Midterm 2 Review.

Midterm 2 Review.

Midterm 2 Review.

Midterm Topics: Notes 6-14.

Modular Arithmetic. Inverses. GCD/Extended-GCD.

RSA/Cryptography.

Polynomials.
  Secret Sharing.
  Erasure Resistant Encoding.
  Error Correction.

Counting.

Countability.

Computability.

Probability Topics covered by Prof. Walrand.

Midterm format

Time: 120 minutes

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
mix of short and longer questions

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
  mix of short and longer questions

Prep/Exam Strategy:
  plan out sequence of questions...

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
  mix of short and longer questions

Prep/Exam Strategy:
  plan out sequence of questions...
  solve problems with a time bound

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
mix of short and longer questions

Prep/Exam Strategy:
plan out sequence of questions...
solve problems with a time bound

Proofs,

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
mix of short and longer questions

Prep/Exam Strategy:
plan out sequence of questions...
solve problems with a time bound

Proofs, algorithms,

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
    mix of short and longer questions

Prep/Exam Strategy:
    plan out sequence of questions...
    solve problems with a time bound

Proofs, algorithms, properties.

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
  mix of short and longer questions

Prep/Exam Strategy:
  plan out sequence of questions...
  solve problems with a time bound

Proofs, algorithms, properties.
 Some mild calculation (no calculators needed though!).

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
  mix of short and longer questions

Prep/Exam Strategy:
  plan out sequence of questions...
  solve problems with a time bound

Proofs, algorithms, properties.
  Some mild calculation (no calculators needed though!).

Midterm format

Time: 120 minutes

Will broadly follow Midterm1 format:
  mix of short and longer questions

Prep/Exam Strategy:
  plan out sequence of questions...
  solve problems with a time bound

Proofs, algorithms, properties.
 Some mild calculation (no calculators needed though!).

Be familiar with Midterm1 topics... but MT2 will focus on Notes 6-14.

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 $gcd(x,m) = 1$.

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 $gcd(x,m) = 1$.

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 $gcd(x, m) = 1$.

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 $gcd(x,m) = 1$.

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 $gcd(x,m) = 1$.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

Extended-gcd(x,y)

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x, m) = 1$.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x, m) = 1$.

Finding gcd.
$gcd(x, y) = gcd(y, x - y) = gcd(y, x \pmod{y})$.

Extended-gcd(x, y) returns (d, a, b)

Modular Arithmetic Inverses and GCD

x has inverse modulo m if and only if $gcd(x, m) = 1$.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x, m) = 1$.

Finding gcd.
$gcd(x, y) = gcd(y, x - y) = gcd(y, x \pmod y)$.

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 $gcd(x,m) = 1$.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod{y})$.

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 $gcd(x,m) = 1$.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

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 $gcd(x,m) = 1$.

Proof Idea:
$\{0x,\ldots,(m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y,x-y) = gcd(y,x \pmod{y})$.

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 $gcd(x,m) = 1$.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

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 $gcd(x,m) = 1$.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

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 $gcd(x,m) = 1$.

Proof Idea:
$\{0x, \ldots, (m-1)x\}$ are distinct modulo m if and only if $gcd(x,m) = 1$.

Finding gcd.
$gcd(x,y) = gcd(y, x-y) = gcd(y, x \pmod y)$.

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! $1 = ax + bm = ax \pmod m$.

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

RSA:

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

RSA:

$N = p, q$

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

RSA:

$N = p, q$

e with $\gcd(e, (p-1)(q-1)) = 1$.

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

RSA:

$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

RSA:

$N = p, q$
e with $\gcd(e, (p-1)(q-1)) = 1$.
$d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: $x^{ed} = x \pmod{N}$

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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:

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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!

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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!

$x^{ed} - x$

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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!

$x^{ed} - x = x^{k(p-1)(q-1)+1} - x$

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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!

$$x^{ed} - x = x^{k(p-1)(q-1)+1} - x = x((x^{k(q-1)})^{p-1} - 1)$$

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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.

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod p$,

$$a^{p-1} \equiv 1 \pmod p.$$

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.

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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.

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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.

Fermat/RSA

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

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. $\square$

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. $\square$

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:

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:

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.

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. $\square$

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. $\square$

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.

Proof Ideas:
Lagrange Interpolation gives existence.
Property 1 gives uniqueness. $\square$

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 $k-1$ 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 $k-1$ 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 $k-1$ 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 $k-1$ 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 $k-1$ 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, $P(x)$.

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 $k-1$ 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, $P(x)$.
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 $k-1$ 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, $P(x)$.
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 $k-1$ 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, $P(x)$.
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 sufficient 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 $k-1$ 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, $P(x)$.
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 sufficient 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 $k-1$ 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, $P(x)$.
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 sufficient 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 $k-1$ 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, $P(x)$.
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 sufficient 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 $k-1$ 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, $P(x)$.
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 sufficient 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 $k-1$ 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, $P(x)$.
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 sufficient 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 $k-1$ 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, $P(x)$.
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 sufficient 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 $k-1$ 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, $P(x)$.
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 sufficient 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.

Berlekamp-Welch

Idea: Error locator polynomial of degree k with zeros at errors.

Berlekamp-Welch

Idea: Error locator polynomial of degree k with zeros at errors.

Berlekamp-Welch

Idea: Error locator polynomial of degree k with zeros at errors.

For all points $i = 1, \ldots, i, n + 2k$, $P(i)E(i) = R(i)E(i) \pmod{p}$

Berlekamp-Welch

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.

Berlekamp-Welch

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 $Q(x) = P(x)E(x)$.

Berlekamp-Welch

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 $Q(x) = P(x)E(x)$.

$$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0.$$

Berlekamp-Welch

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 $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.

Berlekamp-Welch

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 $Q(x) = P(x)E(x)$.

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots a_0$.
$E(x) = x^k + b_{k-1}x^{k-1} + \cdots b_0$.

Berlekamp-Welch

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.

Berlekamp-Welch

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.

$$a_{n+k-1} + \ldots a_0 \equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p}$$

Berlekamp-Welch

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.

$$\begin{aligned} a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\ a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\ &\vdots \end{aligned}$$

Berlekamp-Welch

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.

$$
\begin{aligned}
a_{n+k-1} + \ldots a_0 &\equiv R(1)(1 + b_{k-1} \cdots b_0) \pmod{p} \\
a_{n+k-1}(2)^{n+k-1} + \ldots a_0 &\equiv R(2)((2)^k + b_{k-1}(2)^{k-1} \cdots b_0) \pmod{p} \\
&\vdots \\
a_{n+k-1}(m)^{n+k-1} + \ldots a_0 &\equiv R(m)((m)^k + b_{k-1}(m)^{k-1} \cdots b_0) \pmod{p}
\end{aligned}
$$

Berlekamp-Welch

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 $E(x)$!

Berlekamp-Welch

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 $E(x)$.

Berlekamp-Welch

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 $P(x) = Q(x)/E(x)$.

Berlekamp-Welch

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 $P(x) = Q(x)/E(x)$.

Berlekamp-Welch

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 $P(x) = Q(x)/E(x)$.

Berlekamp-Welch

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 $P(x) = Q(x)/E(x)$.

Countability

Isomorphism principle.

Countability

Isomorphism principle.
Countable and Uncountable.

Countability

Isomorphism principle.
Countable and Uncountable.
Enumeration

Countability

Isomorphism principle.
Countable and Uncountable.
Enumeration
Diagonalization.

Isomorphism principle.

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

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

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?

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

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

Cardinalities of uncountable sets?

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

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

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

One to one.

Cardinalities of uncountable sets?

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

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

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

One to one. $x \neq y$

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?

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

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

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

Cardinalities of uncountable sets?

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

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

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

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

Cardinalities of uncountable sets?

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

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

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

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

Cardinalities of uncountable sets?

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

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

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

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

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 $\implies f(x) \neq f(y)$.

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,

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?

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

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

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 $\implies f(x) \neq f(y)$.

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?

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

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

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 $\implies f(x) \neq f(y)$.
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)

- E even numbers.

Examples

Countably infinite (same cardinality as naturals)

- E even numbers.
  Where are the odds?

Examples

Countably infinite (same cardinality as naturals)

- E even numbers.
  Where are the odds? Half as big?

Examples

Countably infinite (same cardinality as naturals)

- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.

Examples

Countably infinite (same cardinality as naturals)

- 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)

- 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)

- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Enumerate: $0,$

Examples

Countably infinite (same cardinality as naturals)

- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Enumerate: $0, -1,$

Examples

Countably infinite (same cardinality as naturals)

- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Enumerate: $0, -1, 1,$

Examples

Countably infinite (same cardinality as naturals)

- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  Enumerate: $0, -1, 1, -2,$

Examples

Countably infinite (same cardinality as naturals)

- E even numbers.
  Where are the odds? Half as big?
  Bijection: $f(e) = e/2$.
- Z- all integers.
  Twice as big?
  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.
  Enumerate: $(0,0), (0,1), (0,2), \ldots$

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  Enumerate: $(0,0),(0,1),(0,2),\ldots$???
  Never get to $(1,1)$!

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  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.
  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.
  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.
  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.
  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.
  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 $(a+b+1)(a+b+2)/2$ in this order.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  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 $(a+b+1)(a+b+2)/2$ in this order.
- Positive Rational numbers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  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 $(a+b+1)(a+b+2)/2$ in this order.
- Positive Rational numbers.
  Infinite Subset of pairs of natural numbers.

Examples: Countable by enumeration

- $N \times N$ - Pairs of integers.
  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 $(a+b+1)(a+b+2)/2$ 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.
  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 $(a+b+1)(a+b+2)/2$ 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.
  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 $(a+b+1)(a+b+2)/2$ 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.
  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 $(a+b+1)(a+b+2)/2$ 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.
  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 $(a+b+1)(a+b+2)/2$ 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.
  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 $(a+b+1)(a+b+2)/2$ 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.
  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 $(a+b+1)(a+b+2)/2$ 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.
  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 $(a+b+1)(a+b+2)/2$ 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 undecidable (not solvable by computer).

Uncomputability.

Halting problem is undecidable (not solvable by computer).

Diagonalization.

Uncomputability.

Halting problem is undecidable (not solvable by computer).

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 and Turing.

Theorem: There is no program HALT.
Proof:

Halt and Turing.

Theorem: There is no program HALT.

Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Halt and Turing.

Theorem: There is no program HALT.
Proof: Assume there is a program $HALT(\cdot, \cdot)$.

Turing(P)

Halt and Turing.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
Proof: Assume there is a program $\mathit{HALT}(\cdot, \cdot)$.

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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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.

Theorem: There is no program HALT.
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 of arbitrary program to test for halting
and add statement: **Print** “Hello World.”

Undecidable problems.

Does a program print "Hello World"?
Find exit points of arbitrary program to test for halting
and add statement: **Print** "Hello World."

Undecidable problems.

Does a program print “Hello World”?
Find exit points of arbitrary program to test for halting
and add statement: **Print** “Hello World.”

Does a program halt in 1000 steps?

Undecidable problems.

Does a program print “Hello World”?
Find exit points of arbitrary program to test for halting
and add statement: **Print** “Hello World.”

Does a program halt in 1000 steps?
Decidable! Just run it for 1000 steps and see if it terminates.

Undecidable problems.

Does a program print “Hello World”?
Find exit points of arbitrary program to test for halting
  and add statement: **Print** “Hello World.”

Does a program halt in 1000 steps?
  Decidable! Just run it for 1000 steps and see if it terminates.

Be careful!

Undecidable problems.

Does a program print "Hello World"?
Find exit points of arbitrary program to test for halting
  and add statement: **Print** "Hello World."

Does a program halt in 1000 steps?
  Decidable! Just run it for 1000 steps and see if it terminates.

Be careful!

Undecidable problems.

Does a program print "Hello World"?
Find exit points of arbitrary program to test for halting
and add statement: **Print** "Hello World."

Does a program halt in 1000 steps?
Decidable! Just run it for 1000 steps and see if it terminates.

Be careful!

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.

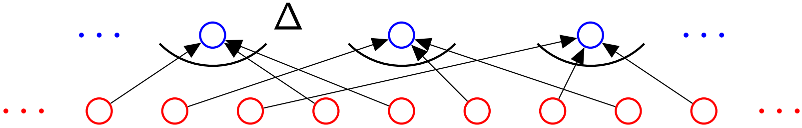
Counting

First Rule
Second Rule
Stars/Bars
Common Scenarios: Sampling, Balls in Bins.
Sum Rule. Inclusion/Exclusion.
Combinatorial Proofs.

Example: visualize.

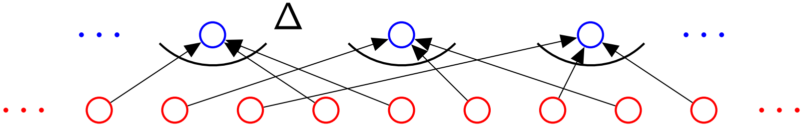
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**

Second rule: when order doesn't matter divide..when possible.



Example: visualize.

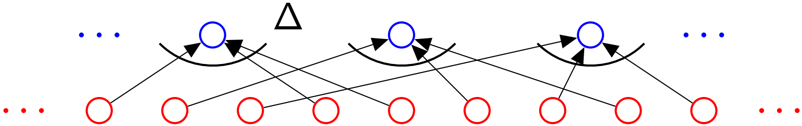
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: 52

Example: visualize.

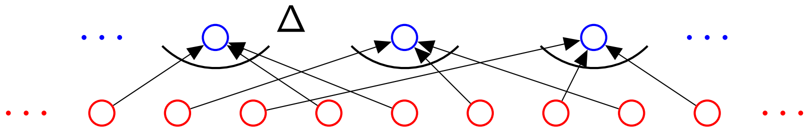
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: 52×51

Example: visualize.

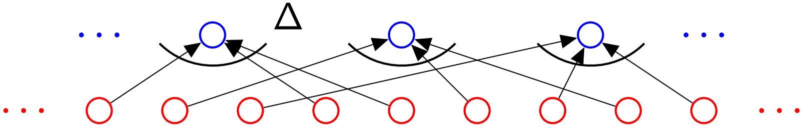
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50$

Example: visualize.

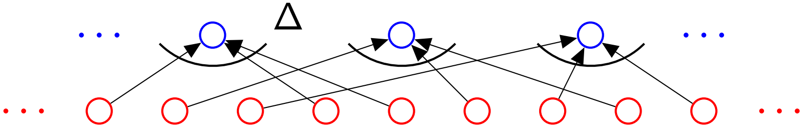
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$.

Example: visualize.

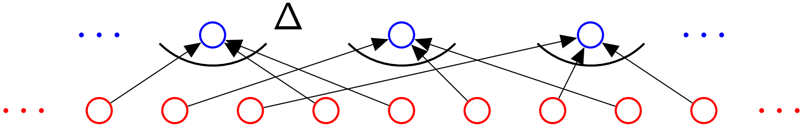
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.

Example: visualize.

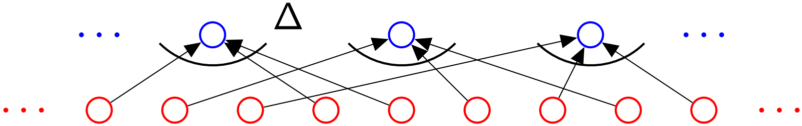
First rule: $n_1 \times n_2 \cdots \times n_3$. Product Rule.
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?

Example: visualize.

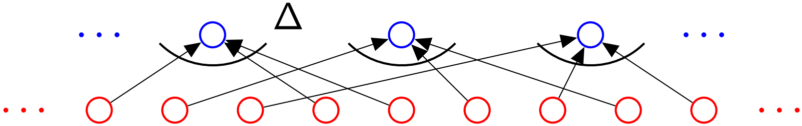
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.

Example: visualize.

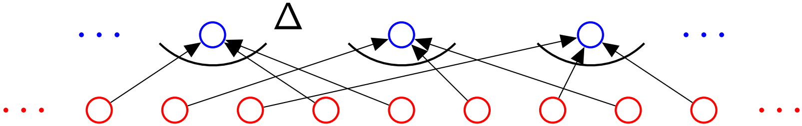
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



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.

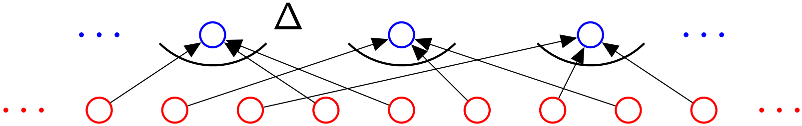
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



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.

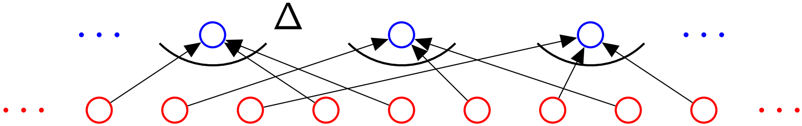
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



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.

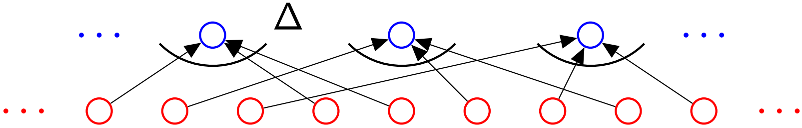
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



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?

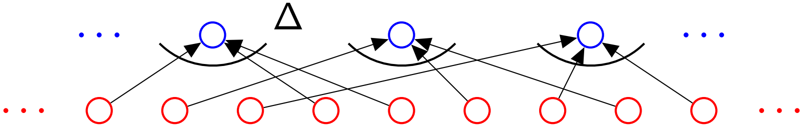
Hand: Q, K, A.

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.

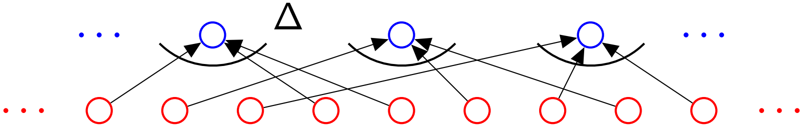
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



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.

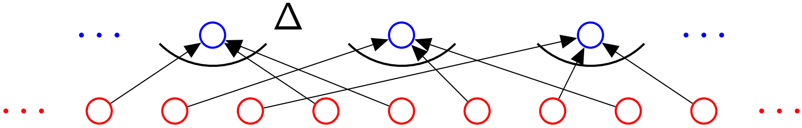
First rule: $n_1 \times n_2 \cdots \times n_3$. **Product Rule.**
Second rule: when order doesn't matter divide..when possible.



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.

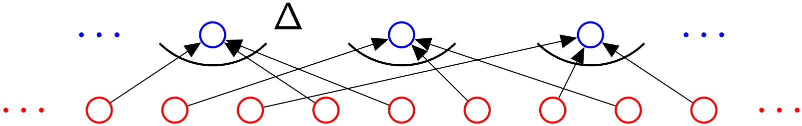
First rule: $n_1 \times n_2 \cdots \times n_3$**. Product Rule.**
Second rule: when order doesn't matter divide..when possible.



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.
Δ = $3 \times 2 \times 1$ 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.

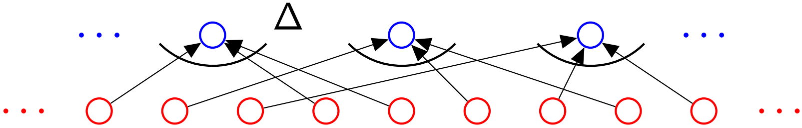
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.

$\Delta = 3 \times 2 \times 1$ 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.

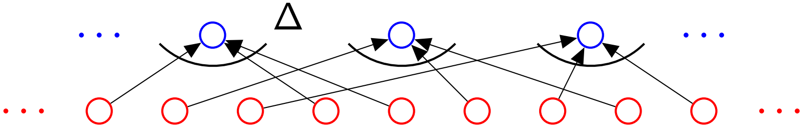


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

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!

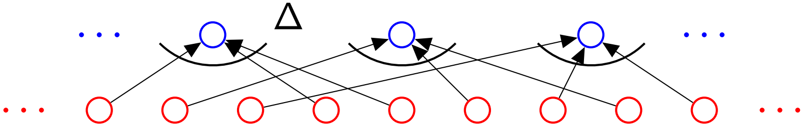
Choose k out of n.

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.

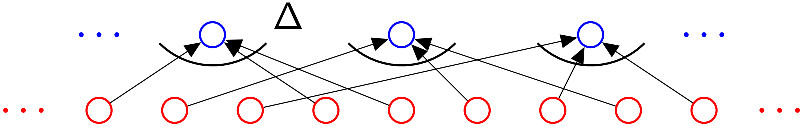


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
Hand: Q, K, A.
Deals: Q, K, A, Q, A, K, K, A, Q, K, A, Q, A, K, Q, A, Q, K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

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.

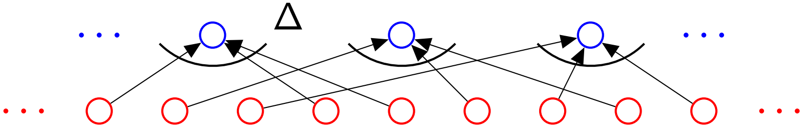


3 card Poker deals: $52 \times 51 \times 50 = \frac{52!}{49!}$. First rule.
Poker hands: Δ?
  Hand: Q, K, A.
  Deals: Q,K,A, Q,A,K, K,A,Q,K,A,Q, A,K,Q, A,Q,K.
$\Delta = 3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

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.

Δ = $3 \times 2 \times 1$ First rule again.
Total: $\frac{52!}{49!3!}$ Second Rule!

Choose k out of n.

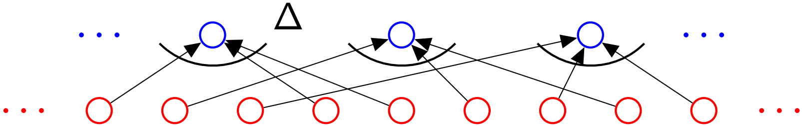
Ordered set: $\frac{n!}{(n-k)!}$

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.

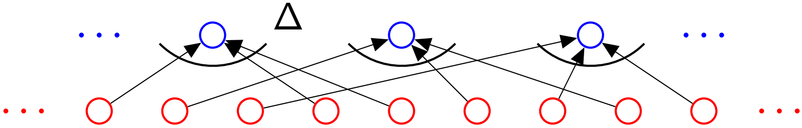
Ordered set: $\frac{n!}{(n-k)!}$

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.

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.

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: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$.

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, $\binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}$. $\square$

Wrapup.

Wrapup.

Watch Piazza for Logistics!

Wrapup.

Watch Piazza for Logistics!

Watch Piazza for Advice!

Wrapup.

Watch Piazza for Logistics!

Watch Piazza for Advice!

Note your Midterm2 room assignments!!!

Wrapup.

Watch Piazza for Logistics!

Watch Piazza for Advice!

Note your Midterm2 room assignments!!!

Other issues....

Wrapup.

Watch Piazza for Logistics!

Watch Piazza for Advice!

Note your Midterm2 room assignments!!!

Other issues....
 Email logistics@eecs70.org

Wrapup.

Watch Piazza for Logistics!

Watch Piazza for Advice!

Note your Midterm2 room assignments!!!

Other issues....
- Email logistics@eecs70.org
- Private message on piazza.

Wrapup.

Watch Piazza for Logistics!

Watch Piazza for Advice!

Note your Midterm2 room assignments!!!

Other issues....

Email logistics@eecs70.org

Private message on piazza.

Wrapup.

Watch Piazza for Logistics!

Watch Piazza for Advice!

Note your Midterm2 room assignments!!!

Other issues....
  Email logistics@eecs70.org
  Private message on piazza.

Good Studying and Good Luck!!!