Today

Review for Final..

Today

Review for Final..

Rao's Cheat Sheet.

Today

Review for Final..

Rao's Cheat Sheet.

A million slides.

Notes: Logic/Proofs.

Statement?

Notes: Logic/Proofs.

Statement? “3 = 4+1”?

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3?

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No
Logic.

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
Direct.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
  Direct. Square of even number is even.
  Contrapositive.

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
  Direct. Square of even number is even.
  Contrapositive. Square root of even number is even.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
  Direct. Square of even number is even.
  Contrapositive. Square root of even number is even.
  Contradiciton.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
Direct. Square of even number is even.
Contrapositive. Square root of even number is even.
Contradiciton. $\sqrt{2}$ is irrational.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
 Direct. Square of even number is even.
 Contrapositive. Square root of even number is even.
 Contradiciton. $\sqrt{2}$ is irrational.
Induction.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
 Direct. Square of even number is even.
 Contrapositive. Square root of even number is even.
 Contradiciton. $\sqrt{2}$ is irrational.
Induction.
 Statement: $\forall n, P(n)$.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No

Logic. $P \implies Q \equiv Q \implies P$ False.

Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.

Proofs.

Direct. Square of even number is even.

Contrapositive. Square root of even number is even.

Contradiciton. $\sqrt{2}$ is irrational.

Induction.

Statement: $\forall n, P(n)$. Base: $P(0)$.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
Direct. Square of even number is even.
Contrapositive. Square root of even number is even.
Contradiciton. $\sqrt{2}$ is irrational.
Induction.
Statement: $\forall n, P(n)$. Base: $P(0)$. Step: $P(k) \implies P(k+1)$.
Simple: $\sum_i i = n(n+1)/2$.

Notes: Logic/Proofs.

Statement? "3 = 4+1"? Yes. 3? No
Logic. $P \implies Q \equiv Q \implies P$ False.
Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.
Proofs.
Direct. Square of even number is even.
Contrapositive. Square root of even number is even.
Contradiciton. $\sqrt{2}$ is irrational.
Induction.
Statement: $\forall n, P(n)$. Base: $P(0)$. Step: $P(k) \implies P(k+1)$.
Simple: $\sum_i i = n(n+1)/2$. Strengthen: See midterm 1.
Strong induction: primes have a factorization.

Notes: Logic/Proofs.

Statement? “3 = 4+1”? Yes. 3? No

Logic. $P \implies Q \equiv Q \implies P$ False.

Quantifiers. $\neg \forall x, Q(x) \equiv \exists x, \neg Q(x)$.

Proofs.

Direct. Square of even number is even.

Contrapositive. Square root of even number is even.

Contradiciton. $\sqrt{2}$ is irrational.

Induction.

Statement: $\forall n, P(n)$. Base: $P(0)$. Step: $P(k) \implies P(k+1)$.

Simple: $\sum_i i = n(n+1)/2$. Strengthen: See midterm 1.

Strong induction: primes have a factorization.

Stable Marriage

Stable Marriage:

Stable Marriage

Stable Marriage:
Improvement Lemma. Optimality/Pessimality.

Stable Marriage

Stable Marriage:

Improvement Lemma. Optimality/Pessimality.

An an instance, with a stable instance where man 1 and woman 1 are optimal.

Stable Marriage

Stable Marriage:

Improvement Lemma. Optimality/Pessimality.

An an instance, with a stable instance where man 1 and woman 1 are optimal.

There is only one stable marriage!

Stable Marriage

Stable Marriage:
Improvement Lemma. Optimality/Pessimality.
An an instance, with a stable instance where man 1 and woman 1 are optimal.
There is only one stable marriage! False.

Graphs

Graphs:

Graphs

Graphs:
$\sum_v d(v) =$

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian:

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Coloring: Degree d graph can be colored with $d+1$ colors.

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Coloring: Degree d graph can be colored with $d+1$ colors.

Algorithm:

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Coloring: Degree d graph can be colored with $d+1$ colors.

Algorithm:

Remove vertex.

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Coloring: Degree d graph can be colored with $d+1$ colors.

Algorithm:

Remove vertex. Color remaining.

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Coloring: Degree d graph can be colored with $d+1$ colors.

Algorithm:

Remove vertex. Color remaining. Add vertex.

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Coloring: Degree d graph can be colored with $d+1$ colors.

Algorithm:

Remove vertex. Color remaining. Add vertex. Available color!

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v - 1$, $f = 1$.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v - 1$, $f = 1$.
Step: $v + f = e + 2$.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v - 1$, $f = 1$.
Step: $v + f = e + 2$. Add edge, adds face.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v+f = e+2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v+f = e+2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.

$$2e \geq 3f \implies v + 2e/3 \geq e+2 \implies e \leq 3v-6.$$

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v+f = e+2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.
$2e \geq 3f \implies v + 2e/3 \geq e+2 \implies e \leq 3v-6$.
6-color theorem.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v - 1$, $f = 1$.
Step: $v + f = e + 2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.
$2e \geq 3f \implies v + 2e/3 \geq e + 2 \implies e \leq 3v - 6$.
6-color theorem. 5-color is a recoloring argument.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v+f = e+2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.
$2e \geq 3f \implies v+2e/3 \geq e+2 \implies e \leq 3v-6$.
6-color theorem. 5-color is a recoloring argument.
Graphs:

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Coloring: Degree d graph can be colored with $d+1$ colors.

Algorithm:

Remove vertex. Color remaining. Add vertex. Available color!

Planar Graph: Euler Formula?

Proof: Base. Tree. $e = v - 1$, $f = 1$.

Step: $v + f = e + 2$. Add edge, adds face.

Max Degree: remove faces from equation using face-edge incidences.

$2e \geq 3f \implies v + 2e/3 \geq e + 2 \implies e \leq 3v - 6$.

6-color theorem. 5-color is a recoloring argument.

Graphs:

Complete: K_n. How many edges?

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v+f = e+2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.

$$2e \geq 3f \implies v + 2e/3 \geq e+2 \implies e \leq 3v-6.$$

6-color theorem. 5-color is a recoloring argument.
Graphs:
Complete: K_n. How many edges? $\binom{n}{2}$.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v - 1$, $f = 1$.
Step: $v + f = e + 2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.

$2e \geq 3f \implies v + 2e/3 \geq e + 2 \implies e \leq 3v - 6$.

6-color theorem. 5-color is a recoloring argument.
Graphs:
Complete: K_n. How many edges? $\binom{n}{2}$.
Tree: How many edges?

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v + f = e + 2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.
$2e \geq 3f \implies v + 2e/3 \geq e + 2 \implies e \leq 3v - 6$.
6-color theorem. 5-color is a recoloring argument.
Graphs:
Complete: K_n. How many edges? $\binom{n}{2}$.
Tree: How many edges? $n-1$.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v - 1$, $f = 1$.
Step: $v + f = e + 2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.

$$2e \geq 3f \implies v + 2e/3 \geq e + 2 \implies e \leq 3v - 6.$$

6-color theorem. 5-color is a recoloring argument.
Graphs:
Complete: K_n. How many edges? $\binom{n}{2}$.
Tree: How many edges? $n - 1$. No cycles.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v - 1$, $f = 1$.
Step: $v + f = e + 2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.
$2e \geq 3f \implies v + 2e/3 \geq e + 2 \implies e \leq 3v - 6$.
6-color theorem. 5-color is a recoloring argument.
Graphs:
Complete: K_n. How many edges? $\binom{n}{2}$.
Tree: How many edges? $n-1$. No cycles.
Hypercube: d-dimensional.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v+f = e+2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.
$2e \geq 3f \implies v+2e/3 \geq e+2 \implies e \leq 3v-6$.
6-color theorem. 5-color is a recoloring argument.
Graphs:
Complete: K_n. How many edges? $\binom{n}{2}$.
Tree: How many edges? $n-1$. No cycles.
Hypercube: d-dimensional. Degree?

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v + f = e + 2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.
$2e \geq 3f \implies v + 2e/3 \geq e + 2 \implies e \leq 3v - 6$.
6-color theorem. 5-color is a recoloring argument.
Graphs:
Complete: K_n. How many edges? $\binom{n}{2}$.
Tree: How many edges? $n-1$. No cycles.
Hypercube: d-dimensional. Degree? d.

Graphs

Graphs:

$\sum_v d(v) = 2|E|$.

Eulerian: All degrees even.

Coloring: Degree d graph can be colored with $d+1$ colors.

Algorithm:

Remove vertex. Color remaining. Add vertex. Available color!

Planar Graph: Euler Formula?

Proof: Base. Tree. $e = v-1$, $f = 1$.

Step: $v+f = e+2$. Add edge, adds face.

Max Degree: remove faces from equation using face-edge incidences.

$2e \geq 3f \implies v + 2e/3 \geq e+2 \implies e \leq 3v-6$.

6-color theorem. 5-color is a recoloring argument.

Graphs:

Complete: K_n. How many edges? $\binom{n}{2}$.

Tree: How many edges? $n-1$. No cycles.

Hypercube: d-dimensional. Degree? d. Edges: $d2^d/2$.

Graphs

Graphs:
$\sum_v d(v) = 2|E|$.
Eulerian: All degrees even.
Coloring: Degree d graph can be colored with $d+1$ colors.
Algorithm:
Remove vertex. Color remaining. Add vertex. Available color!
Planar Graph: Euler Formula?
Proof: Base. Tree. $e = v-1$, $f = 1$.
Step: $v+f = e+2$. Add edge, adds face.
Max Degree: remove faces from equation using face-edge incidences.
$2e \geq 3f \implies v + 2e/3 \geq e+2 \implies e \leq 3v-6$.
6-color theorem. 5-color is a recoloring argument.
Graphs:
Complete: K_n. How many edges? $\binom{n}{2}$.
Tree: How many edges? $n-1$. No cycles.
Hypercube: d-dimensional. Degree? d. Edges: $d2^d/2$.

Notes: Modular Arithmetic.

Euclid:

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x, y-x)$

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended:

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses!

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$. $x^{-1} \mod m$?

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x, y-x) = gcd(x, y-kx)$
Extended: $ax + by = gcd(x,y)$.
Start with $(1)x + (0)y = x$ and $(0)x + 1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax + bm = 1$. $x^{-1} \mod m$?
Fermats: $a^{p-1} = 1 \mod p$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x, y-x) = gcd(x, y-kx)$
Extended: $ax + by = gcd(x,y)$.
Start with $(1)x + (0)y = x$ and $(0)x + 1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax + bm = 1$. $x^{-1} \mod m$?
Fermats: $a^{p-1} = 1 \mod p$.
Proof: Multiplying by a is bijection on $\{1, \ldots p\}$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$. $x^{-1} \mod m$?
Fermats: $a^{p-1} = 1 \mod p$.
Proof: Multiplying by a is bijection on $\{1,\ldots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \mod (p-1)(q-1)$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x, y-x) = gcd(x, y-kx)$
Extended: $ax + by = gcd(x,y)$.
Start with $(1)x + (0)y = x$ and $(0)x + 1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax + bm = 1$. $x^{-1} \mod m$?
Fermats: $a^{p-1} = 1 \mod p$.
Proof: Multiplying by a is bijection on $\{1, \ldots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \mod (p-1)(q-1)$.
Works because: $a^{(p-1)(q-1)} = 1 \pmod 1$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$. $x^{-1} \mod m$?
Fermats: $a^{p-1} = 1 \mod p$.
Proof: Multiplying by a is bijection on $\{1,\ldots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \mod (p-1)(q-1)$.
Works because: $a^{(p-1)(q-1)} = 1 \pmod 1$. ....

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax + by = gcd(x,y)$.
Start with $(1)x + (0)y = x$ and $(0)x + 1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax + bm = 1$. $x^{-1} \pmod m$?
Fermats: $a^{p-1} = 1 \pmod p$.
Proof: Multiplying by a is bijection on $\{1,\ldots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \pmod{(p-1)(q-1)}$.
Works because: $a^{(p-1)(q-1)} = 1 \pmod 1$. .... Public Key Encryption/Signature Scheme.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$. $x^{-1} \pmod m$?
Fermats: $a^{p-1} = 1 \pmod p$.
Proof: Multiplying by a is bijection on $\{1,\ldots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \pmod{(p-1)(q-1)}$.
Works because: $a^{(p-1)(q-1)} = 1 \pmod 1$. .... Public Key Encryption/Signature Scheme.
Encrypt:

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$. $x^{-1} \pmod m$?
Fermats: $a^{p-1} = 1 \pmod p$.
Proof: Multiplying by a is bijection on $\{1,\ldots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \pmod{(p-1)(q-1)}$.
Works because: $a^{(p-1)(q-1)} = 1 \pmod 1$. .... Public Key Encryption/Signature Scheme.
Encrypt: $x^e \pmod N$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$. $x^{-1} \pmod m$?
Fermats: $a^{p-1} = 1 \pmod p$.
Proof: Multiplying by a is bijection on $\{1,\dots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \pmod{(p-1)(q-1)}$.
Works because: $a^{(p-1)(q-1)} = 1 \pmod 1$. .... Public Key Encryption/Signature Scheme.
Encrypt: $x^e \pmod N$. Sign:

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$. $x^{-1} \mod m$?
Fermats: $a^{p-1} = 1 \mod p$.
Proof: Multiplying by a is bijection on $\{1,\dots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \mod (p-1)(q-1)$.
Works because: $a^{(p-1)(q-1)} = 1 \pmod 1$. .... Public Key Encryption/Signature Scheme.
Encrypt: $x^e \mod N$. Sign: $x^d \mod N$.

Notes: Modular Arithmetic.

Euclid: $gcd(x,y) = gcd(x,y-x) = gcd(x,y-kx)$
Extended: $ax+by = gcd(x,y)$.
Start with $(1)x+(0)y = x$ and $(0)x+1y = y$.
Can reduce right hand side. By factor of two in two steps.
Multiplicative inverses! $ax+bm = 1$. $x^{-1} \mod m$?
Fermats: $a^{p-1} = 1 \mod p$.
Proof: Multiplying by a is bijection on $\{1,\ldots p\}$.
RSA: $(N = pq, e)$ where $e = d^{-1} \mod (p-1)(q-1)$.
Works because: $a^{(p-1)(q-1)} = 1 \pmod 1$. .... Public Key Encryption/Signature Scheme.
Encrypt: $x^e \mod N$. Sign: $x^d \mod N$.
Avoid Attack: add randomness to x.

Notes: Modular + Polynomials

Polynomials:

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.

Prop 1: $\leq d$ roots.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.
Prop 1: $\leq d$ roots. Factoring.
Prop 2: $d + 1$ points gives unique polynomial.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.

Prop 1: $\leq d$ roots. Factoring.

Prop 2: $d+1$ points gives unique polynomial.

Lagrange: 1 at a point, 0 elsewhere.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.
 Prop 1: $\leq d$ roots. Factoring.
 Prop 2: $d+1$ points gives unique polynomial.
  Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.
  Equations: $d+1$ unknowns, $d+1$ equations.
   Modulo prime: inverses gives hope.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.

Prop 1: $\leq d$ roots. Factoring.

Prop 2: $d+1$ points gives unique polynomial.

Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.

Equations: $d+1$ unknowns, $d+1$ equations.

Modulo prime: inverses gives hope.

Linearly independent from uniqueness.

Applications.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.

Prop 1: $\leq d$ roots. Factoring.

Prop 2: $d+1$ points gives unique polynomial.

Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.

Equations: $d+1$ unknowns, $d+1$ equations.

Modulo prime: inverses gives hope.

Linearly independent from uniqueness.

Applications.

Secret Sharing: Property 2.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.
Prop 1: $\leq d$ roots. Factoring.
Prop 2: $d+1$ points gives unique polynomial.
Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.
Equations: $d+1$ unknowns, $d+1$ equations.
Modulo prime: inverses gives hope.
Linearly independent from uniqueness.
Applications.
Secret Sharing: Property 2. Large prime for secrecy.
Erasure Coding: Property 2.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.
- Prop 1: $\leq d$ roots. Factoring.
- Prop 2: $d+1$ points gives unique polynomial.
  - Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.
  - Equations: $d+1$ unknowns, $d+1$ equations.
    - Modulo prime: inverses gives hope.
    - Linearly independent from uniqueness.
- Applications.
  - Secret Sharing: Property 2. Large prime for secrecy.
  - Erasure Coding: Property 2. Smaller prime for efficiency.
  - Error Correction: Property 2.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.
  Prop 1: $\leq d$ roots. Factoring.
  Prop 2: $d+1$ points gives unique polynomial.
    Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.
    Equations: $d+1$ unknowns, $d+1$ equations.
      Modulo prime: inverses gives hope.
      Linearly independent from uniqueness.
  Applications.
    Secret Sharing: Property 2. Large prime for secrecy.
    Erasure Coding: Property 2. Smaller prime for efficiency.
    Error Correction: Property 2.
      Argument that $n+2k$ is enough with k errros.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.

Prop 1: $\leq d$ roots. Factoring.

Prop 2: $d+1$ points gives unique polynomial.

Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.

Equations: $d+1$ unknowns, $d+1$ equations.

Modulo prime: inverses gives hope.

Linearly independent from uniqueness.

Applications.

Secret Sharing: Property 2. Large prime for secrecy.

Erasure Coding: Property 2. Smaller prime for efficiency.

Error Correction: Property 2.

Argument that $n+2k$ is enough with k errros.

Unique degree $n-1$ polynomial that fits at least $n+k$ points.

Why?

Welsh-Berlekamp:

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.
- Prop 1: $\leq d$ roots. Factoring.
- Prop 2: $d+1$ points gives unique polynomial.
  - Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.
  - Equations: $d+1$ unknowns, $d+1$ equations.
    - Modulo prime: inverses gives hope.
    - Linearly independent from uniqueness.
- Applications.
  - Secret Sharing: Property 2. Large prime for secrecy.
  - Erasure Coding: Property 2. Smaller prime for efficiency.
  - Error Correction: Property 2.
    - Argument that $n+2k$ is enough with k errros.
      - Unique degree $n-1$ polynomial that fits at least $n+k$ points.
      - Why?
    - Welsh-Berlekamp:
      - Linear System from $Q(x) = P(x)E(x)$ with error poly, $E(x)$.

Notes: Modular + Polynomials

Polynomials: $a_d x^d + \cdots a + 0 \mod p$.
- Prop 1: $\leq d$ roots. Factoring.
- Prop 2: $d+1$ points gives unique polynomial.
  - Lagrange: 1 at a point, 0 elsewhere. Degree d polynomial suffices.
  - Equations: $d+1$ unknowns, $d+1$ equations.
    - Modulo prime: inverses gives hope.
    - Linearly independent from uniqueness.
- Applications.
  - Secret Sharing: Property 2. Large prime for secrecy.
  - Erasure Coding: Property 2. Smaller prime for efficiency.
  - Error Correction: Property 2.
    - Argument that $n+2k$ is enough with k errros.
      - Unique degree $n-1$ polynomial that fits at least $n+k$ points.
      - Why?
    - Welsh-Berlekamp:
      - Linear System from $Q(x) = P(x)E(x)$ with error poly, $E(x)$.
      - Divide $Q(x)$ by $E(x)$ to get $P(x)$.

Notes: Countability/Computability

Countability/Computability.

Notes: Countability/Computability

Countability/Computability.
Countable: bijection with natural numbers or a listing.

Notes: Countability/Computability

Countability/Computability.

Countable: bijection with natural numbers or a listing.

Countable infinities: pairs of countable sets, rationals...

Notes: Countability/Computability

Countability/Computability.

Countable: bijection with natural numbers or a listing.

Countable infinities: pairs of countable sets, rationals...

all forms of pairs: interleave elements of uncountable sets.

Notes: Countability/Computability

Countability/Computability.
Countable: bijection with natural numbers or a listing.
Countable infinities: pairs of countable sets, rationals...
all forms of pairs: interleave elements of uncountable sets.
Uncountable infinities: real numbers, power set of integers.

Notes: Countability/Computability

Countability/Computability.
Countable: bijection with natural numbers or a listing.
Countable infinities: pairs of countable sets, rationals...
all forms of pairs: interleave elements of uncountable sets.
Uncountable infinities: real numbers, power set of integers.
Diagonalization:

Notes: Countability/Computability

Countability/Computability.
Countable: bijection with natural numbers or a listing.
Countable infinities: pairs of countable sets, rationals...
all forms of pairs: interleave elements of uncountable sets.
Uncountable infinities: real numbers, power set of integers.
Diagonalization: Assume list, construct element not on list.

Notes: Countability/Computability

Countability/Computability.

- Countable: bijection with natural numbers or a listing.
  - Countable infinities: pairs of countable sets, rationals...
    - all forms of pairs: interleave elements of uncountable sets.
  - Uncountable infinities: real numbers, power set of integers.
    - Diagonalization: Assume list, construct element not on list.
- Uncomputability.

Notes: Countability/Computability

Countability/Computability.

- Countable: bijection with natural numbers or a listing.
  - Countable infinities: pairs of countable sets, rationals...
    - all forms of pairs: interleave elements of uncountable sets.
  - Uncountable infinities: real numbers, power set of integers.
    - Diagonalization: Assume list, construct element not on list.
- Uncomputability.
  - Halt: With halt can construct diagonalizer Turing.

Notes: Countability/Computability

Countability/Computability.

- Countable: bijection with natural numbers or a listing.
  - Countable infinities: pairs of countable sets, rationals...
    - all forms of pairs: interleave elements of uncountable sets.
  - Uncountable infinities: real numbers, power set of integers.
    - Diagonalization: Assume list, construct element not on list.
- Uncomputability.
  - Halt: With halt can construct diagonalizer Turing.
    - and no Turing $\implies$ no halt.

Notes: Countability/Computability

Countability/Computability.
Countable: bijection with natural numbers or a listing.
Countable infinities: pairs of countable sets, rationals...
all forms of pairs: interleave elements of uncountable sets.
Uncountable infinities: real numbers, power set of integers.
Diagonalization: Assume list, construct element not on list.
Uncomputability.
Halt: With halt can construct diagonalizer Turing.
and no Turing $\implies$ no halt.
Concepts: Program can call subroutine!

Notes: Countability/Computability

Countability/Computability.
- Countable: bijection with natural numbers or a listing.
  - Countable infinities: pairs of countable sets, rationals...
    - all forms of pairs: interleave elements of uncountable sets.
  - Uncountable infinities: real numbers, power set of integers.
    - Diagonalization: Assume list, construct element not on list.
- Uncomputability.
  - Halt: With halt can construct diagonalizer Turing.
    - and no Turing $\implies$ no halt.
    - Concepts: Program can call subroutine!
      - With subroutine can write program.

Notes: Countability/Computability

Countability/Computability.
- Countable: bijection with natural numbers or a listing.
  - Countable infinities: pairs of countable sets, rationals...
    - all forms of pairs: interleave elements of uncountable sets.
  - Uncountable infinities: real numbers, power set of integers.
    - Diagonalization: Assume list, construct element not on list.
- Uncomputability.
  - Halt: With halt can construct diagonalizer Turing.
    - and no Turing $\implies$ no halt.
    - Concepts: Program can call subroutine!
      - With subroutine can write program.
  - Reduce from Halt:

Notes: Countability/Computability

Countability/Computability.

Countable: bijection with natural numbers or a listing.

Countable infinities: pairs of countable sets, rationals...

all forms of pairs: interleave elements of uncountable sets.

Uncountable infinities: real numbers, power set of integers.

Diagonalization: Assume list, construct element not on list.

Uncomputability.

Halt: With halt can construct diagonalizer Turing.

and no Turing $\implies$ no halt.

Concepts: Program can call subroutine!

With subroutine can write program.

Reduce from Halt:

Transform instance of halt to instance of problem X.

Notes: Countability/Computability

Countability/Computability.
- Countable: bijection with natural numbers or a listing.
  - Countable infinities: pairs of countable sets, rationals...
    - all forms of pairs: interleave elements of uncountable sets.
  - Uncountable infinities: real numbers, power set of integers.
    - Diagonalization: Assume list, construct element not on list.
- Uncomputability.
  - Halt: With halt can construct diagonalizer Turing.
    - and no Turing $\implies$ no halt.
    - Concepts: Program can call subroutine!
      - With subroutine can write program.
  - Reduce from Halt:
    - Transform instance of halt to instance of problem X.
    - Concept: Programs are text. Can change text.

Notes: Countability/Computability

Countability/Computability.
- Countable: bijection with natural numbers or a listing.
  - Countable infinities: pairs of countable sets, rationals...
    - all forms of pairs: interleave elements of uncountable sets.
  - Uncountable infinities: real numbers, power set of integers.
    - Diagonalization: Assume list, construct element not on list.
- Uncomputability.
  - Halt: With halt can construct diagonalizer Turing.
    - and no Turing $\implies$ no halt.
    - Concepts: Program can call subroutine!
      - With subroutine can write program.
  - Reduce from Halt:
    - Transform instance of halt to instance of problem X.
    - Concept: Programs are text. Can change text.
- Computability/Enumerability.

Notes: Countability/Computability

Countability/Computability.
- Countable: bijection with natural numbers or a listing.
  - Countable infinities: pairs of countable sets, rationals...
    - all forms of pairs: interleave elements of uncountable sets.
  - Uncountable infinities: real numbers, power set of integers.
    - Diagonalization: Assume list, construct element not on list.
- Uncomputability.
  - Halt: With halt can construct diagonalizer Turing.
    - and no Turing $\implies$ no halt.
    - Concepts: Program can call subroutine!
      - With subroutine can write program.
  - Reduce from Halt:
    - Transform instance of halt to instance of problem X.
    - Concept: Programs are text. Can change text.
- Computability/Enumerability.
  - Can run programs and see!

Notes: Countability/Computability

Countability/Computability.
Countable: bijection with natural numbers or a listing.
Countable infinities: pairs of countable sets, rationals...
all forms of pairs: interleave elements of uncountable sets.
Uncountable infinities: real numbers, power set of integers.
Diagonalization: Assume list, construct element not on list.
Uncomputability.
Halt: With halt can construct diagonalizer Turing.
and no Turing $\implies$ no halt.
Concepts: Program can call subroutine!
With subroutine can write program.
Reduce from Halt:
Transform instance of halt to instance of problem X.
Concept: Programs are text. Can change text.
Computability/Enumerability.
Can run programs and see!
Can enumerate halting programs.

Notes: Counting

Counting.

Notes: Counting

Counting.
First rule of counting.

Notes: Counting

Counting.

First rule of counting.

Make elt of set with sequence of choices.

Notes: Counting

Counting.

First rule of counting.

Make elt of set with sequence of choices. Multiply.

Notes: Counting

Counting.
First rule of counting.
Make elt of set with sequence of choices. Multiply.
Second rule of counting.

Notes: Counting

Counting.

- First rule of counting.
  - Make elt of set with sequence of choices. Multiply.
- Second rule of counting.
  - Divide with order to get number of unordered.

Notes: Counting

Counting.

First rule of counting.

Make elt of set with sequence of choices. Multiply.

Second rule of counting.

Divide with order to get number of unordered. Sometimes.

Notes: Counting

Counting.
- First rule of counting.
  - Make elt of set with sequence of choices. Multiply.
- Second rule of counting.
  - Divide with order to get number of unordered. Sometimes.

Stars and Bars.

Notes: Counting

Counting.

First rule of counting.

Make elt of set with sequence of choices. Multiply.

Second rule of counting.

Divide with order to get number of unordered. Sometimes.

Stars and Bars. Use bars to group stars into different groups.

Notes: Counting

Counting.

- First rule of counting.
  - Make elt of set with sequence of choices. Multiply.
- Second rule of counting.
  - Divide with order to get number of unordered. Sometimes.

Stars and Bars. Use bars to group stars into different groups.

Inclusion/Exclusion.

Notes: Counting

Counting.
- First rule of counting.
  - Make elt of set with sequence of choices. Multiply.
- Second rule of counting.
  - Divide with order to get number of unordered. Sometimes.

Stars and Bars. Use bars to group stars into different groups.

Inclusion/Exclusion.
- Number in union is sum minus the intersection.

Notes: Counting

Counting.
- First rule of counting.
  - Make elt of set with sequence of choices. Multiply.
- Second rule of counting.
  - Divide with order to get number of unordered. Sometimes.

Stars and Bars. Use bars to group stars into different groups.

Inclusion/Exclusion.
- Number in union is sum minus the intersection.

Combinatorial Arguments:

Notes: Counting

Counting.

First rule of counting.

Make elt of set with sequence of choices. Multiply.

Second rule of counting.

Divide with order to get number of unordered. Sometimes.

Stars and Bars. Use bars to group stars into different groups.

Inclusion/Exclusion.

Number in union is sum minus the intersection.

Combinatorial Arguments: Bijection means same number.

Notes: Counting

Counting.
First rule of counting.
Make elt of set with sequence of choices. Multiply.
Second rule of counting.
Divide with order to get number of unordered. Sometimes.
Stars and Bars. Use bars to group stars into different groups.
Inclusion/Exclusion.
Number in union is sum minus the intersection.
Combinatorial Arguments: Bijection means same number.
$2^n = \sum_i \binom{n}{i}$.

Notes: Counting

Counting.
First rule of counting.
Make elt of set with sequence of choices. Multiply.
Second rule of counting.
Divide with order to get number of unordered. Sometimes.
Stars and Bars. Use bars to group stars into different groups.
Inclusion/Exclusion.
Number in union is sum minus the intersection.
Combinatorial Arguments: Bijection means same number.
$2^n = \sum_i \binom{n}{i}$.

Note: for sample spaces, usually first rule of counting is easier.

Notes: Counting

Counting.
First rule of counting.
Make elt of set with sequence of choices. Multiply.
Second rule of counting.
Divide with order to get number of unordered. Sometimes.
Stars and Bars. Use bars to group stars into different groups.
Inclusion/Exclusion.
Number in union is sum minus the intersection.
Combinatorial Arguments: Bijection means same number.
$2^n = \sum_i \binom{n}{i}$.

Note: for sample spaces, usually first rule of counting is easier.
for events, may need second or others.

First there was logic...

A statement is a true or false.

First there was logic...

A statement is a true or false.

Statements?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a statement.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a statement.

Quantifiers:

First there was logic...

A statement is a true or false.
Statements?
$3 = 4 - 1$? Statement!
$3 = 5$? Statement!
3 ? Not a statement!
$n = 3$? Not a statement...but a predicate.
Predicate: Statement with free variable(s).
Example: $x = 3$ Given a value for x, becomes a statement.
Predicate?
$n > 3$? Predicate: $P(n)$!
$x = y$? Predicate: $P(x, y)$!
$x + y$? No. An expression, not a statement.
Quantifiers:
$(\forall x)\ P(x)$.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x, y)$!

$x + y$? No. An expression, not a statement.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x,y)$!

$x + y$? No. An expression, not a statement.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

$(\exists x)\ P(x)$.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x,y)$!

$x + y$? No. An expression, not a statement.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

$(\exists x)\ P(x)$. There exists an x, where $P(x)$ is true.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x,y)$!

$x + y$? No. An expression, not a statement.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

$(\exists x)\ P(x)$. There exists an x, where $P(x)$ is true.

$(\forall n \in N), n^2 \geq n$.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x,y)$!

$x + y$? No. An expression, not a statement.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

$(\exists x)\ P(x)$. There exists an x, where $P(x)$ is true.

$(\forall n \in N), n^2 \geq n$.

$(\forall x \in R)(\exists y \in R) y > x$.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

Example: $x = 3$ Given a value for x, becomes a statement.

Predicate?

$n > 3$? Predicate: $P(n)$!

$x = y$? Predicate: $P(x,y)$!

$x + y$? No. An expression, not a statement.

Quantifiers:

$(\forall x)\ P(x)$. For every x, $P(x)$ is true.

$(\exists x)\ P(x)$. There exists an x, where $P(x)$ is true.

$(\forall n \in N), n^2 \geq n$.

$(\forall x \in R)(\exists y \in R) y > x$.

Connecting Statements

$A \wedge B$, $A \vee B$, $\neg A$.

Connecting Statements

$A \wedge B$, $A \vee B$, $\neg A$.

You got this!

Connecting Statements

$A \wedge B$, $A \vee B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

Connecting Statements

$A \wedge B$, $A \vee B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \vee B)$

Connecting Statements

$A \wedge B$, $A \vee B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \vee B)$
$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$

Connecting Statements

$A \wedge B$, $A \vee B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

$(A \implies B) \equiv (\neg A \vee B)$
$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$

Connecting Statements

$A \wedge B$, $A \vee B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

$$(A \implies B) \equiv (\neg A \vee B)$$
$$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$$

Proofs: truth table or manipulation of known formulas.

Connecting Statements

$A \wedge B$, $A \vee B$, $\neg A$.

You got this!

Propositional Expressions and Logical Equivalence

$$(A \implies B) \equiv (\neg A \vee B)$$
$$\neg(A \vee B) \equiv (\neg A \wedge \neg B)$$

Proofs: truth table or manipulation of known formulas.

$$(\forall x)(P(x) \wedge Q(x)) \equiv (\forall x)P(x) \wedge (\forall x)Q(x)$$

..and then proofs...

Direct: $P \implies Q$

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even?

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies$ a^2 is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$a^2 = 2(2k^2)$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes does not exist.

..and then proofs...

Direct: $P \implies Q$

Example: a is even $\implies a^2$ is even.

Approach: What is even? $a = 2k$

$a^2 = 4k^2$.

What is even?

$$a^2 = 2(2k^2)$$

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$ or $\neg Q \implies \neg P$.

Example: a^2 is odd $\implies a$ is odd.

Contrapositive: a is even $\implies a^2$ is even.

Contradiction: P

$\neg P \implies$ **false**

$\neg P \implies R \wedge \neg R$

Useful for prove something does not exist:

Example: rational representation of $\sqrt{2}$ does not exist.

Example: finite set of primes does not exist.

Example: rogue couple does not exist.

...jumping forward..

Contradiction in induction:

...jumping forward..

Contradiction in induction:
    contradict place where induction step doesn't hold.

...jumping forward..

Contradiction in induction:
 contradict place where induction step doesn’t hold.

Well Ordering Principle.

...jumping forward..

Contradiction in induction:
  contradict place where induction step doesn't hold.

Well Ordering Principle.
 Stable Marriage:

...jumping forward..

Contradiction in induction:
contradict place where induction step doesn't hold.

Well Ordering Principle.
Stable Marriage:
first day where women does not improve.

...jumping forward..

Contradiction in induction:
  contradict place where induction step doesn't hold.

Well Ordering Principle.
  Stable Marriage:
    first day where women does not improve.
    first day where any man rejected by optimal women.

...jumping forward..

Contradiction in induction:
    contradict place where induction step doesn't hold.

Well Ordering Principle.
  Stable Marriage:
    first day where women does not improve.
    first day where any man rejected by optimal women.
  Do not exist.

...jumping forward..

Contradiction in induction:
    contradict place where induction step doesn't hold.

Well Ordering Principle.
  Stable Marriage:
    first day where women does not improve.
    first day where any man rejected by optimal women.
  Do not exist.

...jumping forward..

Contradiction in induction:
    contradict place where induction step doesn't hold.

Well Ordering Principle.
  Stable Marriage:
    first day where women does not improve.
    first day where any man rejected by optimal women.
  Do not exist.

...and then induction...

$$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$$

...and then induction...

$$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$$

Thm: For all $n \geq 1$, $8|3^{2n}-1$.

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n)$.

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n} - 1$.

Induction on n.

Base: $8|3^2 - 1$.

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: Assume $P(n)$: True for some n.

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n)$.

Thm: For all $n \geq 1$, $8|3^{2n}-1$.

Induction on n.

Base: $8|3^2-1$.

Induction Hypothesis: Assume $P(n)$: True for some n.

Induction Step: Prove $P(n+1)$

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n)$.

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: Assume $P(n)$: True for some n.

Induction Step: Prove $P(n+1)$

$3^{2n+2} - 1 =$

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n} - 1$.

Induction on n.

Base: $8|3^2 - 1$.

Induction Hypothesis: Assume $P(n)$: True for some n.

Induction Step: Prove $P(n+1)$

$$3^{2n+2} - 1 = 9(3^{2n}) - 1$$

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: Assume $P(n)$: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step: Prove $P(n+1)$

$3^{2n+2} - 1 = 9(3^{2n}) - 1$ (by induction hypothesis)

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n}-1$.

Induction on n.

Base: $8|3^2-1$.

Induction Hypothesis: Assume $P(n)$: True for some n.
$(3^{2n}-1 = 8d)$

Induction Step: Prove $P(n+1)$

$$\begin{aligned} 3^{2n+2}-1 &= 9(3^{2n})-1 \quad \text{(by induction hypothesis)} \\ &= 9(8d+1)-1 \end{aligned}$$

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n)$.

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: Assume $P(n)$: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step: Prove $P(n+1)$

$$\begin{aligned} 3^{2n+2} - 1 &= 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \\ &= 9(8d + 1) - 1 \\ &= 72d + 8 \end{aligned}$$

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: Assume $P(n)$: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step: Prove $P(n+1)$

$$
\begin{aligned}
3^{2n+2} - 1 &= 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \\
&= 9(8d+1) - 1 \\
&= 72d + 8 \\
&= 8(9d+1)
\end{aligned}
$$

...and then induction...

$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$

Thm: For all $n \geq 1$, $8|3^{2n} - 1$.

Induction on n.

Base: $8|3^2 - 1$.

Induction Hypothesis: Assume $P(n)$: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step: Prove $P(n+1)$

$$\begin{aligned} 3^{2n+2} - 1 &= 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \\ &= 9(8d+1) - 1 \\ &= 72d + 8 \\ &= 8(9d+1) \end{aligned}$$

Divisible by 8.

...and then induction...

$$P(0) \wedge ((\forall n)(P(n) \implies P(n+1) \equiv (\forall n \in N)\ P(n).$$

Thm: For all $n \geq 1$, $8 | 3^{2n} - 1$.

Induction on n.

Base: $8 | 3^2 - 1$.

Induction Hypothesis: Assume $P(n)$: True for some n.
$(3^{2n} - 1 = 8d)$

Induction Step: Prove $P(n+1)$

$$\begin{aligned} 3^{2n+2} - 1 &= 9(3^{2n}) - 1 \quad \text{(by induction hypothesis)} \\ &= 9(8d+1) - 1 \\ &= 72d + 8 \\ &= 8(9d+1) \end{aligned}$$

Divisible by 8.

□

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

Each person has completely ordered preference list

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their partners

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their partners

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their partners

Stable Pairing.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their partners

Stable Pairing.
Pairing with no rogue couples.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their partners

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their partners

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list
contains every person of opposite gender.

Pairing.
Set of pairs (m_i, w_j) containing all people *exactly* once.
How many pairs? n.
People in pair are **partners** in pairing.

Rogue Couple in a pairing.
A m_j and w_k who like each other more than their partners

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

No, for roommates problem.

TMA.

Traditional Marriage Algorithm:

TMA.

Traditional Marriage Algorithm:

Each Day:

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject."

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman,

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman,
$\implies$ any future man on string is better.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman,
$\Longrightarrow$ any future man on string is better.

Stability:

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman,
$\implies$ any future man on string is better.

Stability: No rogue couple.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman,
$\Longrightarrow$ any future man on string is better.

Stability: No rogue couple.
rogue couple (M,W)

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman,
$\implies$ any future man on string is better.

Stability: No rogue couple.
rogue couple (M,W)
$\implies$ M proposed to W

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman,
$\implies$ any future man on string is better.

Stability: No rogue couple.
rogue couple (M,W)
$\implies$ M proposed to W
$\implies$ W ended up with someone she liked better than M.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite woman who has not yet rejected him.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:
Man **crosses off** woman who rejected him.
Woman's current proposer is **"on string."**

"Propose and Reject." : Either men propose or women. But not both.
Traditional propose and reject where men propose.

Key Property: Improvement Lemma:
Every day, if man on string for woman,
$\implies$ any future man on string is better.

Stability: No rogue couple.
rogue couple (M,W)
$\implies$ M proposed to W
$\implies$ W ended up with someone she liked better than M.
Not rogue couple!

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.
  Possibly no stable pairing with that partner.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
- Not necessarily first in list.
- Possibly no stable pairing with that partner.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.
  Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.
  Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
  Not necessarily first in list.
  Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, *S*.
  First man *M* to lose optimal partner.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
 Not necessarily first in list.
 Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
 First man M to lose optimal partner.
 Better partner W for M.
 Different stable pairing T.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
 Not necessarily first in list.
 Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
 First man M to lose optimal partner.
 Better partner W for M.
 Different stable pairing T.
 TMA: M asked W first!

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.
Different stable pairing T.
TMA: M asked W first!
There is M' who bumps M in TMA.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
 Not necessarily first in list.
 Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
 First man M to lose optimal partner.
 Better partner W for M.
 Different stable pairing T.
 TMA: M asked W first!
 There is M' who bumps M in TMA.
 W prefers M'.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.
Different stable pairing T.
TMA: M asked W first!
There is M' who bumps M in TMA.
W prefers M'.
M' likes W at least as much as optimal partner.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.
Different stable pairing T.
TMA: M asked W first!
There is M' who bumps M in TMA.
W prefers M'.
M' likes W at least as much as optimal partner.
Not first bump.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.
Different stable pairing T.
TMA: M asked W first!
There is M' who bumps M in TMA.
W prefers M'.
M' likes W at least as much as optimal partner.
Not first bump.
M' and W is rogue couple in T.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.
Different stable pairing T.
TMA: M asked W first!
There is M' who bumps M in TMA.
W prefers M'.
M' likes W at least as much as optimal partner.
Not first bump.
M' and W is rogue couple in T.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.
Different stable pairing T.
TMA: M asked W first!
There is M' who bumps M in TMA.
W prefers M'.
M' likes W at least as much as optimal partner.
Not first bump.
M' and W is rogue couple in T.

Thm: woman pessimal.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.
Different stable pairing T.
TMA: M asked W first!
There is M' who bumps M in TMA.
W prefers M'.
M' likes W at least as much as optimal partner.
Not first bump.
M' and W is rogue couple in T.

Thm: woman pessimal.

Man optimal $\implies$ Woman pessimal.

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.
Not necessarily first in list.
Possibly no stable pairing with that partner.

Man-optimal pairing is pairing where every man gets optimal partner.

Thm: TMA produces male optimal pairing, S.
First man M to lose optimal partner.
Better partner W for M.
Different stable pairing T.
TMA: M asked W first!
There is M' who bumps M in TMA.
W prefers M'.
M' likes W at least as much as optimal partner.
Not first bump.
M' and W is rogue couple in T.

Thm: woman pessimal.

Man optimal $\implies$ Woman pessimal.
Woman optimal $\implies$ Man pessimal.

...Graphs...

$$G = (V, E)$$

...Graphs...

$G = (V, E)$
V - set of vertices.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.

Pair of Vertices are Connected:

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.

Pair of Vertices are Connected:
If there is a path between them.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.

Pair of Vertices are Connected:
If there is a path between them.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.

Pair of Vertices are Connected:
If there is a path between them.

Connected Component: maximal set of connected vertices.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.

Pair of Vertices are Connected:
If there is a path between them.

Connected Component: maximal set of connected vertices.

...Graphs...

$G = (V, E)$
V - set of vertices.
$E \subseteq V \times V$ - set of edges.

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.
In-degree, Out-degree.

Thm: Sum of degrees is $2|E|$.
Edge is incident to 2 vertices.
Degree of vertices is total incidences.

Pair of Vertices are Connected:
If there is a path between them.

Connected Component: maximal set of connected vertices.

Connected Graph: one connected component.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk using each edge at most once.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Proof Idea: Original graph connected.

Graph Algorithm: Eulerian Tour

Thm: Every connected graph where every vertex has even degree has an Eulerian Tour; a tour which visits every edge exactly once.

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

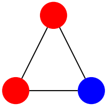
Proof Idea: Original graph connected.

Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.

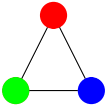
Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



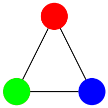
Graph Coloring.

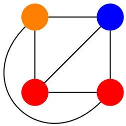
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



Graph Coloring.

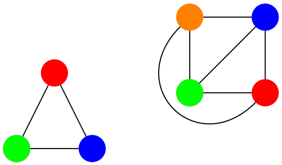
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.





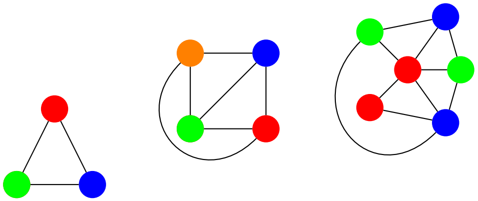
Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



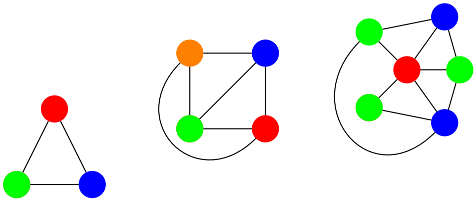
Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



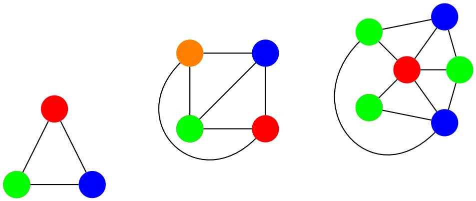
Graph Coloring.

Given $G = (V,E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



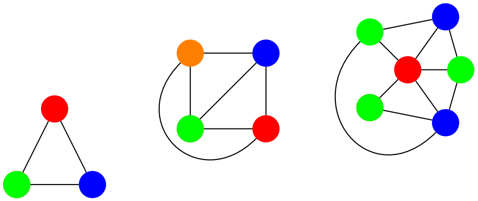
Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



Graph Coloring.

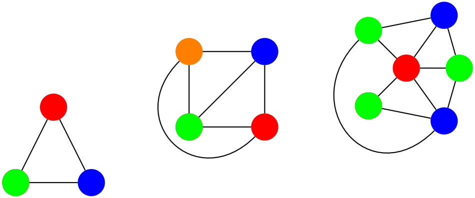
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



Notice that the last one, has one three colors.

Graph Coloring.

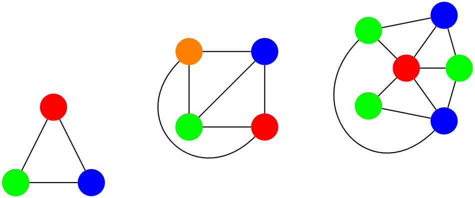
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



Notice that the last one, has one three colors.
  Fewer colors than number of vertices.

Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



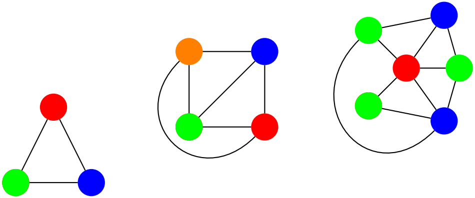
Notice that the last one, has one three colors.

Fewer colors than number of vertices.

Fewer colors than max degree node.

Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



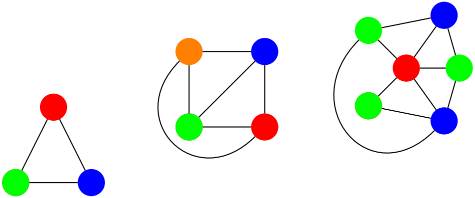
Notice that the last one, has one three colors.

Fewer colors than number of vertices.

Fewer colors than max degree node.

Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.

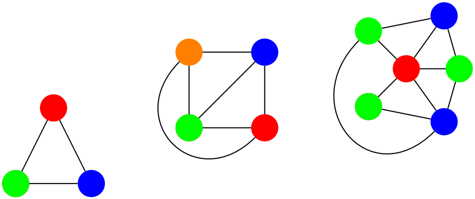


Notice that the last one, has one three colors.
  Fewer colors than number of vertices.
  Fewer colors than max degree node.

Interesting things to do.

Graph Coloring.

Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.

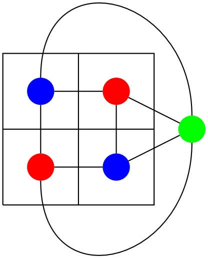


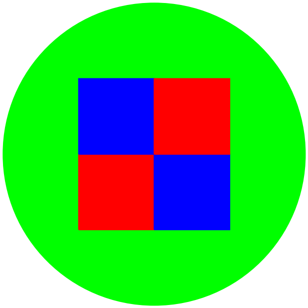
Notice that the last one, has one three colors.
  Fewer colors than number of vertices.
  Fewer colors than max degree node.

Interesting things to do. Algorithm!

Planar graphs and maps.

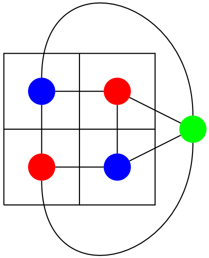
Planar graph coloring $\equiv$ map coloring.

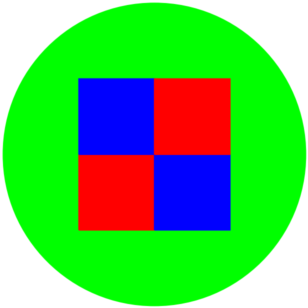




Planar graphs and maps.

Planar graph coloring $\equiv$ map coloring.





Four color theorem is about planar graphs!

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v}$

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v}$

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree < 6

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree < 6 or at most 5.

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree < 6 or at most 5.

Remove vertex v of degree at most 5.

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree < 6 or at most 5.

Remove vertex v of degree at most 5.
Inductively color remaining graph.

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree < 6 or at most 5.

Remove vertex v of degree at most 5.
Inductively color remaining graph.
Color is available for v since only five neighbors...

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree < 6 or at most 5.

Remove vertex v of degree at most 5.
Inductively color remaining graph.
Color is available for v since only five neighbors...
and only five colors are used.

Six color theorem.

Theorem: Every planar graph can be colored with six colors.

Proof:
Recall: $e \leq 3v - 6$ for any planar graph.
From Euler's Formula.

Total degree: $2e$
Average degree: $\leq \frac{2e}{v} \leq \frac{2(3v-6)}{v} \leq 6 - \frac{12}{v}$.

There exists a vertex with degree < 6 or at most 5.

Remove vertex v of degree at most 5.
Inductively color remaining graph.
Color is available for v since only five neighbors...
and only five colors are used. □

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.

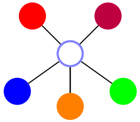
Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.

Assume neighbors are colored all differently.



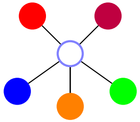
Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.

Assume neighbors are colored all differently.
Otherwise done.

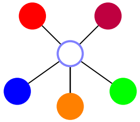


Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



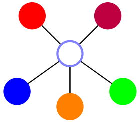
Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



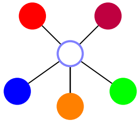
Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



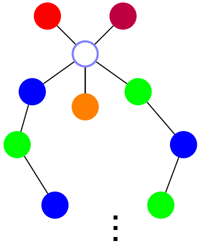
Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



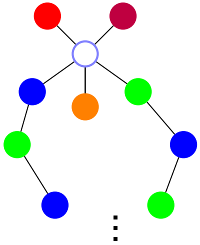
Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



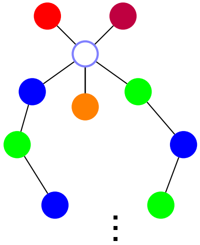
Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



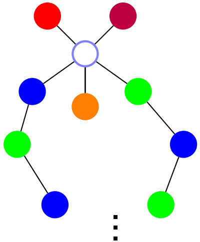
Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



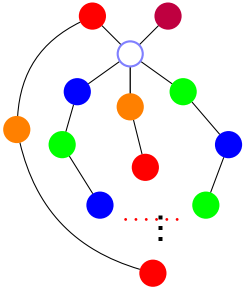
Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



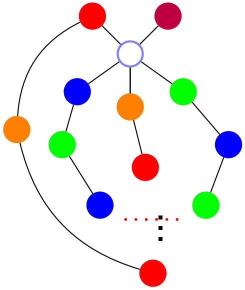
Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

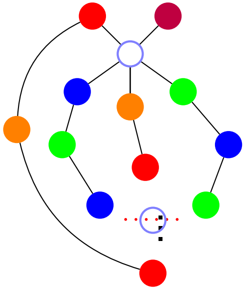
Planar.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

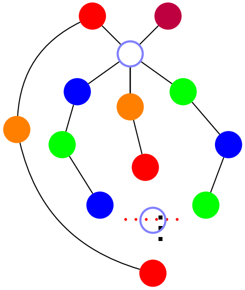
Planar. $\Longrightarrow$ paths intersect at a vertex!

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

Planar. $\Longrightarrow$ paths intersect at a vertex!

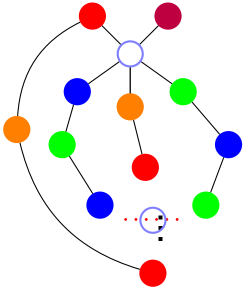
What color is it?

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

Planar. $\Longrightarrow$ paths intersect at a vertex!

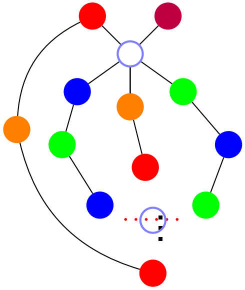
What color is it?

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

Planar. $\implies$ paths intersect at a vertex!

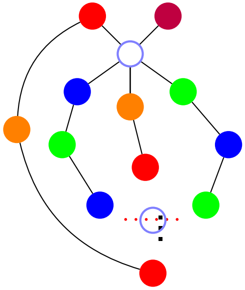
What color is it?
Must be blue or green to be on that path.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

Planar. $\implies$ paths intersect at a vertex!

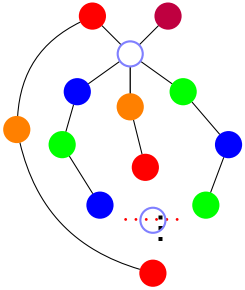
What color is it?
Must be blue or green to be on that path.
Must be red or orange to be on that path.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

Planar. $\Longrightarrow$ paths intersect at a vertex!

What color is it?
Must be blue or green to be on that path.
Must be red or orange to be on that path.

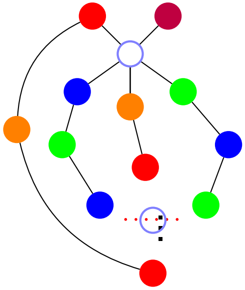
Contradiction.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
Otherwise done.
Switch green to blue in component.
Done. Unless blue-green path to blue.
Switch red to orange in its component.
Done. Unless red-orange path to red.

Planar. $\Longrightarrow$ paths intersect at a vertex!

What color is it?
Must be blue or green to be on that path.
Must be red or orange to be on that path.

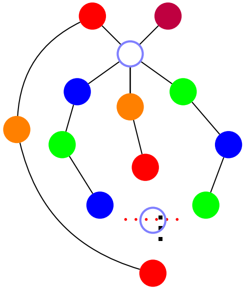
Contradiction. Can recolor one of the neighbors.
And recolor “center” vertex.

Five color theorem

Theorem: Every planar graph can be colored with five colors.

Preliminary Observation: Connected components of vertices with two colors in a legal coloring can switch colors.

Proof:
Again with the degree 5 vertex. Again recurse.



Assume neighbors are colored all differently.
  Otherwise done.
Switch green to blue in component.
  Done. Unless blue-green path to blue.
Switch red to orange in its component.
  Done. Unless red-orange path to red.

Planar. $\Longrightarrow$ paths intersect at a vertex!

What color is it?
  Must be blue or green to be on that path.
  Must be red or orange to be on that path.

Contradiction. Can recolor one of the neighbors.
And recolor "center" vertex. □

Four Color Theorem

Four Color Theorem

Theorem: Any planar graph can be colored with four colors.

Four Color Theorem

Theorem: Any planar graph can be colored with four colors.

Proof:

Four Color Theorem

Theorem: Any planar graph can be colored with four colors.

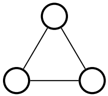
Proof: Not Today!

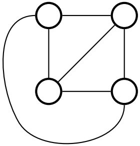
Four Color Theorem

Theorem: Any planar graph can be colored with four colors.

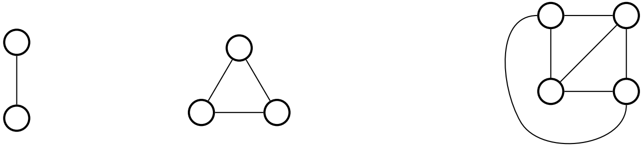
Proof: Not Today!

Graph Types: Complete Graph.



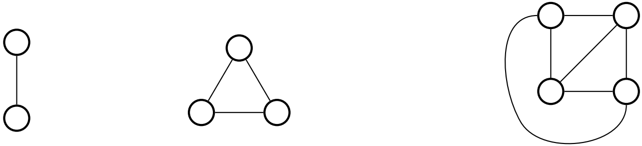


Graph Types: Complete Graph.



K_n, $|V| = n$

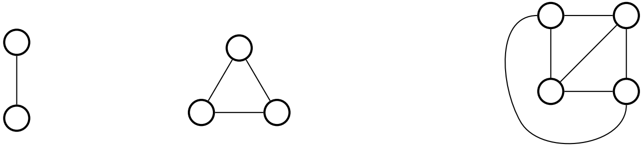
Graph Types: Complete Graph.



K_n, $|V| = n$

every edge present.

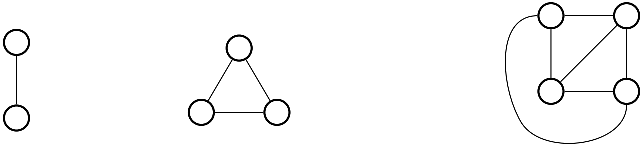
Graph Types: Complete Graph.



K_n, $|V| = n$

every edge present.
degree of vertex?

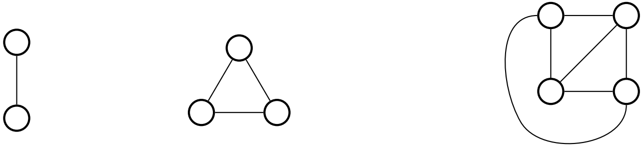
Graph Types: Complete Graph.



K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Graph Types: Complete Graph.

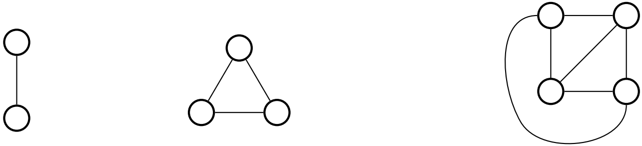


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Very connected.

Graph Types: Complete Graph.

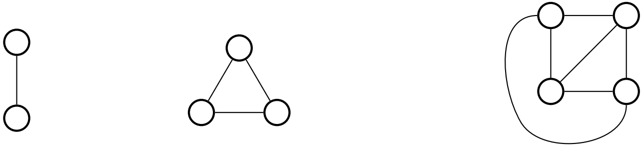


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Very connected.
Lots of edges:

Graph Types: Complete Graph.

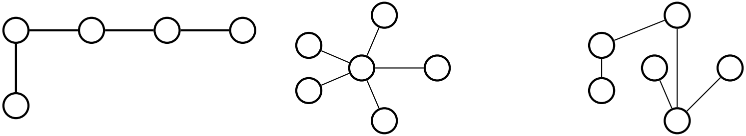


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

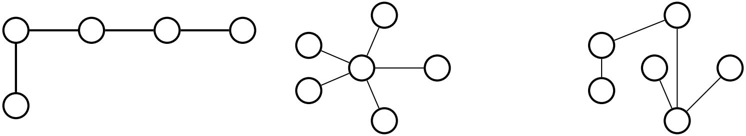
Very connected.
Lots of edges: $n(n-1)/2$.

Trees.



Definitions:

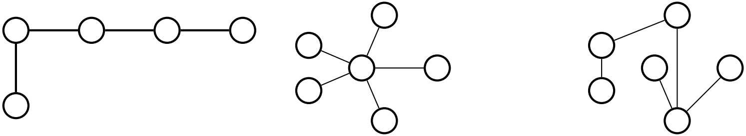
Trees.



Definitions:

A connected graph without a cycle.

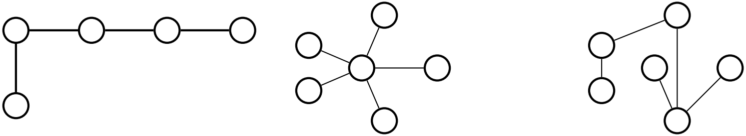
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.

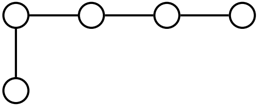
Trees.

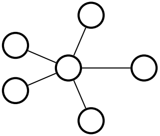


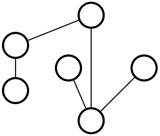
Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.

Trees.



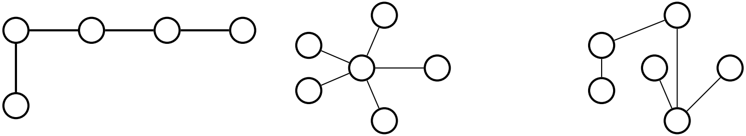




Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

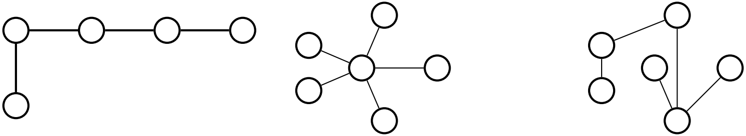
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

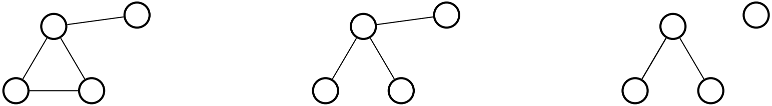
Trees.



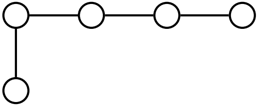
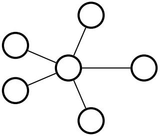
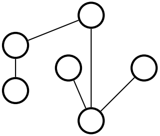
Definitions:

A connected graph without a cycle.
A connected graph with $|V|-1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



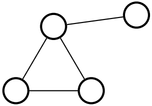
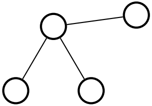
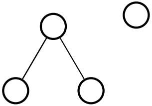
Trees.

Definitions:

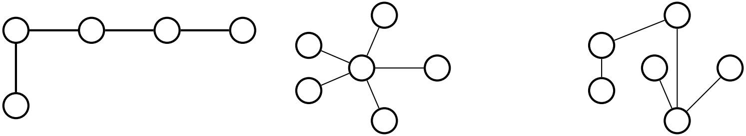
A connected graph without a cycle.
A connected graph with $|V|-1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!

Minimally connected, minimum number of edges to connect.

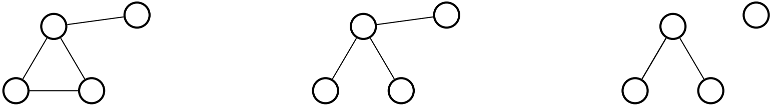
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V|-1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

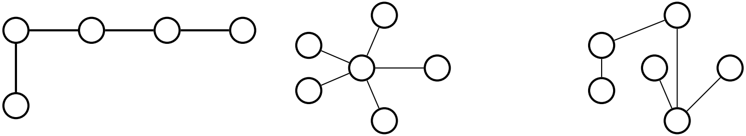
To tree or not to tree!



Minimally connected, minimum number of edges to connect.

Property:

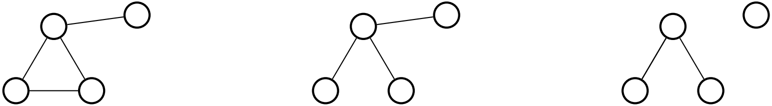
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



Minimally connected, minimum number of edges to connect.

Property:
Can remove a single node and break into components of size at most $|V|/2$.

Hypercube

Hypercubes.

Hypercube

Hypercubes. Really connected.

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0, 1\}^n$,

Hypercube

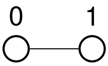
Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

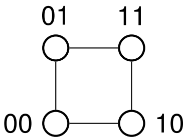
$G = (V, E)$
$|V| = \{0, 1\}^n$,
$|E| = \{(x, y)|x$ and y differ in one bit position.$\}$

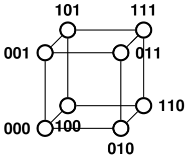
Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0,1\}^n$,
$|E| = \{(x,y) | x$ and y differ in one bit position.$\}$







Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

Recursive Definition.

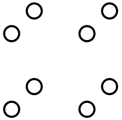
A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.

Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

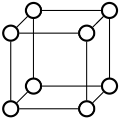
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

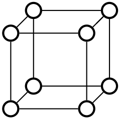
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.

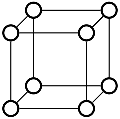


Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.

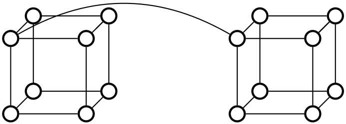




Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

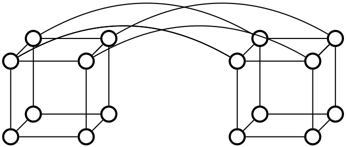
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

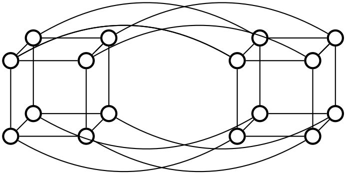
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Hypercube:properties

Rudrata Cycle: cycle that visits every node.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
  Eulerian?

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If *n* is even.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
 Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
  Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
  Best cut?

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
 Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
 Best cut? Cut apart subcubes:

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI:

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
 Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
 Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

 FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
 Get from 000100 to 101000.
 $000100 \rightarrow 100100 \rightarrow 101100 \rightarrow 101000$

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
000100 $\rightarrow$ 100100 $\rightarrow$ 101100 $\rightarrow$ 101000
Correct bits in string, moves along path in hypercube!

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
000100 $\rightarrow$ 100100 $\rightarrow$ 101100 $\rightarrow$ 101000
Correct bits in string, moves along path in hypercube!

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
 Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
 Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

 FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
 Get from 000100 to 101000.
  $000100 \rightarrow 100100 \rightarrow 101100 \rightarrow 101000$
 Correct bits in string, moves along path in hypercube!

Good communication network!

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0,\ldots,m-1\}$
    and integer $i \equiv a \pmod{m}$
      if $i = a + km$ for integer k.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

or if the remainder of i divided by m is a.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

or if the remainder of i divided by m is a.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0,\ldots,m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

or if the remainder of i divided by m is a.

Can do calculations by taking remainders

at the beginning,

in the middle

or at the end.

$58 + 32 = 90 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58+32=90=6 \pmod{7}$
$58+32=2+4=6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \dots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}$

Additive inverses are intuitively negative numbers.

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$?

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5
$5^{-1} \pmod{7}$?

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5
$5^{-1} \pmod{7}$? 3

Inverse Unique?

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$

$axb = bxb = b \pmod n$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$

$axb = bxb = b \pmod n$

$a = b \pmod n$.

$3^{-1} \pmod 6$?

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5
$5^{-1} \pmod{7}$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod{n}$
$ax = bx = 1 \pmod{n}$
$axb = bxb = b \pmod{n}$
$a = b \pmod{n}$.

$3^{-1} \pmod{6}$? No, no, no....

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.

Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.

Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

See,

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

See,... no inverse!

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x, m) = 1$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y,x-y)$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod{y})$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm!

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case?

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y)

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod{y})$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod{y})$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse!

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y,x-y) = gcd(y,x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod m$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y,x-y) = gcd(y,x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax+by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax+bm = ax \pmod m$.

Idea: egcd.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y,x-y) = gcd(y,x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax+by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax+bm = ax \pmod m$.

Idea: egcd.
gcd produces 1

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod m$.

Idea: egcd.
gcd produces 1
by adding and subtracting multiples of x and y

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod{y})$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.
gcd produces 1
by adding and subtracting multiples of x and y

Example: $p = 7$, $q = 11$.

Example: $p = 7$, $q = 11$.

$N = 77$.

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7, 60) = 1$.

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7, 60) = 1$.
  egcd(7,60).

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$7(0) + 60(1) = 60$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \\ 7(-8) + 60(1) &= 4 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \\ 7(-8) + 60(1) &= 4 \\ 7(9) + 60(-1) &= 3 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}
$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \\ 7(9)+60(-1) &= 3 \\ 7(-17)+60(2) &= 1 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \\ 7(9)+60(-1) &= 3 \\ 7(-17)+60(2) &= 1 \end{aligned}$$

Confirm:

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \\ 7(-8) + 60(1) &= 4 \\ 7(9) + 60(-1) &= 3 \\ 7(-17) + 60(2) &= 1 \end{aligned}$$

Confirm: $-119 + 120 = 1$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7, 60) = 1$.
egcd(7,60).

$$\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}$$

Confirm: $-119 + 120 = 1$

$d = e^{-1} = -17 = 43 = \pmod{60}$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function:

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$.
p is prime.
$\implies T = S$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.

Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,
mulitply by inverses to get...

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod p$,

$$a^{p-1} \equiv 1 \pmod p.$$

Proof: Consider $T = \{a \cdot 1 \pmod p, \ldots, a \cdot (p-1) \pmod p\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,
mulitply by inverses to get...

$$a^{(p-1)} \equiv 1 \mod p.$$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.

Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,
mulitply by inverses to get...

$$a^{(p-1)} \equiv 1 \mod p.$$

□

RSA

RSA:

RSA

RSA:

$N = p, q$

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.

RSA

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

RSA

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and $q \implies$ theorem!

$x^{ed} - x$

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and $q \implies$ theorem!

$x^{ed} - x = x^{k(p-1)(q-1)+1} - x$

RSA

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:

$x^{ed} - x$ is divisible by p and $q \implies$ theorem!

$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and $q \implies$ theorem!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

If x is divisible by p, the product is.

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

If x is divisible by p, the product is.
Otherwise $(x^{k(q-1)})^{p-1} = 1 \pmod{p}$ by Fermat.

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and $q \implies$ theorem!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

If x is divisible by p, the product is.
Otherwise $(x^{k(q-1)})^{p-1} = 1 \pmod{p}$ by Fermat.
$\implies (x^{k(q-1)})^{p-1} - 1$ divisible by p.

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod N$

Proof:
$x^{ed} - x$ is divisible by p and $q \implies$ theorem!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

If x is divisible by p, the product is.
Otherwise $(x^{k(q-1)})^{p-1} = 1 \pmod p$ by Fermat.
$\implies (x^{k(q-1)})^{p-1} - 1$ divisible by p.

Similarly for q.

RSA

RSA:

$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and $q \implies$ theorem!

$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$

If x is divisible by p, the product is.
Otherwise $(x^{k(q-1)})^{p-1} = 1 \pmod{p}$ by Fermat.
$\implies (x^{k(q-1)})^{p-1} - 1$ divisible by p.

Similarly for q. $\square$

RSA, Public Key, and Signatures.

RSA, Public Key, and Signatures.

RSA:

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m, K), k) = (m^e)^d \mod N = m$.

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

RSA, Public Key, and Signatures.

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

$S(C) = D(C)$.

RSA, Public Key, and Signatures.

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

$S(C) = D(C)$.
Announce $(C, S(C))$

RSA, Public Key, and Signatures.

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

$S(C) = D(C)$.
Announce $(C, S(C))$

RSA, Public Key, and Signatures.

RSA:

$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

$S(C) = D(C)$.
Announce $(C, S(C))$

Verify: Check $C = E(C)$.

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

$S(C) = D(C)$.

Announce $(C, S(C))$

Verify: Check $C = E(C)$.

$E(D(C,k),K) = (C^d)^e = C \pmod{N}$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x, m) = 1$.

Group structures more generally.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y)$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod{y})$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod{y})$.

Give recursive Algorithm!

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case?

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y,x-y) = gcd(y,x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y)

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y,x-y) = gcd(y,x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m)=1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m)=1$.

Finding gcd.
$gcd(x,y)=gcd(y,x-y)=gcd(y,x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0)=x$.

Extended-gcd(x,y) returns (d,a,b)
$d=gcd(x,y)$ and $d=ax+by$

Multiplicative inverse of (x,m).
egcd$(x,m)=(1,a,b)$
a is inverse!

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm$

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y,x-y) = gcd(y,x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod m$.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd(x,m) $= (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod m$.

Idea: egcd.

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod{y})$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd(x,m) $= (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.
gcd produces 1

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x,m) = 1$.

Group structures more generally.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod m$.

Idea: egcd.
gcd produces 1
by adding and subtracting multiples of x and y

Example: $p = 7$, $q = 11$.

Example: $p = 7$, $q = 11$.

$N = 77$.

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7, 60) = 1$.

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7, 60) = 1$.
egcd(7,60).

$$7(0) + 60(1) \quad = \quad 60$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77.$
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \\ 7(-8) + 60(1) &= 4 \\ 7(9) + 60(-1) &= 3 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}
$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}
$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7, 60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}
$$

Confirm:

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \\ 7(9)+60(-1) &= 3 \\ 7(-17)+60(2) &= 1 \end{aligned}$$

Confirm: $-119+120 = 1$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}
$$

Confirm: $-119 + 120 = 1$

$d = e^{-1} = -17 = 43 = \pmod{60}$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$a^{p-1} \equiv 1 \pmod{p}$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function:

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$a^{p-1} \equiv 1 \pmod{p}$.

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.

Invertible function: one-to-one.

$T \subseteq S$ since $0 \notin T$.

p is prime.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \not\in T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,
mulitply by inverses to get...

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.

Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,
mulitply by inverses to get...

$$a^{(p-1)} \equiv 1 \mod p.$$

Fermat from Bijection.

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Proof: Consider $T = \{a \cdot 1 \pmod{p}, \ldots, a \cdot (p-1) \pmod{p}\}$.

T is range of function $f(x) = ax \mod (p)$ for set $S = \{1, \ldots, p-1\}$.
Invertible function: one-to-one.
$T \subseteq S$ since $0 \notin T$.
p is prime.
$\implies T = S$.
Product of elts of T = Product of elts of S.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,
mulitply by inverses to get...

$$a^{(p-1)} \equiv 1 \mod p. \qquad \square$$

RSA

RSA:

RSA

RSA:

$N = p, q$

RSA

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

RSA

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

RSA

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

$x^{ed} - x$

RSA

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:

$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

$x^{ed} - x = x^{k(p-1)(q-1)+1} - x$

RSA

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:

$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

If x is divisible by p, the product is.

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

If x is divisible by p, the product is.
Otherwise $(x^{k(q-1)})^{p-1} = 1 \pmod{p}$ by Fermat.

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and $q \implies$ theorem!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

If x is divisible by p, the product is.
Otherwise $(x^{k(q-1)})^{p-1} = 1 \pmod{p}$ by Fermat.
$\implies (x^{k(q-1)})^{p-1} - 1$ divisible by p.

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

If x is divisible by p, the product is.
Otherwise $(x^{k(q-1)})^{p-1} = 1 \pmod{p}$ by Fermat.
$\implies (x^{k(q-1)})^{p-1} - 1$ divisible by p.

Similarly for q.

RSA

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Proof:
$x^{ed} - x$ is divisible by p and q $\implies$ theorem!

$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$

If x is divisible by p, the product is.
Otherwise $(x^{k(q-1)})^{p-1} = 1 \pmod{p}$ by Fermat.
$\implies (x^{k(q-1)})^{p-1} - 1$ divisible by p.

Similarly for q. □

RSA, Public Key, and Signatures.

RSA, Public Key, and Signatures.

RSA:

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e,(p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

RSA, Public Key, and Signatures.

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1))$.

$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

RSA, Public Key, and Signatures.

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K), k) = (m^e)^d \mod N = m$.

Signature scheme:

$S(C) = D(C)$.

RSA, Public Key, and Signatures.

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m.$

Signature scheme:

$S(C) = D(C)$.
Announce $(C, S(C))$

RSA, Public Key, and Signatures.

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m.$

Signature scheme:

$S(C) = D(C)$.
Announce $(C, S(C))$

RSA, Public Key, and Signatures.

RSA:
$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

$S(C) = D(C)$.
Announce $(C, S(C))$

Verify: Check $C = E(C)$.

RSA, Public Key, and Signatures.

RSA:

$N = p, q$
e with $\gcd(e, (p-1)(q-1))$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

$S(C) = D(C)$.
Announce $(C, S(C))$

Verify: Check $C = E(C)$.

$E(D(C,k),K) = (C^d)^e = C \pmod{N}$

Fermat/RSA

$3^6 \pmod 7$?

Fermat/RSA

$3^6 \pmod 7$? 1.

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$?

Fermat/RSA

$3^6 \pmod{7}$? 1.   Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod{7}$? 1.
$3^{60} \pmod{7}$?

Fermat/RSA

$3^6 \pmod 7$? 1.   Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

$2^{12} \pmod{21}$?

Fermat/RSA

$3^6 \pmod 7$? 1.   Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

$2^{12} \pmod{21}$? 1.
$21 = (3)(7)$

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

$2^{12} \pmod{21}$? 1.
$21 = (3)(7)$ $(p-1)(q-1) = (2)(6) = 12$

Fermat/RSA

$3^6 \pmod 7$? 1.   Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

$2^{12} \pmod{21}$? 1.
$21 = (3)(7)$ $(p-1)(q-1) = (2)(6) = 12$
$gcd(2,12) = 1$, $x^{(p-1)(q-1)} = 1 \pmod{pq}$

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

$2^{12} \pmod{21}$? 1.
$21 = (3)(7)$ $(p-1)(q-1) = (2)(6) = 12$
$gcd(2, 12) = 1$, $x^{(p-1)(q-1)} = 1 \pmod{pq}$ $2^{12} = 1 \pmod{21}$.

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

$2^{12} \pmod{21}$? 1.
$21 = (3)(7)$ $(p-1)(q-1) = (2)(6) = 12$
$gcd(2, 12) = 1$, $x^{(p-1)(q-1)} = 1 \pmod{pq}$ $2^{12} = 1 \pmod{21}$.

$2^14 \pmod{21}$?

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

$2^{12} \pmod{21}$? 1.
$21 = (3)(7)$ $(p-1)(q-1) = (2)(6) = 12$
$gcd(2,12) = 1$, $x^{(p-1)(q-1)} = 1 \pmod{pq}$ $2^{12} = 1 \pmod{21}$.

$2^1 4 \pmod{21}$? 4.

Fermat/RSA

$3^6 \pmod 7$? 1. Fermat: $p = 7$, $p - 1 = 6$
$3^{18} \pmod 7$? 1.
$3^{60} \pmod 7$? 1.
$3^{61} \pmod 7$?3.

$2^{12} \pmod{21}$? 1.
$21 = (3)(7)$ $(p-1)(q-1) = (2)(6) = 12$
$gcd(2,12) = 1$, $x^{(p-1)(q-1)} = 1 \pmod{pq}$ $2^{12} = 1 \pmod{21}$.

$2^14 \pmod{21}$? 4. Technically $4 \pmod{21}$.

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x - r_1) \cdots (x - r_k) Q(x)$.

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:

Any polynomial with roots $r_1, \ldots, r_k$.

written as $(x - r_1) \cdots (x - r_k) Q(x)$.

using polynomial division.

Degree at least the number of roots.

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x - r_1) \cdots (x - r_k) Q(x)$.
using polynomial division.
Degree at least the number of roots. $\square$

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x - r_1) \cdots (x - r_k) Q(x)$.
using polynomial division.
Degree at least the number of roots. $\square$

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d + 1$:
$(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x - r_1) \cdots (x - r_k)Q(x)$.
using polynomial division.
Degree at least the number of roots. □

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$:
$(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Proof Ideas:

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x - r_1) \cdots (x - r_k)Q(x)$.
using polynomial division.
Degree at least the number of roots. □

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$:
$(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Proof Ideas:

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x-r_1)\cdots(x-r_k)Q(x)$.
using polynomial division.
Degree at least the number of roots. □

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$:
$(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Proof Ideas:
Lagrange Interpolation gives existence.

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x - r_1) \cdots (x - r_k) Q(x)$.
using polynomial division.
Degree at least the number of roots. $\square$

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$:
$(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Proof Ideas:
Lagrange Interpolation gives existence.
Property 1 gives uniqueness.

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x - r_1) \cdots (x - r_k) Q(x)$.
using polynomial division.
Degree at least the number of roots. □

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Proof Ideas:
Lagrange Interpolation gives existence.
Property 1 gives uniqueness. □

Polynomials

Property 1: Any degree d polynomial over a field has at most d roots.

Proof Idea:
Any polynomial with roots $r_1, \ldots, r_k$.
written as $(x - r_1) \cdots (x - r_k)Q(x)$.
using polynomial division.
Degree at least the number of roots. □

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$:
$(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Proof Ideas:
Lagrange Interpolation gives existence.
Property 1 gives uniqueness. □

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
 Scheme: degree $n-1$ polynomial, $P(x)$.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.

Scheme: degree $n-1$ polynomial, $P(x)$.

Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1,y_1),\ldots,(x_{d+1},y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1,P(1)),\ldots(n,P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+k-1, P(n+k-1))$.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+k-1, P(n+k-1))$.
Recover Message: Any n packets are cool by property 2.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+k-1, P(n+k-1))$.
Recover Message: Any n packets are cool by property 2.

Corruptions Coding: n packets, k corruptions.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+k-1, P(n+k-1))$.
Recover Message: Any n packets are cool by property 2.

Corruptions Coding: n packets, k corruptions.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1,y_1),\ldots,(x_{d+1},y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1,P(1)),\ldots(n,P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0)=m_0, P(1)=m_1,\ldots P(n-1)=m_{n-1}$
Send: $(0,P(0)),\ldots(n+k-1,P(n+k-1))$.
Recover Message: Any n packets are cool by property 2.

Corruptions Coding: n packets, k corruptions.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0)=m_0, P(1)=m_1,\ldots P(n-1)=m_{n-1}$

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+k-1, P(n+k-1))$.
Recover Message: Any n packets are cool by property 2.

Corruptions Coding: n packets, k corruptions.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+2k-1, P(n+2k-1))$.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+k-1, P(n+k-1))$.
Recover Message: Any n packets are cool by property 2.

Corruptions Coding: n packets, k corruptions.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+2k-1, P(n+2k-1))$.
Recovery:

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
Scheme: degree $n-1$ polynomial, $P(x)$.
Secret: $P(0)$ **Shares:** $(1, P(1)), \ldots (n, P(n))$.
Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+k-1, P(n+k-1))$.
Recover Message: Any n packets are cool by property 2.

Corruptions Coding: n packets, k corruptions.
Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
Message: $P(0) = m_0, P(1) = m_1, \ldots P(n-1) = m_{n-1}$
Send: $(0, P(0)), \ldots (n+2k-1, P(n+2k-1))$.
Recovery: $P(x)$ is only consistent polynomial with $n+k$ points.

Applications.

Property 2: There is exactly 1 polynomial of degree $\leq d$ with arithmetic modulo prime p that contains any $d+1$: $(x_1,y_1),\ldots,(x_{d+1},y_{d+1})$ with x_i distinct.

Secret Sharing: k out of n people know secret.
 Scheme: degree $n-1$ polynomial, $P(x)$.
 Secret: $P(0)$ **Shares:** $(1,P(1)),\ldots(n,P(n))$.
 Recover Secret: Reconstruct $P(x)$ with any k points.

Erasure Coding: n packets, k losses.
 Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
 Message: $P(0)=m_0, P(1)=m_1,\ldots P(n-1)=m_{n-1}$
 Send: $(0,P(0)),\ldots(n+k-1,P(n+k-1))$.
Recover Message: Any n packets are cool by property 2.

Corruptions Coding: n packets, k corruptions.
 Scheme: degree $n-1$ polynomial, $P(x)$. Reed-Solomon.
 Message: $P(0)=m_0, P(1)=m_1,\ldots P(n-1)=m_{n-1}$
 Send: $(0,P(0)),\ldots(n+2k-1,P(n+2k-1))$.
Recovery: $P(x)$ is only consistent polynomial with $n+k$ points.
 Property 2 and pigeonhole principle.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod p$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0.$$

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.
Gives system of $n+2k$ linear equations.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.
Gives system of $n+2k$ linear equations.

$$a_{n+k-1} + \ldots a_0 \equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p}$$

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0.$
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0.$

Gives system of $n+2k$ linear equations.

$$\begin{aligned} a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\ a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\ &\vdots \end{aligned}$$

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.

Gives system of $n+2k$ linear equations.

$$
\begin{aligned}
a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\
a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\
&\vdots \\
a_{n+k-1}(m)^{n+k-1} + \ldots a_0 &\equiv R(m)((m)^k + b_{k-1}(m)^{k-1} \cdots b_0) \pmod{p}
\end{aligned}
$$

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.

Gives system of $n+2k$ linear equations.

$$
\begin{aligned}
a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\
a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\
&\vdots \\
a_{n+k-1}(m)^{n+k-1} + \ldots a_0 &\equiv R(m)((m)^k + b_{k-1}(m)^{k-1} \cdots b_0) \pmod{p}
\end{aligned}
$$

..and $n+2k$ unknown coefficients of $Q(x)$ and $E(x)$!

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.
Gives system of $n+2k$ linear equations.

$$
\begin{aligned}
a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\
a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\
&\vdots \\
a_{n+k-1}(m)^{n+k-1} + \ldots a_0 &\equiv R(m)((m)^k + b_{k-1}(m)^{k-1} \cdots b_0) \pmod{p}
\end{aligned}
$$

..and $n+2k$ unknown coefficients of $Q(x)$ and $E(x)$!

Solve for coefficients of $Q(x)$ and $E(x)$.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0.$
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0.$

Gives system of $n+2k$ linear equations.

$$
\begin{aligned}
a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\
a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\
&\vdots \\
a_{n+k-1}(m)^{n+k-1} + \ldots a_0 &\equiv R(m)((m)^k + b_{k-1}(m)^{k-1} \cdots b_0) \pmod{p}
\end{aligned}
$$

..and $n+2k$ unknown coefficients of $Q(x)$ and $E(x)$!

Solve for coefficients of $Q(x)$ and $E(x)$.

Find $P(x) = Q(x)/E(x)$.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0.$
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0.$

Gives system of $n+2k$ linear equations.

$$\begin{aligned}
a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\
a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\
&\vdots \\
a_{n+k-1}(m)^{n+k-1} + \ldots a_0 &\equiv R(m)((m)^k + b_{k-1}(m)^{k-1} \cdots b_0) \pmod{p}
\end{aligned}$$

..and $n+2k$ unknown coefficients of $Q(x)$ and $E(x)$!

Solve for coefficients of $Q(x)$ and $E(x)$.

Find $P(x) = Q(x)/E(x)$.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \dots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod p$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0.$
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0.$

Gives system of $n+2k$ linear equations.

$$\begin{aligned}
a_{n+k-1} + \dots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod p \\
a_{n+k-1}(2)^{n+k-1} + \dots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod p \\
&\vdots \\
a_{n+k-1}(m)^{n+k-1} + \dots a_0 &\equiv R(m)((m)^k + b_{k-1}(m)^{k-1} \cdots b_0) \pmod p
\end{aligned}$$

..and $n+2k$ unknown coefficients of $Q(x)$ and $E(x)$!

Solve for coefficients of $Q(x)$ and $E(x)$.

Find $P(x) = Q(x)/E(x)$.

Welsh-Berlekamp

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n+2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$
since $E(i) = 0$ at points where there are errors.
Let $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.
Gives system of $n+2k$ linear equations.

$$
\begin{aligned}
a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\
a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\
&\vdots \\
a_{n+k-1}(m)^{n+k-1} + \ldots a_0 &\equiv R(m)((m)^k + b_{k-1}(m)^{k-1} \cdots b_0) \pmod{p}
\end{aligned}
$$

..and $n+2k$ unknown coefficients of $Q(x)$ and $E(x)$!

Solve for coefficients of $Q(x)$ and $E(x)$.

Find $P(x) = Q(x)/E(x)$.

Counting

First Rule

Counting

First Rule
Second Rule

Counting

First Rule
Second Rule
Stars/Bars

Counting

First Rule
Second Rule
Stars/Bars
Common Scenarios: Sampling, Balls in Bins.

Counting

First Rule
Second Rule
Stars/Bars
Common Scenarios: Sampling, Balls in Bins.
Sum Rule. Inclusion/Exclusion.

Counting

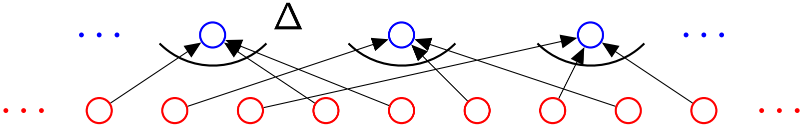
First Rule
Second Rule
Stars/Bars
Common Scenarios: Sampling, Balls in Bins.
Sum Rule. Inclusion/Exclusion.
Combinatorial Proofs.

Counting

First Rule
Second Rule
Stars/Bars
Common Scenarios: Sampling, Balls in Bins.
Sum Rule. Inclusion/Exclusion.
Combinatorial Proofs.

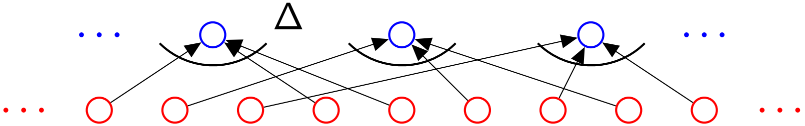
Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.

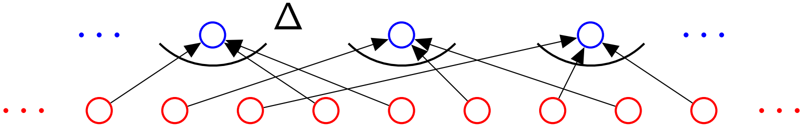


3 card Poker deals: 52

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**

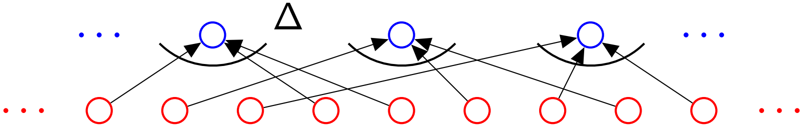
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: 52×51

Example: visualize.

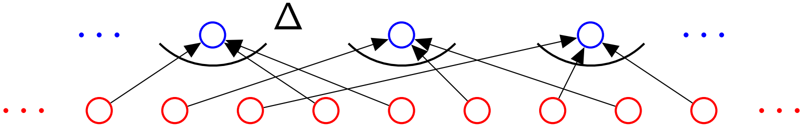
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50$

Example: visualize.

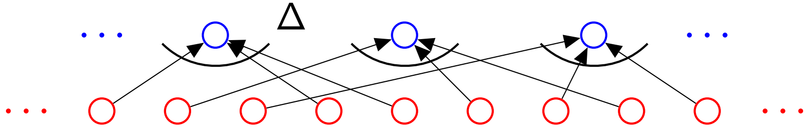
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$.

Example: visualize.

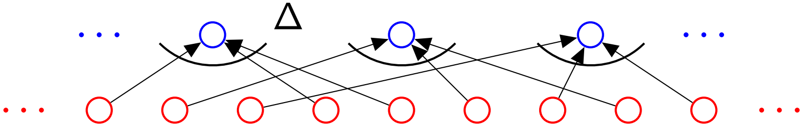
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.

Example: visualize.

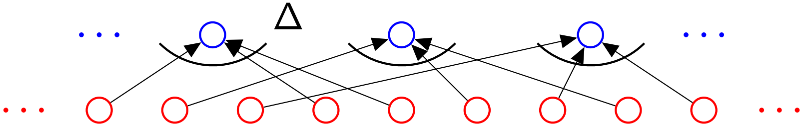
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

Example: visualize.

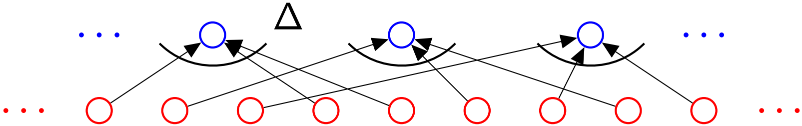
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.

Example: visualize.

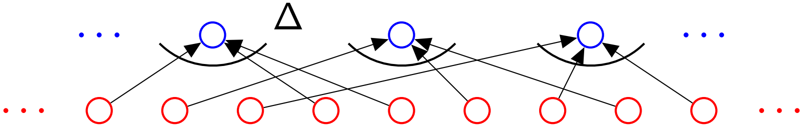
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
  Hand: Q, K, A.
  Deals: Q, K, A,

Example: visualize.

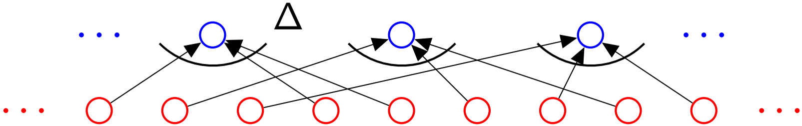
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K,

Example: visualize.

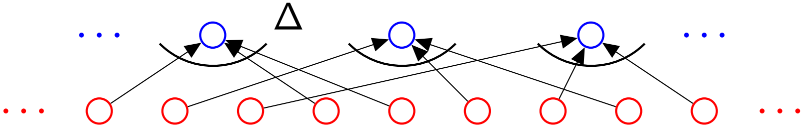
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.

Example: visualize.

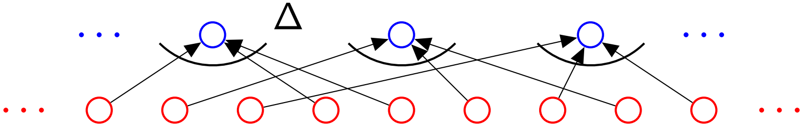
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q,K,A, Q,A,K, K,A,Q,K,A,Q, A,K,Q, A,Q,K.
$\Delta = 3 \times 2 \times 1$

Example: visualize.

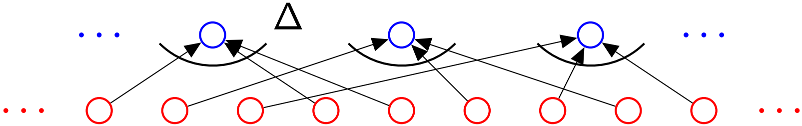
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.

Example: visualize.

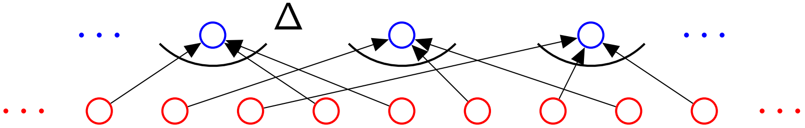
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total:

Example: visualize.

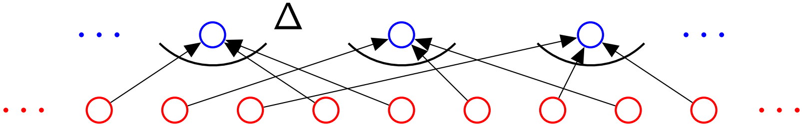
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$

Example: visualize.

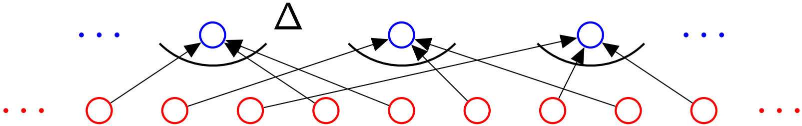
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

Hand: Q, K, A.

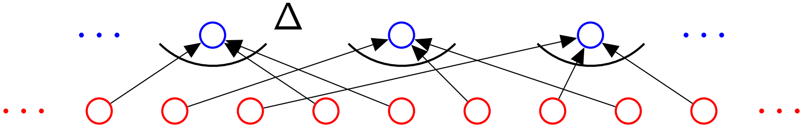
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.

$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.

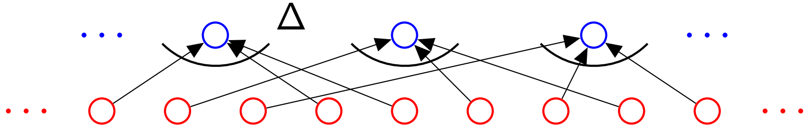


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
  Hand: Q, K, A.
  Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
  Ordered set: $\frac{n!}{(n-k)!}$

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.

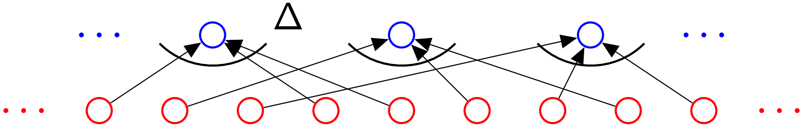


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
  Hand: Q, K, A.
  Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
  Ordered set: $\frac{n!}{(n-k)!}$
  What is Δ?

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

Hand: Q, K, A.

Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.

$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

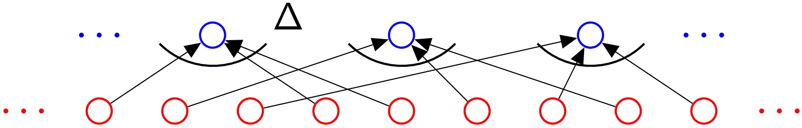
Choose k out of n.

Ordered set: $\frac{n!}{(n-k)!}$

What is Δ? $k!$

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.

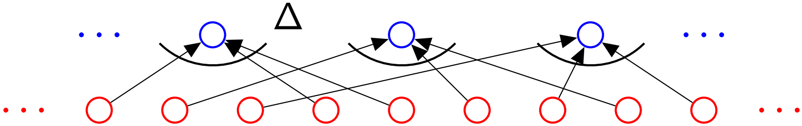


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
Ordered set: $\frac{n!}{(n-k)!}$
What is Δ? $k!$ First rule again.

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.

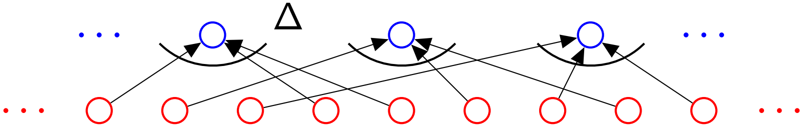


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q,K,A, Q,A,K, K,A,Q,K,A,Q, A,K,Q, A,Q,K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
Ordered set: $\frac{n!}{(n-k)!}$
What is Δ? $k!$ First rule again.
$\implies$ Total: $\frac{n!}{(n-k)!k!}$

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.

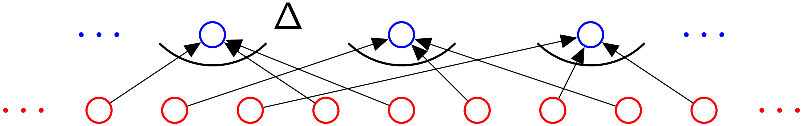


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
Ordered set: $\frac{n!}{(n-k)!}$
What is Δ? $k!$ First rule again.
$\implies$ Total: $\frac{n!}{(n-k)!k!}$ Second rule.

Example: visualize.

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.

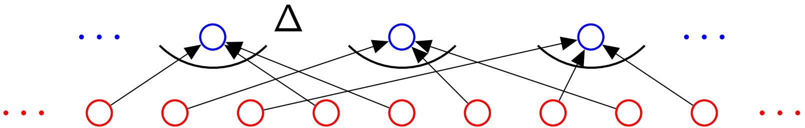


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q,K,A, Q,A,K, K,A,Q,K,A,Q, A,K,Q, A,Q,K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.
Ordered set: $\frac{n!}{(n-k)!}$
What is Δ? $k!$ First rule again.
$\Longrightarrow$ Total: $\frac{n!}{(n-k)!k!}$ Second rule.

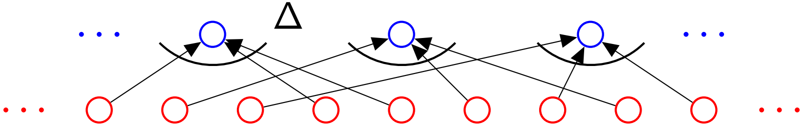
Example: visualize

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Example: visualize

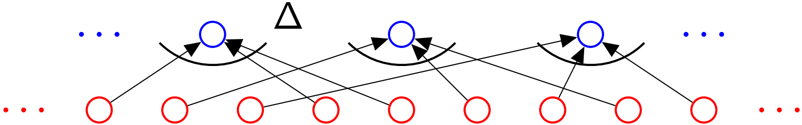
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?

Example: visualize

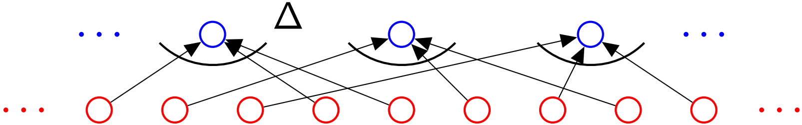
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7!

Example: visualize

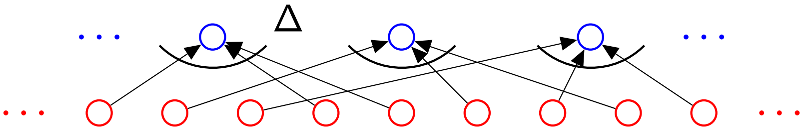
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.

Example: visualize

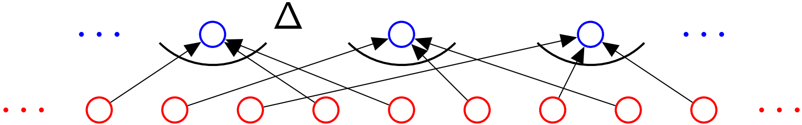
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!

Example: visualize

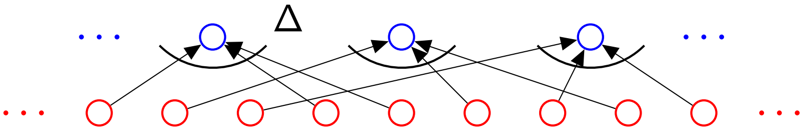
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?

Example: visualize

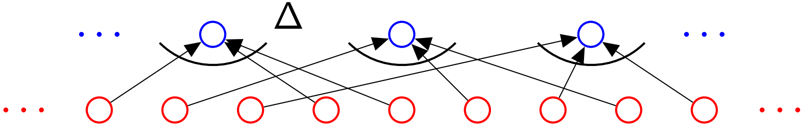
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?
ANAGRAM

Example: visualize

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

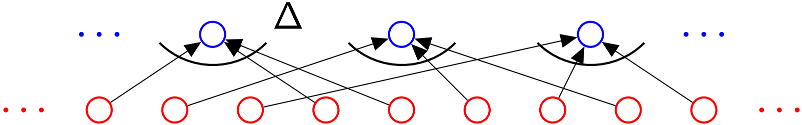
What is Δ?

ANAGRAM

$A_1NA_2GRA_3M$,

Example: visualize

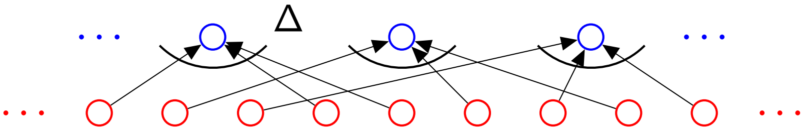
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?
ANAGRAM
$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$,

Example: visualize

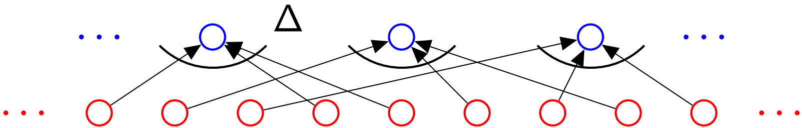
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?
ANAGRAM
$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

Example: visualize

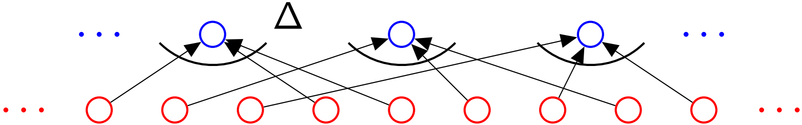
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?
ANAGRAM
$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...
$\Delta = 3 \times 2 \times 1$

Example: visualize

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

What is Δ?

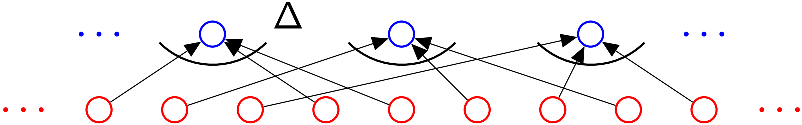
ANAGRAM

$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

$\Delta = 3 \times 2 \times 1 = 3!$

Example: visualize

First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?

Ordered Set: 7! First rule.

A's are the same!

What is Δ?

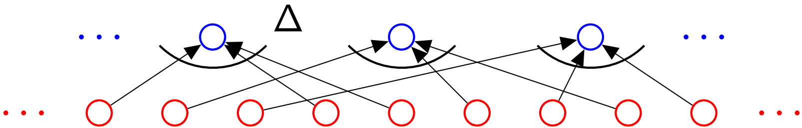
ANAGRAM

$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...

$\Delta = 3 \times 2 \times 1 = 3!$ First rule!

Example: visualize

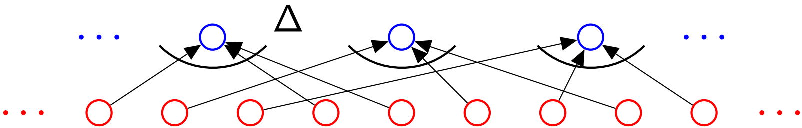
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?
ANAGRAM
$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...
$\Delta = 3 \times 2 \times 1 = 3!$ First rule!
$\implies \frac{7!}{3!}$

Example: visualize

First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



Orderings of ANAGRAM?
Ordered Set: 7! First rule.
A's are the same!
What is Δ?
ANAGRAM
$A_1NA_2GRA_3M$, $A_2NA_1GRA_3M$, ...
$\Delta = 3 \times 2 \times 1 = 3!$ First rule!
$\implies \frac{7!}{3!}$ Second rule!

Summary.

k Samples with replacement from n items: n^k.

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Sample without replacement and order doesn't matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
"n choose k"

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Sample without replacement and order doesn't matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
"n choose k"
(Count using first rule and second rule.)

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Sample without replacement and order doesn't matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
"n choose k"
(Count using first rule and second rule.)

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Sample without replacement and order doesn't matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
"n choose k"
(Count using first rule and second rule.)

Sample with replacement and order doesn't matter: $\binom{k+n-1}{n-1}$.

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Sample without replacement and order doesn't matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
"n choose k"
(Count using first rule and second rule.)

Sample with replacement and order doesn't matter: $\binom{k+n-1}{n-1}$.

Count with stars and bars:

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Sample without replacement and order doesn't matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
"n choose k"
(Count using first rule and second rule.)

Sample with replacement and order doesn't matter: $\binom{k+n-1}{n-1}$.

Count with stars and bars:
how many ways to add up n numbers to get k.

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Sample without replacement and order doesn't matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
"n choose k"
(Count using first rule and second rule.)

Sample with replacement and order doesn't matter: $\binom{k+n-1}{n-1}$.

Count with stars and bars:
how many ways to add up n numbers to get k.
Each number is number of samples of type i

Summary.

k Samples with replacement from n items: n^k.
Sample without replacement: $\frac{n!}{(n-k)!}$

Sample without replacement and order doesn't matter: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.
"n choose k"
(Count using first rule and second rule.)

Sample with replacement and order doesn't matter: $\binom{k+n-1}{n-1}$.

Count with stars and bars:
how many ways to add up n numbers to get k.
Each number is number of samples of type i which adds to total, k.

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

Balls in bins.

“k Balls in n bins” $\equiv$ “k samples from n possibilities.”

“indistinguishable balls” $\equiv$ “order doesn’t matter”

Balls in bins.

“k Balls in n bins” $\equiv$ “k samples from n possibilities.”

“indistinguishable balls” $\equiv$ “order doesn’t matter”

“only one ball in each bin” $\equiv$ “without replacement”

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins
5 samples from 10 possibilities with replacement

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins
5 samples from 10 possibilities with replacement
Example: 5 digit numbers.

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins
5 samples from 10 possibilities with replacement
Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins
5 samples from 10 possibilities with replacement
Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins
5 samples from 10 possibilities with replacement
Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
Example: Poker hands.

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins
5 samples from 10 possibilities with replacement
Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
Example: Poker hands.

5 indistinguishable balls into 3 bins

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins
5 samples from 10 possibilities with replacement
Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
Example: Poker hands.

5 indistinguishable balls into 3 bins
5 samples from 3 possibilities with replacement and no order

Balls in bins.

"k Balls in n bins" $\equiv$ "k samples from n possibilities."

"indistinguishable balls" $\equiv$ "order doesn't matter"

"only one ball in each bin" $\equiv$ "without replacement"

5 balls into 10 bins
5 samples from 10 possibilities with replacement
Example: 5 digit numbers.

5 indistinguishable balls into 52 bins only one ball in each bin
5 samples from 52 possibilities without replacement
Example: Poker hands.

5 indistinguishable balls into 3 bins
5 samples from 3 possibilities with replacement and no order
Dividing 5 dollars among Alice, Bob and Eve.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)!$

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S **and** T**,**
$|S \cup T| = |S| + |T| - |S \cap T|$.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit. $|S| = 10^9$

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S **and** T**,**
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit. $|S| = 10^9$

T = phone numbers with 7 as second digit. $|T| = 10^9$.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S **and** T**,** $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S **and** T**,**
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit. $|T| = 10^9$.

$S \cap T$ = phone numbers with 7 as first and second digit.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit. $|T| = 10^9$.

$S \cap T$ = phone numbers with 7 as first and second digit. $|S \cap T| = 10^8$.

Simple Inclusion/Exclusion

Sum Rule: For disjoint sets S and T, $|S \cup T| = |S| + |T|$

Example: How many permutations of n items start with 1 or 2?
$1 \times (n-1)! + 1 \times (n-1)!$

Inclusion/Exclusion Rule: For any S and T,
$|S \cup T| = |S| + |T| - |S \cap T|$.

Example: How many 10-digit phone numbers have 7 as their first or second digit?

S = phone numbers with 7 as first digit.$|S| = 10^9$

T = phone numbers with 7 as second digit. $|T| = 10^9$.

$S \cap T$ = phone numbers with 7 as first and second digit. $|S \cap T| = 10^8$.

Answer: $|S| + |T| - |S \cap T| = 10^9 + 10^9 - 10^8$.

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$?

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.
Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element,

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.
$\implies \binom{n}{k-1}$

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.
$\implies \binom{n}{k-1}$

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.
$\implies \binom{n}{k-1}$

How many don't contain the first element ?

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.
$\implies \binom{n}{k-1}$

How many don't contain the first element ?
Need to choose k elements from remaining n elts.

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.
Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.
$\implies \binom{n}{k-1}$

How many don't contain the first element ?
Need to choose k elements from remaining n elts.
$\implies \binom{n}{k}$

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.
$\implies \binom{n}{k-1}$

How many don't contain the first element ?
Need to choose k elements from remaining n elts.
$\implies \binom{n}{k}$

Combinatorial Proofs.

Theorem: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

Proof: How many size k subsets of $n+1$? $\binom{n+1}{k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.
$\implies \binom{n}{k-1}$

How many don't contain the first element ?
Need to choose k elements from remaining n elts.
$\implies \binom{n}{k}$

So, $\binom{n}{k-1} + \binom{n}{k}$

Combinatorial Proofs.

Theorem: ${n+1 \choose k} = {n \choose k} + {n \choose k-1}$.

Proof: How many size k subsets of $n+1$? ${n+1 \choose k}$.

How many size k subsets of $n+1$?
How many contain the first element?
Chose first element, need to choose $k-1$ more from remaining n elements.
$\implies {n \choose k-1}$

How many don't contain the first element ?
Need to choose k elements from remaining n elts.
$\implies {n \choose k}$

So, ${n \choose k-1} + {n \choose k} = {n+1 \choose k}$. □

Countability

Isomporphism principle.

Countability

Isomporphism principle.
Example.

Countability

Isomporphism principle.
Example.
Countability.

Countability

Isomporphism principle.
Example.
Countability.
Diagonalization.

Isomorphism principle.

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

Isomorphism principle.

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

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, $f : D \to R$.
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, $f : D \rightarrow R$.

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 $|D| = |R|$.

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?

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

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

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?

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

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

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

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

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?

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

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

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

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 $[0,1/2]$ different mult inverses $\implies f(x) \neq f(y)$.

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 $[0,1/2]$ different mult inverses $\implies f(x) \neq f(y)$.
If one is in $[0,1/2]$ and one isn't,

Cardinalities of uncountable sets?

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

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

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

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

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

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

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

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

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

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

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 $[0,1/2]$ different mult inverses $\implies f(x) \neq f(y)$.
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!

Countable.

Countable.

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

Countable.

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.

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**.

Countable.

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**.

Bijection to or from natural numbers implies countably infinite.

Countable.

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**.

Bijection to or from natural numbers implies countably infinite.

Enumerable means countable.

Countable.

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**.

Bijection to or from natural numbers implies countably infinite.

Enumerable means countable.

Subset of countable set is countable.

Countable.

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**.

Bijection to or from natural numbers implies countably infinite.

Enumerable means countable.

Subset of countable set is countable.

All countably infinite sets are the same cardinality as each other.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1?

Examples

Countably infinite (same cardinality as naturals)

- Z^{+} - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0,$

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0, -1,$

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0, -1, 1,$

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0, -1, 1, -2,$

Examples

Countably infinite (same cardinality as naturals)

- Z^+ - positive integers
  Where's 0?
  Bijection: $f(z) = z - 1$.
  (Where's 0? 1 Where's 1? 2 ...)
- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Bijection: $f(z) = 2|z| - sign(z)$.
  Enumerate: $0, -1, 1, -2, 2...$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0), (0,1), (0,2), \ldots$???
  Never get to $(1,1)$!

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0,

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0, first positive,

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0, first positive, first negative,

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0, first positive, first negative, second positive..

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Square of countably infinite?
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!
  Enumerate: $(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\ldots$
  (a,b) at position $(a+b-1)(a+b)/2+b$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.
  Countably infinite.
- All rational numbers.
  Enumerate: list 0, positive and negative. How?
  Enumerate: 0, first positive, first negative, second positive..
  Will eventually get to any rational.

Diagonalization: power set of Integers.

The set of all subsets of N.

Diagonalization: power set of Integers.

The set of all subsets of N.

Assume is countable.

Diagonalization: power set of Integers.

The set of all subsets of N.

Assume is countable.

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

Diagonalization: power set of Integers.

The set of all subsets of N.

Assume is countable.

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

Define a diagonal set, D:

Diagonalization: power set of Integers.

The set of all subsets of N.

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, $i \in D$.

Diagonalization: power set of Integers.

The set of all subsets of N.

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 $i \notin D$.

Diagonalization: power set of Integers.

The set of all subsets of N.

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 $i \notin D$.

Diagonalization: power set of Integers.

The set of all subsets of N.

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.

Diagonalization: power set of Integers.

The set of all subsets of N.

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.

Diagonalization: power set of Integers.

The set of all subsets of N.

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.

Diagonalization: power set of Integers.

The set of all subsets of N.

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.

Diagonalization: power set of Integers.

The set of all subsets of N.

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.

Diagonalization: power set of Integers.

The set of all subsets of N.

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.

Diagonalization: power set of Integers.

The set of all subsets of N.

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.)

Uncomputability.

Halting problem is undecibable.

Uncomputability.

Halting problem is undecibable.

Diagonalization.

Uncomputability.

Halting problem is undecibable.

Diagonalization.

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! .. □

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.

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.

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.

Either way is 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.

Either way is contradiction. Program HALT does not exist! $\square$

Another view: diagonalization.

Any program is a fixed length string.

Another view: diagonalization.

Any program is a fixed length string.
Fixed length strings are enumerable.

Another view: 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: 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	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Another view: 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.

Another view: 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.

Another view: 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: 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.

Another view: 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: 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: 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!

Undecidable problems.

Does a program print “Hello World”?

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Undecidable problems.

Does a program print "Hello World"?
Find exit points and add statement: **Print** "Hello World."

Undecidable problems.

Does a program print “Hello World”?
Find exit points and add statement: **Print** “Hello World.”

Can a set of notched tiles tile the infinite plane?

Undecidable problems.

Does a program print “Hello World”?
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 print “Hello World”?
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 print "Hello World"?
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 print “Hello World”?
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: Ask program if “ $x^n + y^n = 1$?” has integer solutions.

Undecidable problems.

Does a program print “Hello World”?
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: Ask program if “ $x^n + y^n = 1$?” has integer solutions.
Problem is undecidable.

Undecidable problems.

Does a program print “Hello World”?
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: Ask program if “ $x^n + y^n = 1$?” has integer solutions.
Problem is undecidable.

Be careful!

Undecidable problems.

Does a program print “Hello World”?
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: Ask program if “ $x^n + y^n = 1$?” has integer solutions.
Problem is undecidable.

Be careful!

Undecidable problems.

Does a program print “Hello World”?
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: Ask program if “ $x^n + y^n = 1$?” has integer solutions.
Problem is undecidable.

Be careful!

Is there a solution to $x^n + y^n = 1$?
(Diophantine equation.)

Undecidable problems.

Does a program print “Hello World”?
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: Ask program if “ $x^n + y^n = 1$?” has integer solutions.
Problem is undecidable.

Be careful!

Is there a 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 print “Hello World”?
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: Ask program if “ $x^n + y^n = 1$?” has integer solutions.
Problem is undecidable.

Be careful!

Is there a 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 print "Hello World"?
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: Ask program if " $x^n + y^n = 1$?" has integer solutions.
Problem is undecidable.

Be careful!

Is there a 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

Undecidable problems.

Does a program print “Hello World”?
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: Ask program if “ $x^n + y^n = 1$?” has integer solutions.
Problem is undecidable.

Be careful!

Is there a 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 output correct answer.

Simulate

1) Any program with finite time/space.

Simulate

1) Any program with finite time/space.

2) Can output all programs that halt on themselves?

Simulate

1) Any program with finite time/space.

2) Can output all programs that halt on themselves?

Why?

Simulate

1) Any program with finite time/space.

2) Can output all programs that halt on themselves?

Why?

1) Run it and check!

Simulate

1) Any program with finite time/space.

2) Can output all programs that halt on themselves?

Why?

1) Run it and check!

2) Like enumerating pairs of natural numbers.

Simulate

1) Any program with finite time/space.

2) Can output all programs that halt on themselves?

Why?

1) Run it and check!

2) Like enumerating pairs of natural numbers.
– (program, time).

Simulate

1) Any program with finite time/space.

2) Can output all programs that halt on themselves?

Why?

1) Run it and check!

2) Like enumerating pairs of natural numbers.
– (program, time). and run program for that time.

Simulate

1) Any program with finite time/space.

2) Can output all programs that halt on themselves?

Why?

1) Run it and check!

2) Like enumerating pairs of natural numbers.
– (program, time). and run program for that time.
Each program that halts, halts at some time.

Final format

Time: approximately 180 minutes

Final format

Time: approximately 180 minutes

Many short answers.

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well:

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast,

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium:

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower,

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower, less correct.

Final format

Time: approximately 180 minutes

Many short answers.
  Get at ideas that we study.
   Know material well: fast, correct.
   Know material medium: slower, less correct.
   Know material not so well:

Final format

Time: approximately 180 minutes

Many short answers.
  Get at ideas that we study.
   Know material well: fast, correct.
   Know material medium: slower, less correct.
   Know material not so well: Uh oh.

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.

Priming: sequence of questions...

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower, less correct.
Know material not so well: Uh oh.

Some longer questions.

Priming: sequence of questions...
but don't overdo this as test strategy!!!

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower, less correct.
Know material not so well: Uh oh.

Some longer questions.

Priming: sequence of questions...
but don't overdo this as test strategy!!!

Proofs,

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.

Priming: sequence of questions...
but don't overdo this as test strategy!!!

Proofs, algorithms,

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower, less correct.
Know material not so well: Uh oh.

Some longer questions.

Priming: sequence of questions...
but don’t overdo this as test strategy!!!

Proofs, algorithms, properties.

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower, less correct.
Know material not so well: Uh oh.

Some longer questions.

Priming: sequence of questions...
but don't overdo this as test strategy!!!

Proofs, algorithms, properties.
Some calculation.

Final format

Time: approximately 180 minutes

Many short answers.
Get at ideas that we study.
Know material well: fast, correct.
Know material medium: slower, less correct.
Know material not so well: Uh oh.

Some longer questions.

Priming: sequence of questions...
but don't overdo this as test strategy!!!

Proofs, algorithms, properties.
Some calculation.

Wrapup.

Wrapup.

Watch Piazza for Logistics!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
Other arrangements.

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
- Other arrangements.
- Should have recieved an email today from me.

Other issues....
- satishr@cs.berkeley.edu
- Private message on piazza.

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
   Other arrangements.
   Should have recieved an email today from me.

Other issues....
   satishr@cs.berkeley.edu
   Private message on piazza.

Good Studying!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
   Other arrangements.
   Should have recieved an email today from me.

Other issues....
   satishr@cs.berkeley.edu
   Private message on piazza.

Good Studying!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
- Other arrangements.
- Should have recieved an email today from me.

Other issues....
- satishr@cs.berkeley.edu
- Private message on piazza.

Good Studying!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
   Other arrangements.
   Should have recieved an email today from me.

Other issues....
   satishr@cs.berkeley.edu
   Private message on piazza.

Good Studying!!!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
- Other arrangements.
- Should have recieved an email today from me.

Other issues....
- satishr@cs.berkeley.edu
- Private message on piazza.

Good Studying!!!!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
Other arrangements.
Should have recieved an email today from me.

Other issues....
satishr@cs.berkeley.edu
Private message on piazza.

Good Studying!!!!!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!!!!!!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
Other arrangements.
Should have recieved an email today from me.

Other issues....
satishr@cs.berkeley.edu
Private message on piazza.

Good Studying!!!!!!!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
- Other arrangements.
- Should have recieved an email today from me.

Other issues....
- satishr@cs.berkeley.edu
- Private message on piazza.

Good Studying!!!!!!!!!!!!!!!

Wrapup.

Watch Piazza for Logistics!
Watch Piazza for Advice!

If you sent me email about Final conflicts
  Other arrangements.
  Should have recieved an email today from me.

Other issues....
  satishr@cs.berkeley.edu
  Private message on piazza.

Good Studying!!!!!!!!!!!!!!!!!