
4: .345212312...
$\vdots$

Construct "diagonal" number: $.77677\ldots$

Diagonal Number: Digit i is 7 if number i's ith digit is not 7
and 6 otherwise.

Diagonal number for a list differs from every number in list!
Diagonal number not in list.

Diagonal number is real.

Contradiction!

Subset $[0,1]$ is not countable!!

All reals?

Subset $[0, 1]$ is not countable!!

All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?

All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
No.

All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
No.

Any subset of a countable set is countable.

All reals?

Subset $[0, 1]$ is not countable!!

What about all reals?
No.

Any subset of a countable set is countable.

If reals are countable then so is $[0, 1]$.

Diagonalization.

1. Assume that a set S can be enumerated.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list
   $\implies$ t is not in the list.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list
   $\implies$ t is not in the list.
5. Show that t is in S.

Diagonalization.

1. Assume that a set S can be enumerated.
2. Consider an arbitrary list of all the elements of S.
3. Use the diagonal from the list to construct a new element t.
4. Show that t is different from all elements in the list
   $\implies$ t is not in the list.
5. Show that t is in S.
6. Contradiction.

Another diagonalization.

The set of all subsets of N.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.

Another diagonalization.

The set of all subsets of *N*.

Example subsets of *N*: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, *L*, that contains all subsets of *N*.

Define a diagonal set, *D*:
If *i*th set in *L* does not contain *i*, $i \in D$.
otherwise $i \notin D$.

D is different from *i*th set in *L* for every *i*.
$\implies$ *D* is not in the listing.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.
$\implies$ D is not in the listing.

D is a subset of N.

Another diagonalization.

The set of all subsets of *N*.

Example subsets of *N*: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, *L*, that contains all subsets of *N*.

Define a diagonal set, *D*:
If *i*th set in *L* does not contain *i*, $i \in D$.
otherwise $i \not\in D$.

D is different from *i*th set in *L* for every *i*.
$\implies$ *D* is not in the listing.

D is a subset of *N*.

L does not contain all subsets of *N*.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.
$\implies$ D is not in the listing.

D is a subset of N.

L does not contain all subsets of N.

Contradiction.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0, \ldots, 7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.
$\implies$ D is not in the listing.

D is a subset of N.

L does not contain all subsets of N.

Contradiction.

Theorem: The set of all subsets of N is not countable.

Another diagonalization.

The set of all subsets of N.

Example subsets of N: $\{0\}$, $\{0,\ldots,7\}$,
evens, odds, primes,

Assume is countable.

There is a listing, L, that contains all subsets of N.

Define a diagonal set, D:
If ith set in L does not contain i, $i \in D$.
otherwise $i \notin D$.

D is different from ith set in L for every i.
$\implies$ D is not in the listing.

D is a subset of N.

L does not contain all subsets of N.

Contradiction.

Theorem: The set of all subsets of N is not countable.
(The set of all subsets of S, is the **powerset** of N.)

Diagonalize Natural Number.

Natural numbers have a listing, L.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

D is a natural number...

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

D is a natural number... Not.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

D is a natural number... Not.

Any natural number has a finite number of digits.

Diagonalize Natural Number.

Natural numbers have a listing, L.

Make a diagonal number, D:
differ from ith element of L in ith digit.

Differs from all elements of listing.

D is a natural number... Not.

Any natural number has a finite number of digits.

“Construction” requires an infinite number of digits.

The Continuum hypothesis.

There is no set with cardinality between the naturals and the reals.

The Continuum hypothesis.

There is no set with cardinality between the naturals and the reals.

First of Hilbert's problems!

Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

Cardinalities of uncountable sets?

Cardinality of $[0, 1]$ smaller than all the reals?

$f : R^+ \rightarrow [0, 1]$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$,

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^{+} \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ a division

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ a division $\implies f(x) \neq f(y)$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ a division $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't,

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ a division $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ a division $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \rightarrow [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \leq x \leq 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\Longrightarrow f(x) \neq f(y)$.
If neither in $[0,1/2]$ a division $\Longrightarrow f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\Longrightarrow f(x) \neq f(y)$.
Bijection!

Cardinalities of uncountable sets?

Cardinality of $[0,1]$ smaller than all the reals?

$f : R^+ \to [0,1]$.

$$f(x) = \begin{cases} x + \frac{1}{2} & 0 \le x \le 1/2 \\ \frac{1}{4x} & x > 1/2 \end{cases}$$

One to one. $x \neq y$
If both in $[0,1/2]$, a shift $\implies f(x) \neq f(y)$.
If neither in $[0,1/2]$ a division $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't, different ranges $\implies f(x) \neq f(y)$.
Bijection!

$[0,1]$ is same cardinality as nonnegative reals!

Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.

Generalized Continuum hypothesis.

There is no infinite set whose cardinality is between the cardinality of an infinite set and its power set.

The powerset of a set is the set of all subsets.

Resolution of hypothesis?

Resolution of hypothesis?

Gödel. 1940.
Can’t use math!

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.

Can a program refer to a program?

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.

Can a program refer to a program?

Can a program refer to itself?

Resolution of hypothesis?

Gödel. 1940.
Can't use math!
If math doesn't contain a contradiction.

This statement is a lie.

Is the statement above true?

The barber shaves every person who does not shave themselves.

Who shaves the barber?

Self reference.

Can a program refer to a program?

Can a program refer to itself?

Uh oh....

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$. Definable set.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$. Definable set.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$. Definable set.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$. Definable set.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$. Definable set.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Oops!

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$. Definable set.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Oops! Not Definable.

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$. Definable set.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Oops! Not Definable.

What type of object is a set that contain sets?

Russell's Paradox.

Naive Set Theory: Any definable collection is a set.

$$\exists y \forall x (x \in y \iff P(x)) \tag{1}$$

y is the set of elements that satifies the proposition $P(x)$.

$P(x) = x \notin x$. Definable set.

There exists a y that satisfies statement 1 for $P(\cdot)$.

Take $x = y$.

$$y \in y \iff y \notin y.$$

Oops! Not Definable.

What type of object is a set that contain sets?

Axioms changed.

Changing Axioms?

Goedel:
Any set of axioms is either

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine...

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..
Dangerous work?

Changing Axioms?

Goedel:
Any set of axioms is either
inconsistent (can prove false statements) or
incomplete (true statements cannot be proven.)

Concrete example:
Continuum hypothesis: “no cardinatity between reals and naturals.”
Continuum hypothesis not disprovable in ZFC
(Goedel 1940.)

Continuum hypothesis not provable.
(Cohen 1963: only Fields medal in logic)

BTW:
Cantor ..bipolar disorder..
Goedel ..starved himself out of fear of being poisoned..
Russell .. was fine.....but for ...two schizophrenic children..
Dangerous work?

See Logicomix by Doxiaidis, Papadimitriou (professor here),
Papadatos, Di Donna.

Is it actually useful?

Write me a program checker!

Is it actually useful?

Write me a program checker!

Check that the compiler works!

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
    P - program

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
 P - program
 I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine!

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
   P - program
   I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Program can be an input to a program.

Is it actually useful?

Write me a program checker!

Check that the compiler works!

How about.. Check that the compiler terminates on a certain input.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Notice:
Need a computer
...with the notion of a stored program!!!!
(not an adding machine! not a person and an adding machine.)

Program is a text string.
Text string can be an input to a program.
Program can be an input to a program.

Implementing HALT.

Implementing HALT.

$HALT(P, I)$

Implementing HALT.

$HALT(P, I)$

P - program

Implementing HALT.

$HALT(P, I)$
P - program
I - input.

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

Implementing HALT.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

How long do you wait?

Implementing HALT.

$HALT(P, I)$

P - program

I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Run P on I and check!

How long do you wait?

Something about infinity here, maybe?

Halt does not exist.

Halt does not exist.

$HALT(P, I)$

Halt does not exist.

$HALT(P, I)$

P - program

Halt does not exist.

$HALT(P, I)$

P - program

I - input.

Halt does not exist.

$HALT(P, I)$

P - program

I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes!

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No!

Halt does not exist.

$HALT(P, I)$
    P - program
    I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No!

Halt does not exist.

$HALT(P, I)$
 P - program
 I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No!

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No!

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes!

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! ..

Halt does not exist.

$HALT(P, I)$
 P - program
 I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.
(B) Fermat's Theorem.

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.
(B) Fermat's Theorem.
(C) Diagonalization.

Halt does not exist.

$HALT(P, I)$
  P - program
  I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.
(B) Fermat's Theorem.
(C) Diagonalization.
(D) Professor is just strange.

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.
(B) Fermat's Theorem.
(C) Diagonalization.
(D) Professor is just strange.

(C).

Halt does not exist.

$HALT(P, I)$
P - program
I - input.

Determines if $P(I)$ (P run on I) halts or loops forever.

Theorem: There is no program HALT.

Proof: Yes! No! Yes! No! No! Yes! No! Yes! .. $\square$

What is he talking about?
(A) He is confused.
(B) Fermat's Theorem.
(C) Diagonalization.
(D) Professor is just strange.

(C). Maybe (D).

Halt and Turing.

Proof:

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)

1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that “is” the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that “is” the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that “is” the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction.

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist!

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot,\cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist! $\square$

Halt and Turing.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Assumption: there is a program HALT.
There is text that "is" the program HALT.
There is text that is the program Turing.
Can run Turing on Turing!

Does Turing(Turing) halt?

Turing(Turing) halts
$\implies$ then HALTS(Turing, Turing) $=$ halts
$\implies$ Turing(Turing) loops forever.

Turing(Turing) loops forever
$\implies$ then HALTS(Turing, Turing) $\neq$ halts
$\implies$ Turing(Turing) halts.

Contradiction. Program HALT does not exist! $\square$
Questions?

Another view of proof: diagonalization.

Any program is a fixed length string.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & \text{H} & \text{H} & \text{L} & \cdots \\
P_2 & \text{L} & \text{L} & \text{H} & \cdots \\
P_3 & \text{L} & \text{H} & \text{H} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$
\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & \mathrm{H} & \mathrm{H} & \mathrm{L} & \cdots \\
P_2 & \mathrm{L} & \mathrm{L} & \mathrm{H} & \cdots \\
P_3 & \mathrm{L} & \mathrm{H} & \mathrm{H} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Halt - diagonal.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$
\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & \color{blue}{\text{H}} & \text{H} & \text{L} & \cdots \\
P_2 & \text{L} & \color{blue}{\text{L}} & \text{H} & \cdots \\
P_3 & \text{L} & \text{H} & \color{blue}{\text{H}} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Halt - diagonal.
Turing - is not Halt.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$
\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & \color{blue}{H} & H & L & \cdots \\
P_2 & L & \color{blue}{L} & H & \cdots \\
P_3 & L & H & \color{blue}{H} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

$$
\begin{array}{c|cccc}
 & P_1 & P_2 & P_3 & \cdots \\
\hline
 & & & & \\
P_1 & \mathrm{H} & \mathrm{H} & \mathrm{L} & \cdots \\
P_2 & \mathrm{L} & \mathrm{L} & \mathrm{H} & \cdots \\
P_3 & \mathrm{L} & \mathrm{H} & \mathrm{H} & \cdots \\
\vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.
Halt does not exist!

Another view of proof: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.
Program halts or not any input, which is a string.

	P_1	P_2	P_3	$\cdots$
P_1	H	H	L	$\cdots$
P_2	L	L	H	$\cdots$
P_3	L	H	H	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Halt - diagonal.
Turing - is not Halt.
and is different from every P_i on the diagonal.
Turing is not on list. Turing is not a program.
Turing can be constructed from Halt.
Halt does not exist! $\square$

Proof play by play.

Assumed HALT(P, I) existed.

Proof play by play.

Assumed $\text{HALT}(P, I)$ existed.

What is P?

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I?

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
  You have *Text* that is the program HALT(P, I).

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
  You have *Text* that is the program HALT(P, I).

Have ____ that is the program TURING.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have <u>Text</u> that is the program TURING.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have <u>Text</u> that is the program TURING.
Here it is!!

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
Turing(P)

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
Turing(P)
1. If HALT(P,P) ="halts", then go into an infinite loop.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
Turing(P)

1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
Turing(P)

1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Turing “diagonalizes” on list of program.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
Turing(P)

1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Turing “diagonalizes” on list of program.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
  You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
Turing(P)
1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Turing “diagonalizes” on list of program.
It is not a program!!!!

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
You have *Text* that is the program HALT(P, I).

Have Text that is the program TURING.
Here it is!!
Turing(P)

1. If HALT(P,P) =“halts”, then go into an infinite loop.
2. Otherwise, halt immediately.

Turing “diagonalizes” on list of program.
It is not a program!!!!
$\implies$ HALT is not a program.

Proof play by play.

Assumed HALT(P, I) existed.

What is P? Text.
What is I? Text.

What does it mean to have a program HALT(P, I).
  You have *Text* that is the program HALT(P, I).

Have <u>Text</u> that is the program TURING.
Here it is!!
Turing(P)

1. If HALT(P,P) ="halts", then go into an infinite loop.
2. Otherwise, halt immediately.

Turing "diagonalizes" on list of program.
It is not a program!!!!
$\implies$ HALT is not a program.

Questions?

We are so smart!

Wow, that was easy!

We are so smart!

Wow, that was easy!

We should be famous!

No computers for Turing!

In Turing's time.

No computers for Turing!

In Turing's time.

No computers.

No computers for Turing!

In Turing's time.

No computers.

Adding machines.

No computers for Turing!

In Turing's time.

No computers.

Adding machines.
e.g., Babbage (from table of logarithms) 1812.

No computers for Turing!

In Turing's time.

No computers.

Adding machines.
e.g., Babbage (from table of logarithms) 1812.

Concept of program as data wasn't really there.

Turing machine.

Turing machine.

A Turing machine.
– an (infinite) tape with characters

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ...

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing: AI,

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing: AI, self modifying code,

Turing machine.

A Turing machine.
– an (infinite) tape with characters
– be in a state, and read a character
– move left, right, and/or write a character.

Universal Turing machine
– an interpreter program for a Turing machine
– where the tape could be a description of a ... Turing machine!

Now that's a computer!

Turing: AI, self modifying code, learning...

Turing and computing.

Just a mathematician?

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won!

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won! Once anyway.

Turing and computing.

Just a mathematician?

“Wrote” a chess program.

Simulated the program by hand to play chess.

It won! Once anyway.

Involved with computing labs through the 40s.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is...

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages!

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number!

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! !

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! ! !

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! ! ! !

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! ! ! ! Same cardinality as...

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! ! ! ! Same cardinality as...Text.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! ! ! ! Same cardinality as...Text.
Today:

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! ! ! ! Same cardinality as...Text.
Today:Programs can be written in

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! ! ! ! Same cardinality as...Text.
Today:Programs can be written in ascii.

Church, Gödel and Turing.

Church proved an equivalent theorem. (Previously.)

Used λ calculus....which is... Lisp (Scheme)!!!
.. functional part. Scheme's lambda is calculus's λ!

Programming languages! javascript, ruby, python....

Gödel: Incompleteness theorem.

Any formal system either is inconsistent or incomplete.
Inconsistent: A false sentence can be proven.
Incomplete: There is no proof for some sentence in the system.

Along the way: "built" computers out of arithmetic.
Showed that every mathematical statement corresponds to
.... a natural number! ! ! ! Same cardinality as...Text.
Today:Programs can be written in ascii.

Computing on top of computing...

Computer, assembly code, programming language, browser, html, javascript..

Computing on top of computing...

Computer, assembly code, programming language, browser, html, javascript..

We can't get enough of building more Turing machines.

Undecidable problems.

Does a program, P, print “Hello World”?

Undecidable problems.

Does a program, P, print “Hello World”?
How?

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P?

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Undecidable problems.

Does a program, P, print “Hello World”?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?

Undecidable problems.

Does a program, P, print “Hello World”?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”

Undecidable problems.

Does a program, *P*, print “Hello World”?
How? What is *P*? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Undecidable problems.

Does a program, P, print “Hello World”?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?

Undecidable problems.

Does a program, P, print “Hello World”?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

Undecidable problems.

Does a program, P, print “Hello World”?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no.

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This "problem" is not undecidable.

Undecidable problems.

Does a program, P, print “Hello World”?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: “ $x^n + y^n = 1$?”
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This “problem” is not undecidable.

Undecidability for Diophantine set of equations

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This "problem" is not undecidable.

Undecidability for Diophantine set of equations
$\implies$ no program can take any set of integer equations and

Undecidable problems.

Does a program, P, print "Hello World"?
How? What is P? Text!!!!!!

Find exit points and add statement: **Print** "Hello World."

Can a set of notched tiles tile the infinite plane?
Proof: simulate a computer. Halts if finite.

Does a set of integer equations have a solution?
Example: " $x^n + y^n = 1$?"
Problem is undecidable.

Be careful!

Is there an integer solution to $x^n + y^n = 1$?
(Diophantine equation.)

The answer is yes or no. This "problem" is not undecidable.

Undecidability for Diophantine set of equations
$\Longrightarrow$ no program can take any set of integer equations and always corectly output whether it has an integer solution.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis:

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number. Math 54 doesn't really work.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.

  Almost dependent matrices.
- Seminal paper in mathematical biology.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person:

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
      Almost dependent matrices.
- Seminal paper in mathematical biology.
    Person: embryo is blob. Legs,

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
      Almost dependent matrices.
- Seminal paper in mathematical biology.
    Person: embryo is blob. Legs, arms,

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head....

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly:

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: blob.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: blob. Torso becomes striped.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: blob. Torso becomes striped.
  Developed chemical reaction-diffusion networks that break symmetry.

More about Alan Turing.

- Brilliant codebreaker during WWII, helped break German Enigma Code (which probably shortened war by 1 year).
- Seminal paper in numerical analysis: Condition number.
  Math 54 doesn't really work.
  Almost dependent matrices.
- Seminal paper in mathematical biology.
  Person: embryo is blob. Legs, arms, head.... How?
  Fly: blob. Torso becomes striped.
  Developed chemical reaction-diffusion networks that break symmetry.
- Imitation Game.

Turing: personal.

Tragic ending...

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- (possibly) suicided with cyanide at age 42 in 1954.

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- (possibly) suicided with cyanide at age 42 in 1954.
  (A bite from the apple....)

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- (possibly) suicided with cyanide at age 42 in 1954.
  (A bite from the apple....) accident?

Turing: personal.

Tragic ending...

- Arrested as a homosexual, (not particularly closeted)
- given choice of prison or (quackish) injections to eliminate sex drive;
- took injections.
- lost security clearance...
- suffered from depression;
- (possibly) suicided with cyanide at age 42 in 1954.
  (A bite from the apple....) accident?
- British Government apologized (2009) and pardoned (2013).

Back to technical..

This statement is a lie.

Back to technical..

This statement is a lie. Neither true nor false!

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn’t shave themselves is shaved by the barber.

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn’t shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
```

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program.

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")?

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
 if Halts(P,P): while(true): pass
 else:
  return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops!

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
 if Halts(P,P): while(true): pass
 else:
  return
```

...Text of Halt...

Halt Progam $\Longrightarrow$ Turing Program. ($P \Longrightarrow Q$)

Turing("Turing")? Neither halts nor loops! $\Longrightarrow$ No Turing program.

No Turing Program $\Longrightarrow$

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$ No halt program.

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$ No halt program. ($\neg Q \implies \neg P$)

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$ No halt program. ($\neg Q \implies \neg P$)

Program is text, so we can pass it to itself,

Back to technical..

This statement is a lie. Neither true nor false!

Every person who doesn't shave themselves is shaved by the barber.

Who shaves the barber?

```
def Turing(P):
  if Halts(P,P): while(true): pass
  else:
    return
```

...Text of Halt...

Halt Progam $\implies$ Turing Program. ($P \implies Q$)

Turing("Turing")? Neither halts nor loops! $\implies$ No Turing program.

No Turing Program $\implies$ No halt program. ($\neg Q \implies \neg P$)

Program is text, so we can pass it to itself,
or refer to self.

Summary: decidability.

Computer Programs are an interesting thing.

Summary: decidability.

Computer Programs are an interesting thing.
Like Math.

Summary: decidability.

Computer Programs are an interesting thing.
- Like Math.
- Formal Systems.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely “understand” computer programs.

Summary: decidability.

Computer Programs are an interesting thing.
  Like Math.
  Formal Systems.

Computer Programs cannot completely "understand" computer programs.

Computation is a lens for other action in the world.

Probability

What's to come?

Probability

What's to come? Probability.

Probability

What's to come? Probability.

A bag contains:

Probability

What's to come? Probability.

A bag contains:

Probability

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Probability

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue.

Probability

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total.

Probability

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

Probability

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

For now:

Probability

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

For now: Counting!

Probability

What's to come? Probability.

A bag contains:

What is the chance that a ball taken from the bag is blue?

Count blue. Count total. Divide.

For now: Counting!

Later: Probability.