Today

Finish RSA

Today

Finish RSA

Signatures.

Today

Finish RSA

Signatures.

Warnings.

Today

Finish RSA

Signatures.

Warnings.

Midterm Review

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\qquad u = 1 \pmod{m}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad v = 0 \pmod{m}$

Let $x = au + bv$.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\qquad u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\qquad v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution?

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\qquad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\qquad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.
$\implies (x - y)$ is multiple of m and n since gcd(m,n)=1.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.
$\implies$ $(x - y)$ is multiple of m and n since gcd(m,n)=1.
$\implies$ $x - y \geq mn$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\qquad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\qquad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x-y) \equiv 0 \pmod{m}$ and $(x-y) \equiv 0 \pmod{n}$.
$\implies (x-y)$ is multiple of m and n since gcd(m,n)=1.
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn-1\}$.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\qquad u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\qquad v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.
$\implies (x - y)$ is multiple of m and n since gcd(m,n)=1.
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn-1\}$.
Thus, only one solution modulo mn.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.
$\implies (x - y)$ is multiple of m and n since gcd(m,n)=1.
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn - 1\}$.
Thus, only one solution modulo mn. $\square$

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.
$\implies (x - y)$ is multiple of m and n since gcd(m,n)=1.
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn - 1\}$.
Thus, only one solution modulo mn. $\square$
My love is won.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m, n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.
$\implies (x - y)$ is multiple of m and n since gcd(m, n)=1.
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn - 1\}$.
Thus, only one solution modulo mn. $\square$
My love is won. Zero and One.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.
$\implies (x - y)$ is multiple of m and n since gcd(m,n)=1.
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn - 1\}$.
Thus, only one solution modulo mn. $\square$
My love is won. Zero and One. Nothing and nothing done.

Simple Chinese Remainder Theorem.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where gcd(m,n)=1.

CRT Thm: There is a unique solution $x \pmod{mn}$.
Proof:
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\qquad u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\qquad v = 0 \pmod{m}$

Let $x = au + bv$.
$x = au + bv = a \pmod{m}$: $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = au + bv = b \pmod{n}$: $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Only solution? If not, two solutions, x and y.
$(x - y) \equiv 0 \pmod{m}$ and $(x - y) \equiv 0 \pmod{n}$.
$\implies (x - y)$ is multiple of m and n since gcd(m,n)=1.
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn - 1\}$.
Thus, only one solution modulo mn. $\square$
My love is won. Zero and One. Nothing and nothing done.

Fermat's Theorem: Reducing Exponents.

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

Fermat's Theorem: Reducing Exponents.

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

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

Fermat's Theorem: Reducing Exponents.

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

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

Proof:

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p,

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p, solve to get...

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p, solve to get...

$$a^{(p-1)} \equiv 1 \mod p.$$

Fermat's Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p, solve to get...

$$a^{(p-1)} \equiv 1 \mod p.$$

□

Decoding.

$$E(m,(N,e)) = m^e \pmod{N}.$$

Decoding.

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

Decoding.

$E(m,(N,e)) = m^e \pmod N$.
$D(m,(N,d)) = m^d \pmod N$.

Decoding.

$$E(m,(N,e)) = m^e \pmod{N}.$$
$$D(m,(N,d)) = m^d \pmod{N}.$$

$N = pq$

Decoding.

$$E(m,(N,e)) = m^e \pmod{N}.$$
$$D(m,(N,d)) = m^d \pmod{N}.$$

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Decoding.

$$E(m,(N,e)) = m^e \pmod{N}.$$
$$D(m,(N,d)) = m^d \pmod{N}.$$

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want:

Decoding.

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Always decode correctly?

$$E(m,(N,e)) = m^e \pmod{N}.$$

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

Always decode correctly?

$$E(m,(N,e)) = m^e \pmod{N}.$$
$$D(m,(N,d)) = m^d \pmod{N}.$$

$N = pq$

Always decode correctly?

$$E(m,(N,e)) = m^e \pmod N.$$
$$D(m,(N,d)) = m^d \pmod N.$$

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want:

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Always decode correctly?

$E(m,(N,e)) = m^e \pmod N$.
$D(m,(N,d)) = m^d \pmod N$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod N$.

Another view:

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Always decode correctly?

$E(m,(N,e)) = m^e \pmod N$.
$D(m,(N,d)) = m^d \pmod N$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod N$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

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

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

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

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

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

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

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

$$\implies a^{k(p-1)} \equiv 1 \pmod{p}$$

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

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

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

$$\implies a^{k(p-1)} \equiv 1 \pmod{p} \implies$$

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

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

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

$$\implies a^{k(p-1)} \equiv 1 \pmod{p} \implies a^{k(p-1)+1}$$

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

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

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

$$\implies a^{k(p-1)} \equiv 1 \pmod{p} \implies a^{k(p-1)+1} = a \pmod{p}$$

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

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

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

$$\implies a^{k(p-1)} \equiv 1 \pmod{p} \implies a^{k(p-1)+1} = a \pmod{p}$$

versus $\quad a^{k(p-1)(q-1)+1} = a \pmod{pq}$.

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

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

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

$$\implies a^{k(p-1)} \equiv 1 \pmod{p} \implies a^{k(p-1)+1} = a \pmod{p}$$

versus $a^{k(p-1)(q-1)+1} = a \pmod{pq}$.

Similar, not same, but useful.

...decoding correctness...

CRT: Isomorphism between $(a \mod p, b \mod q)$ and $x \pmod{pq}$

...decoding correctness...

CRT: Isomorphism between $(a \mod p, b \mod q)$ and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

...decoding correctness...

CRT: Isomorphism between ($a \mod p, b \mod q$) and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

...decoding correctness...

CRT: Isomorphism between ($a \mod p, b \mod q$) and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

...decoding correctness...

CRT: Isomorphism between ($a \mod p, b \mod q$) and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

...decoding correctness...

CRT: Isomorphism between $(a \mod p, b \mod q)$ and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

...decoding correctness...

CRT: Isomorphism between ($a \mod p, b \mod q$) and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

$a^{1+k(p-1)(q-1)} = a(a^{(p-1)})^{k(q-1)} = a \pmod{p}$

...decoding correctness...

CRT: Isomorphism between ($a \mod p, b \mod q$) and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

$a^{1+k(p-1)(q-1)} = a(a^{(p-1)})^{k(q-1)} = a \pmod{p}$

By Fermat. $a^{p-1} = 1 \pmod{p}$

...decoding correctness...

CRT: Isomorphism between ($a \mod p, b \mod q$) and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

$a^{1+k(p-1)(q-1)} = a(a^{(p-1)})^{k(q-1)} = a \pmod{p}$

By Fermat. $a^{p-1} = 1 \pmod{p}$ or $a = 0$.

...decoding correctness...

CRT: Isomorphism between ($a \mod p, b \mod q$) and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

$a^{1+k(p-1)(q-1)} = a(a^{(p-1)})^{k(q-1)} = a \pmod{p}$

By Fermat. $a^{p-1} = 1 \pmod{p}$ or $a = 0$.

$b^{1+k(p-1)(q-1)} = b(b^{(q-1)})^{k(p-1)} = b \pmod{q}$

...decoding correctness...

CRT: Isomorphism between $(a \mod p, b \mod q)$ and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

$a^{1+k(p-1)(q-1)} = a(a^{(p-1)})^{k(q-1)} = a \pmod{p}$

By Fermat. $a^{p-1} = 1 \pmod{p}$ or $a = 0$.

$b^{1+k(p-1)(q-1)} = b(b^{(q-1)})^{k(p-1)} = b \pmod{q}$

By Fermat. $b^{q-1} = 1 \pmod{q}$

...decoding correctness...

CRT: Isomorphism between $(a \mod p, b \mod q)$ and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

$a^{1+k(p-1)(q-1)} = a(a^{(p-1)})^{k(q-1)} = a \pmod{p}$

By Fermat. $a^{p-1} = 1 \pmod{p}$ or $a = 0$.

$b^{1+k(p-1)(q-1)} = b(b^{(q-1)})^{k(p-1)} = b \pmod{q}$

By Fermat. $b^{q-1} = 1 \pmod{q}$ or $b = 0$.

...decoding correctness...

CRT: Isomorphism between $(a \mod p, b \mod q)$ and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

$a^{1+k(p-1)(q-1)} = a(a^{(p-1)})^{k(q-1)} = a \pmod{p}$

By Fermat. $a^{p-1} = 1 \pmod{p}$ or $a = 0$.

$b^{1+k(p-1)(q-1)} = b(b^{(q-1)})^{k(p-1)} = b \pmod{q}$

By Fermat. $b^{q-1} = 1 \pmod{q}$ or $b = 0$.

$x^{ed} = a \pmod{p}$ and $x^{ed} = b \pmod{q}$.

...decoding correctness...

CRT: Isomorphism between ($a \mod p, b \mod q$) and $x \pmod{pq}$

$e = d^{-1} \mod pq$.

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

Now $x = a \pmod{p}$ and $x = b \pmod{q}$.

$a^{1+k(p-1)(q-1)} = a(a^{(p-1)})^{k(q-1)} = a \pmod{p}$

By Fermat. $a^{p-1} = 1 \pmod{p}$ or $a = 0$.

$b^{1+k(p-1)(q-1)} = b(b^{(q-1)})^{k(p-1)} = b \pmod{q}$

By Fermat. $b^{q-1} = 1 \pmod{q}$ or $b = 0$.

$x^{ed} = a \pmod{p}$ and $x^{ed} = b \pmod{q}$.

CRT $\implies x^{ed} = x \pmod{pq}$.

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

$$\pi(N) \geq N/\ln N.$$

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime?

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test..

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test.. Primes in P).

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test.. Primes in P).

   For 1024 bit number, 1 in 710 is prime.

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test.. Primes in P).

   For 1024 bit number, 1 in 710 is prime.

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

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test.. Primes in P).

   For 1024 bit number, 1 in 710 is prime.

2. Choose e with $\gcd(e, (p-1)(q-1)) = 1$.
   Use gcd algorithm to test.

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test.. Primes in P).

   For 1024 bit number, 1 in 710 is prime.

2. Choose e with $\gcd(e,(p-1)(q-1)) = 1$.
   Use gcd algorithm to test.

3. Find inverse d of e modulo $(p-1)(q-1)$.

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test.. Primes in P).

   For 1024 bit number, 1 in 710 is prime.

2. Choose e with $\gcd(e,(p-1)(q-1)) = 1$.
   Use gcd algorithm to test.

3. Find inverse d of e modulo $(p-1)(q-1)$.
   Use extended gcd algorithm.

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N / \ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test.. Primes in P).

   For 1024 bit number, 1 in 710 is prime.

2. Choose e with $\gcd(e,(p-1)(q-1)) = 1$.
   Use gcd algorithm to test.

3. Find inverse d of e modulo $(p-1)(q-1)$.
   Use extended gcd algorithm.

All steps are polynomial in $O(\log N)$, the number of bits.

Security of RSA.

Security of RSA.

Security?

Security of RSA.

Security?

1. Alice knows p and q.

Security of RSA.

Security?

1. Alice knows p and q.
2. Bob only knows, $N(= pq)$, and e.

Security of RSA.

Security?

1. Alice knows p and q.
2. Bob only knows, $N(= pq)$, and e.

   Does not know, for example, d or factorization of N.

Security of RSA.

Security?

1. Alice knows p and q.
2. Bob only knows, $N(= pq)$, and e.
   Does not know, for example, d or factorization of N.
3. I don't know how to break this scheme without factoring N.

Security of RSA.

Security?

1. Alice knows p and q.
2. Bob only knows, $N(= pq)$, and e.
   Does not know, for example, d or factorization of N.
3. I don't know how to break this scheme without factoring N.

No one I know or have heard of admits to knowing how to factor N.

Security of RSA.

Security?

1. Alice knows p and q.
2. Bob only knows, $N(= pq)$, and e.
   Does not know, for example, d or factorization of N.
3. I don't know how to break this scheme without factoring N.

No one I know or have heard of admits to knowing how to factor N.

Breaking in general sense $\implies$ factoring algorithm.

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

The protocols are built on RSA but more complicated;

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

The protocols are built on RSA but more complicated;

For example, several rounds of challenge/response.

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

The protocols are built on RSA but more complicated;

For example, several rounds of challenge/response.

One trick:
  Bob encodes credit card number, c,

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

The protocols are built on RSA but more complicated;

For example, several rounds of challenge/response.

One trick:
Bob encodes credit card number, c,
concatenated with random k-bit number r.

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

The protocols are built on RSA but more complicated;

For example, several rounds of challenge/response.

One trick:
Bob encodes credit card number, c,
concatenated with random k-bit number r.

Never sends just c.

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

The protocols are built on RSA but more complicated;

For example, several rounds of challenge/response.

One trick:
Bob encodes credit card number, c,
concatenated with random k-bit number r.

Never sends just c.

Again, more work to do to get entire system.

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

The protocols are built on RSA but more complicated;

For example, several rounds of challenge/response.

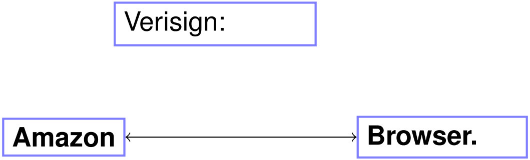
One trick:
  Bob encodes credit card number, c,
    concatenated with random k-bit number r.

Never sends just c.

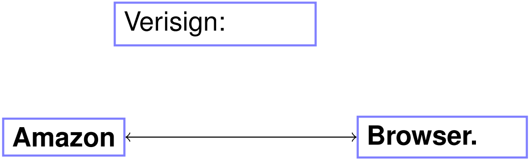
Again, more work to do to get entire system.

CS161...

Signatures using RSA.

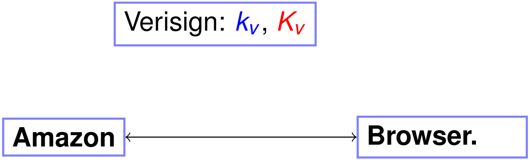


Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

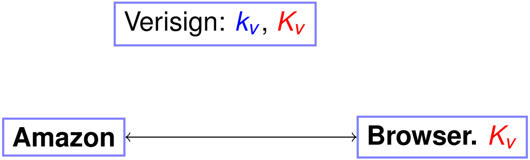
Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Signatures using RSA.

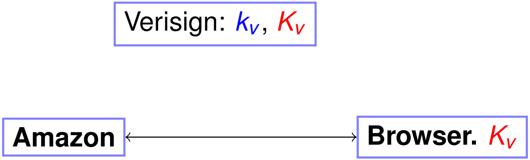


Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Browser "knows" Verisign's public key: K_V.

Signatures using RSA.



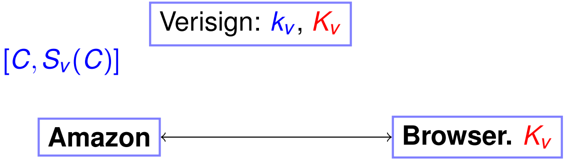
Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

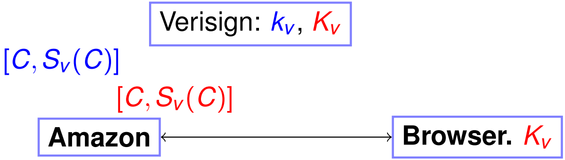
Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

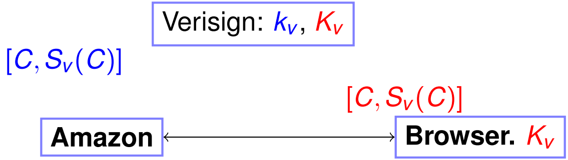
Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

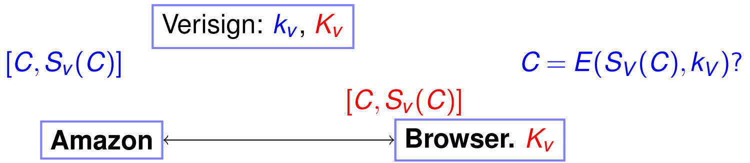
Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Browser "knows" Verisign's public key: K_V.

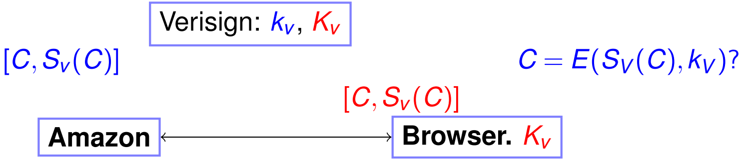
Amazon Certificate: C = "I am Amazon. My public Key is K_A."

Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

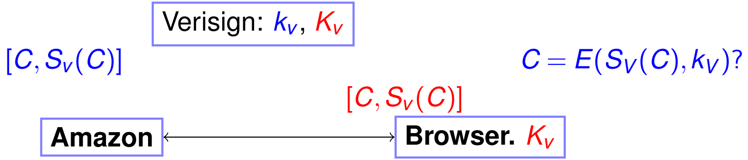
Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

$E(S_v(C), K_V)$

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

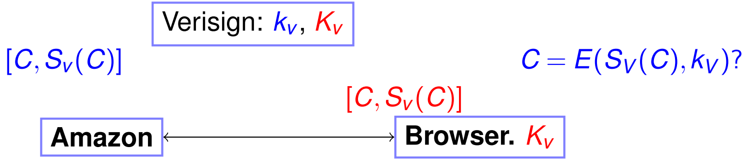
Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

$E(S_v(C), K_V) = (S_v(C))^e$

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ ($N = pq$.)

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

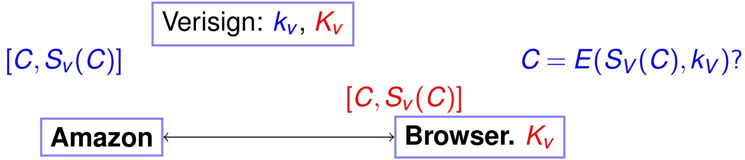
Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

$$E(S_v(C), K_V) = (S_v(C))^e = (C^d)^e$$

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ ($N = pq$.)

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

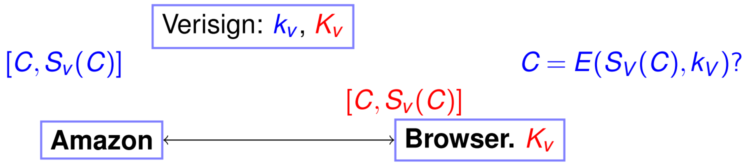
Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

$E(S_v(C), K_V) = (S_v(C))^e = (C^d)^e = C^{de}$

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ ($N = pq$.)

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

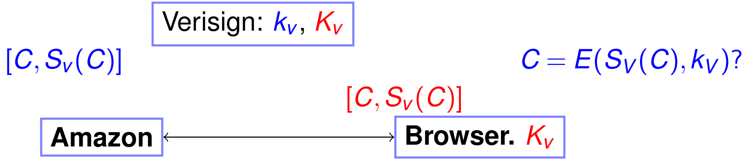
Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

$$E(S_v(C), K_V) = (S_v(C))^e = (C^d)^e = C^{de} = C \pmod N$$

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ ($N = pq$.)

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

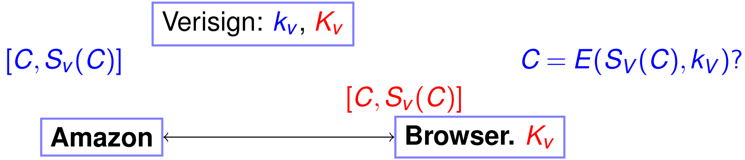
Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

$E(S_v(C), K_V) = (S_v(C))^e = (C^d)^e = C^{de} = C \pmod N$

Valid signature of Amazon certificate C!

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ $(N = pq.)$

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

$E(S_v(C), K_V) = (S_v(C))^e = (C^d)^e = C^{de} = C \pmod N$
Valid signature of Amazon certificate C!

Security: Eve can't forge unless she "breaks" RSA scheme.

RSA

RSA

Public Key Cryptography:

RSA

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m.$

RSA

Public Key Cryptography:

$D(E(m, K), k) = (m^e)^d \mod N = m$.

Signature scheme:

RSA

Public Key Cryptography:

$D(E(m,K),k) = (m^e)^d \mod N = m$.

Signature scheme:

$E(D(C,k),K) = (C^d)^e \mod N = C$

Other Eve.

Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.

Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.

2001..Doh.

Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.

2001..Doh.

... and August 28, 2011 announcement.

Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.

2001..Doh.

... and August 28, 2011 announcement.

DigiNotar Certificate issued for Microsoft!!!

Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.

2001..Doh.

... and August 28, 2011 announcement.

DigiNotar Certificate issued for Microsoft!!!

How does Microsoft get a CA to issue certificate to them ...

Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.

2001..Doh.

... and August 28, 2011 announcement.

DigiNotar Certificate issued for Microsoft!!!

How does Microsoft get a CA to issue certificate to them ...

and only them?

Summary.

Public-Key Encryption.

Summary.

Public-Key Encryption.

RSA Scheme:

Summary.

Public-Key Encryption.

RSA Scheme:
$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.
$E(x) = x^e \pmod{N}$.
$D(y) = y^d \pmod{N}$.

Summary.

Public-Key Encryption.

RSA Scheme:
$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.
$E(x) = x^e \pmod{N}$.
$D(y) = y^d \pmod{N}$.

Repeated Squaring $\implies$ efficiency.

Summary.

Public-Key Encryption.

RSA Scheme:
$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.
$E(x) = x^e \pmod{N}$.
$D(y) = y^d \pmod{N}$.

Repeated Squaring $\Longrightarrow$ efficiency.

Fermat's Theorem + CRT $\Longrightarrow$ correctness.

Summary.

Public-Key Encryption.

RSA Scheme:
$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.
$E(x) = x^e \pmod{N}$.
$D(y) = y^d \pmod{N}$.

Repeated Squaring $\implies$ efficiency.

Fermat's Theorem + CRT $\implies$ correctness.

Good for Encryption

Summary.

Public-Key Encryption.

RSA Scheme:
$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.
$E(x) = x^e \pmod{N}$.
$D(y) = y^d \pmod{N}$.

Repeated Squaring $\implies$ efficiency.

Fermat's Theorem + CRT $\implies$ correctness.

Good for Encryption and Signature Schemes.

Midterm Review

Now...

First there was logic...

A statement is true or false.

First there was logic...

A statement is true or false.
Statements?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

$n > 3$?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

$x = y$?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

$x + y$?

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

$x + y$? No.

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

$(\forall x)\ P(x)$.

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

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

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

First there was logic...

A statement is true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

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

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

Connecting Statements

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

Connecting Statements

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

You got this!

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Proofs: truth table or manipulation of known formulas.

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Proofs: truth table or manipulation of known formulas.

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

..and then proofs...

Direct: $P \implies Q$

..and then proofs...

Direct: $P \implies Q$

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

..and then proofs...

Direct: $P \implies Q$

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

Approach: What is even?

..and then proofs...

Direct: $P \implies Q$

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

Example: finite set of primes

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

Example: finite set of primes does not exist.

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

Example: finite set of primes does not exist.

Example: rogue couple does not exist.

...jumping forward..

Contradiction in induction:

...jumping forward..

Contradiction in induction:

contradict place where induction step doesn't hold.

...jumping forward..

Contradiction in induction:

contradict place where induction step doesn’t hold.

Well Ordering Principle.

...jumping forward..

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

Well Ordering Principle.
  Stable Marriage:

...jumping forward..

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

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

...jumping forward..

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

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

...jumping forward..

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

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

...jumping forward..

Contradiction in induction:

contradict place where induction step doesn’t hold.

Well Ordering Principle.

Stable Marriage:

first day where women does not improve.

first day where any man rejected by optimal women.

Do not exist.

...jumping forward..

Contradiction in induction:

contradict place where induction step doesn't hold.

Well Ordering Principle.

Stable Marriage:

first day where women does not improve.

first day where any man rejected by optimal women.

Do not exist.

...and then induction...

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

...and then induction...

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

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

...and then induction...

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

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

Induction on n.

...and then induction...

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

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

Induction on n.

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

...and then induction...

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

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

Induction on n.

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

...and then induction...

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

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

Induction on n.

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

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

...and then induction...

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

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

Induction on n.

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

Divisible by 8.

...and then induction...

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

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

Induction on n.

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

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

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

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

Divisible by 8.

□

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Each person has completely ordered preference list

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

Pairing.

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Pairing.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Pairing.
Pairing with no rogue couples.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

No, for roommates problem.

TMA.

Traditional Marriage Algorithm:

TMA.

Traditional Marriage Algorithm:

Each Day:

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

Useful Algorithmic Definitions:

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

"Propose and Reject."

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

Key Property: Improvement Lemma:

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

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

Stability:

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

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

Stability: No rogue couple.

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

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

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

TMA.

Traditional Marriage Algorithm:

Each Day:
All men propose to favorite non-rejecting woman.
Every woman rejects all but best men who proposes.

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

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

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

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

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.

Optimality/Pessimal

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

Optimality/Pessimal

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

Optimality/Pessimal

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

Optimality/Pessimal

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

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

Optimality/Pessimal

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

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

Thm: TMA produces male optimal pairing, S.

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Thm: woman pessimal.

Optimality/Pessimal

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

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

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

Thm: woman pessimal.

 Man optimal $\Longrightarrow$ Woman pessimal.

Optimality/Pessimal

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

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

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

Thm: woman pessimal.

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

...Graphs...

$$G = (V, E)$$

...Graphs...

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

...Graphs...

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

...Graphs...

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

...Graphs...

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

Directed: ordered pair of vertices.

...Graphs...

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

Directed: ordered pair of vertices.

...Graphs...

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

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.

...Graphs...

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

Directed: ordered pair of vertices.

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

Pair of Vertices are Connected:

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

Connected Component: maximal set of connected vertices.

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

Connected Component: maximal set of connected vertices.

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

Connected Component: maximal set of connected vertices.

Connected Graph: one connected component.

Graph Algorithm: Eulerian Tour

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

Graph Algorithm: Eulerian Tour

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

Algorithm:

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Proof Idea: Original graph connected.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Proof Idea: Original graph connected.

Euler's formula

How many faces in a planar drawing of a tree?

Euler's formula

How many faces in a planar drawing of a tree?

1.

Euler's formula

How many faces in a planar drawing of a tree?

1.

How many edges?

Euler's formula

How many faces in a planar drawing of a tree?

1.

How many edges?

$v - 1$.

Euler's formula

How many faces in a planar drawing of a tree?

1.

How many edges?

$v-1$.

Euler's Formula: $v+f=e+2$

Euler's formula

How many faces in a planar drawing of a tree?

1.

How many edges?

$v-1$.

Euler's Formula: $v+f=e+2$

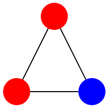
Induction Step: Adding an edge splits one face into two.

Graph Coloring.

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

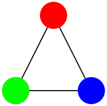
Graph Coloring.

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



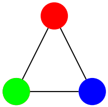
Graph Coloring.

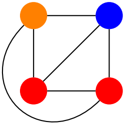
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



Graph Coloring.

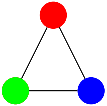
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.

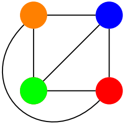




Graph Coloring.

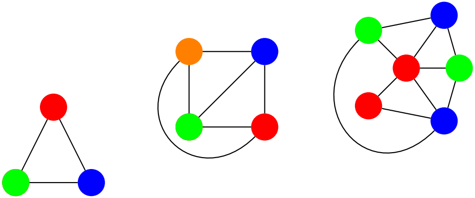
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.





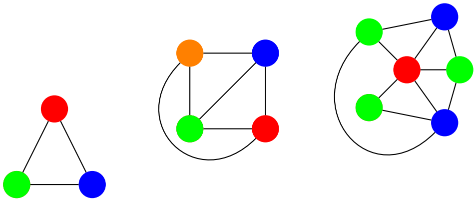
Graph Coloring.

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



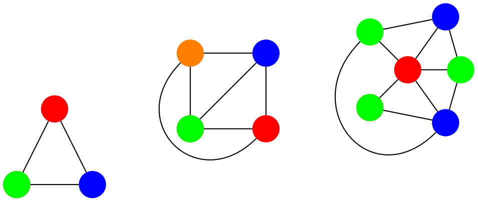
Graph Coloring.

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



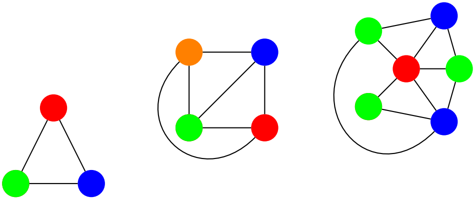
Graph Coloring.

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



Graph Coloring.

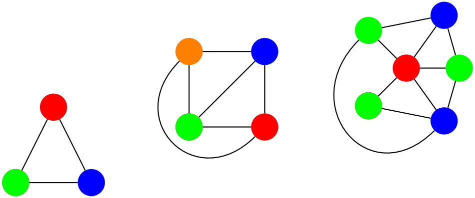
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



Notice that the last one, has one three colors.

Graph Coloring.

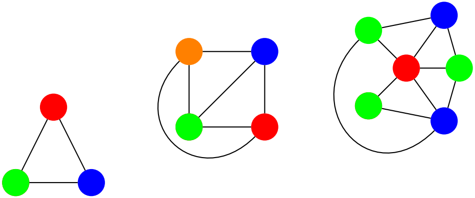
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



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

Graph Coloring.

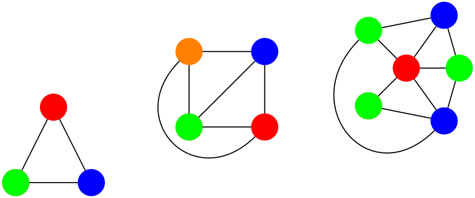
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



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

Graph Coloring.

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



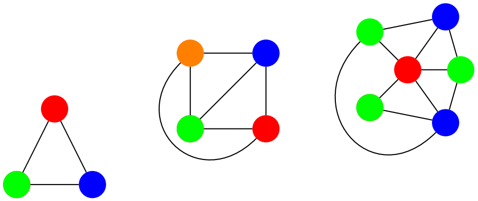
Notice that the last one, has one three colors.

Fewer colors than number of vertices.

Fewer colors than max degree node.

Graph Coloring.

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



Notice that the last one, has one three colors.

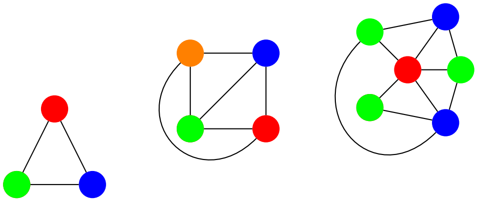
Fewer colors than number of vertices.

Fewer colors than max degree node.

Interesting things to do.

Graph Coloring.

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

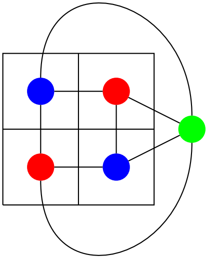


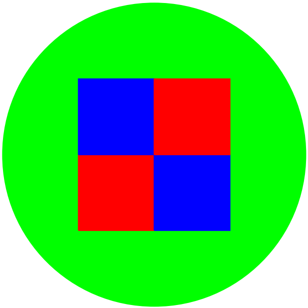
Notice that the last one, has one three colors.
  Fewer colors than number of vertices.
  Fewer colors than max degree node.

Interesting things to do. Algorithm!

Planar graphs and maps.

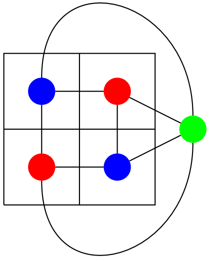
Planar graph coloring $\equiv$ map coloring.

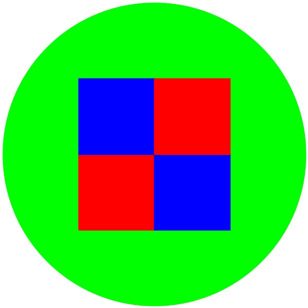




Planar graphs and maps.

Planar graph coloring $\equiv$ map coloring.





Four color theorem is about planar graphs!

Six color theorem.

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

Six color theorem.

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

Proof:

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

Total degree: $2e$

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

There exists a vertex with degree < 6

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

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

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

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

Remove vertex v of degree at most 5.

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

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

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

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

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

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

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

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

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

Six color theorem.

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

Proof:
Recall: $e \leq 3v - 6$ for any planar graph where $v > 2$.
From Euler's Formula.

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

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

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

Five color theorem: summary.

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

Five color theorem: summary.

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

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

Five color theorem: summary.

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

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

Proof:

Five color theorem: summary.

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

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

Proof: Again with the degree 5 vertex.

Five color theorem: summary.

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

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

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

Five color theorem: summary.

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

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

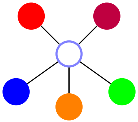
Proof: Again with the degree 5 vertex. Again recurse.

Five color theorem: summary.

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

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

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



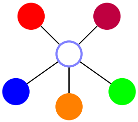
Either switch green.

Five color theorem: summary.

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

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

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



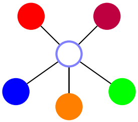
Either switch green.
Or try switching orange.

Five color theorem: summary.

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

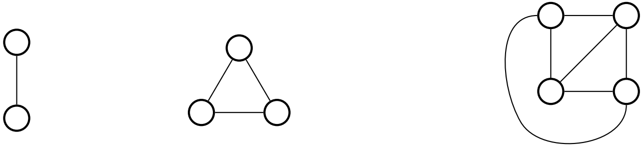
Theorem: Every planar graph can be colored with five colors.

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

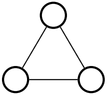


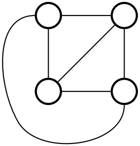
Either switch green.
Or try switching orange.
One will work.

Graph Types: Complete Graph.



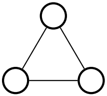
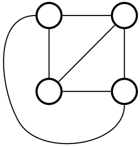
Graph Types: Complete Graph.





K_n, $|V| = n$

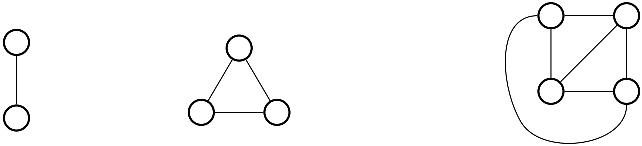
Graph Types: Complete Graph.

K_n, $|V| = n$

every edge present.

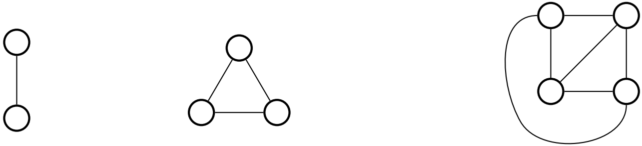
Graph Types: Complete Graph.



K_n, $|V| = n$

every edge present.
degree of vertex?

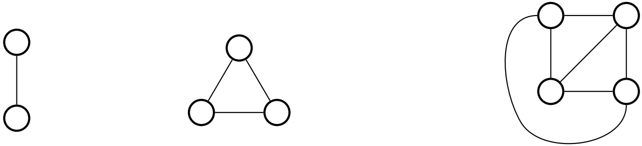
Graph Types: Complete Graph.



K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Graph Types: Complete Graph.

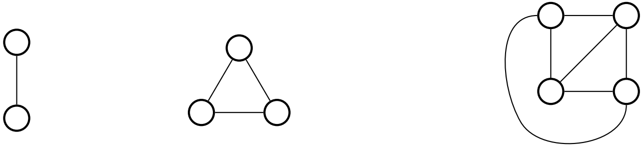


K_n, $|V| = n$

- every edge present.
- degree of vertex? $|V| - 1$.

Very connected.

Graph Types: Complete Graph.

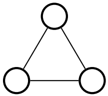


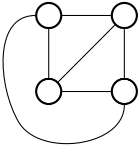
K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Very connected.
Lots of edges:

Graph Types: Complete Graph.



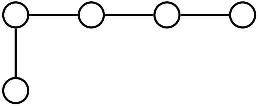
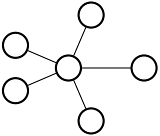
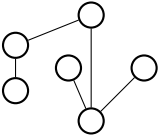


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

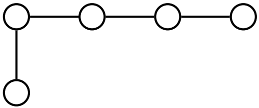
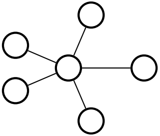
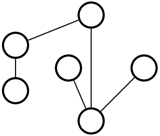
Very connected.
Lots of edges: $n(n-1)/2$.

Trees.

Definitions:

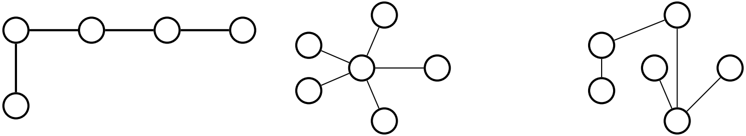
Trees.

Definitions:

A connected graph without a cycle.

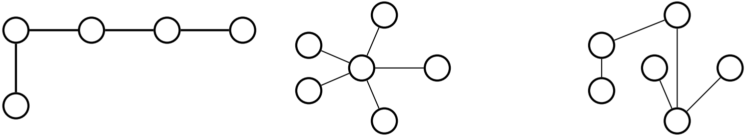
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.

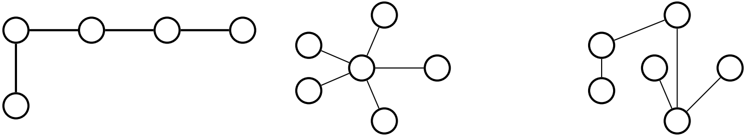
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.

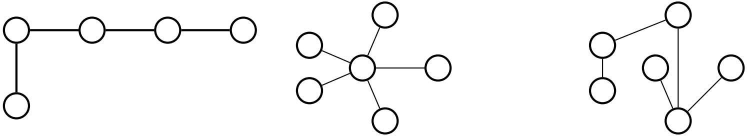
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

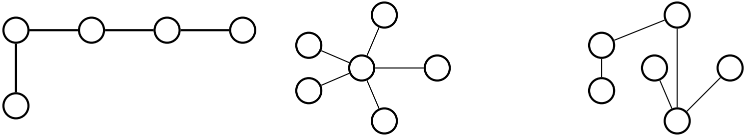
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

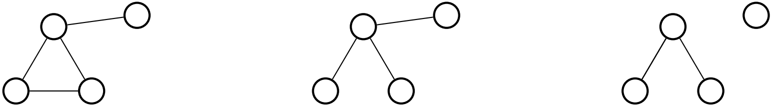
Trees.



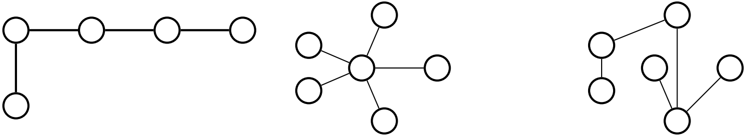
Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



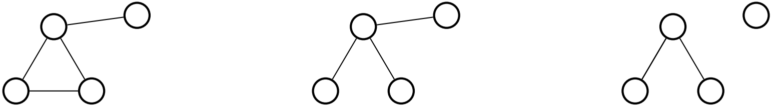
Trees.



Definitions:

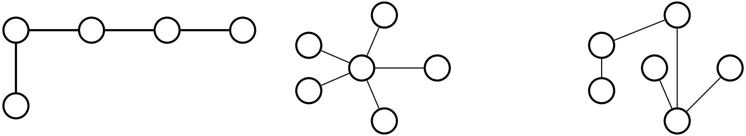
A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



Minimally connected, minimum number of edges to connect.

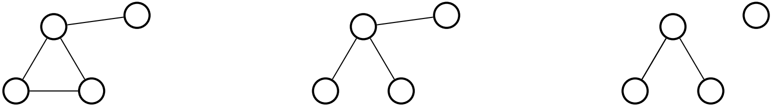
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

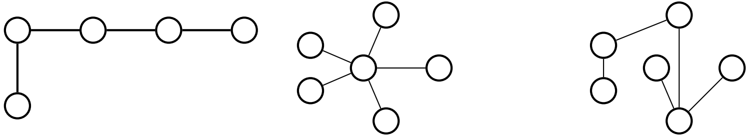
To tree or not to tree!



Minimally connected, minimum number of edges to connect.

Property:

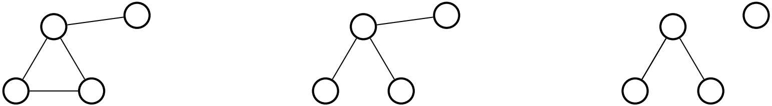
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



Minimally connected, minimum number of edges to connect.

Property:
A single node removal results in components of size $\leq |V|/2$.

Hypercube

Hypercubes.

Hypercube

Hypercubes. Really connected.

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0, 1\}^n$,

Hypercube

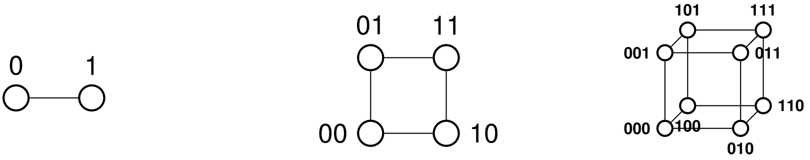
Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0, 1\}^n$,
$|E| = \{(x, y) | x$ and y differ in one bit position.$\}$

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0,1\}^n$,
$|E| = \{(x,y) | x$ and y differ in one bit position.$\}$



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

Recursive Definition.

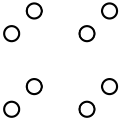
A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.

Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

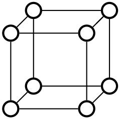
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

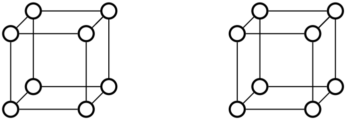
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

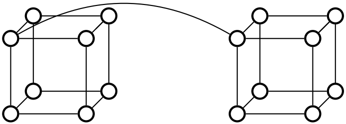
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

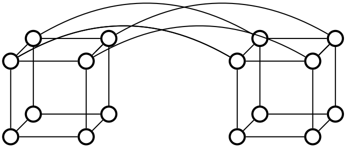
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

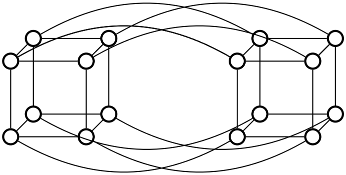
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Hypercube:properties

Rudrata Cycle: cycle that visits every node.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
  Eulerian?

Hypercube:properties

Rudrata Cycle: cycle that visits every node.

Eulerian? If *n* is even.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
  Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
  Best cut?

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
 Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
 Best cut? Cut apart subcubes:

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
 Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
 Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI:

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
 Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
 Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

 FYI: Also cuts represent boolean functions.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
$000100 \rightarrow 100100 \rightarrow 101100 \rightarrow 101000$

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
000100 $\rightarrow$ 100100 $\rightarrow$ 101100 $\rightarrow$ 101000
Correct bits in string, moves along path in hypercube!

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
000100 $\rightarrow$ 100100 $\rightarrow$ 101100 $\rightarrow$ 101000
Correct bits in string, moves along path in hypercube!

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
$000100 \rightarrow 100100 \rightarrow 101100 \rightarrow 101000$
Correct bits in string, moves along path in hypercube!

Good communication network!

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0,\ldots,m-1\}$

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

or if the remainder of i divided by m is a.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

or if the remainder of i divided by m is a.

Can do calculations by taking remainders

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0, \ldots, m-1\}$
    and integer $i \equiv a \pmod{m}$
      if $i = a + km$ for integer k.
        or if the remainder of i divided by m is a.

Can do calculations by taking remainders
  at the beginning,
    in the middle

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0,\ldots,m-1\}$
    and integer $i \equiv a \pmod{m}$
      if $i = a + km$ for integer k.
        or if the remainder of i divided by m is a.

Can do calculations by taking remainders
  at the beginning,
    in the middle
        or at the end.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

or if the remainder of i divided by m is a.

Can do calculations by taking remainders

at the beginning,

in the middle

or at the end.

$58 + 32 = 90 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

or if the remainder of i divided by m is a.

Can do calculations by taking remainders

at the beginning,

in the middle

or at the end.

$58 + 32 = 90 = 6 \pmod{7}$

$58 + 32 = 2 + 4 = 6 \pmod{7}$

$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0 - 3 = 7 - 3 = 4 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58+32 = 90 = 6 \pmod{7}$
$58+32 = 2+4 = 6 \pmod{7}$
$58+32 = 2+-3 = -1 = 6 \pmod{7}$

Negative numbers work the way you are used to.
$-3 = 0-3 = 7-3 = 4 \pmod{7}$

Additive inverses are intuitively negative numbers.

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$?

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$?

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5
$5^{-1} \pmod{7}$? 3

Inverse Unique?

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod{7}$? 5
$5^{-1} \pmod{7}$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod{n}$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$?

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

See,

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

See,... no inverse!

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$

$ax = bx = 1 \pmod n$

$axb = bxb = b \pmod n$

$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$

$\{3, 6, 3, 6, 3\}$

See,... no inverse!

$x = kd, m = jd$.

Modular Arithmetic and multiplicative inverses.

$3^{-1} \pmod 7$? 5
$5^{-1} \pmod 7$? 3

Inverse Unique? Yes.
Proof: a and b inverses of $x \pmod n$
$ax = bx = 1 \pmod n$
$axb = bxb = b \pmod n$
$a = b \pmod n$.

$3^{-1} \pmod 6$? No, no, no....

$\{3(1), 3(2), 3(3), 3(4), 3(5)\}$
$\{3, 6, 3, 6, 3\}$

See,... no inverse!

$x = kd, m = jd$.
$\ell x + im = \ell kd + ijd = d(\ell k + ij) \not\equiv 1 \pmod m$

Modular Arithmetic Inverses and GCD

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

Finding gcd.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm!

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case?

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y)

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x, 0) = x$.

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax+by$

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse!

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm$

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax+by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax+bm = ax \pmod m$.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.
gcd produces 1

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod m$.

Idea: egcd.
gcd produces 1
by adding and subtracting multiples of x and y

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.
gcd produces 1
by adding and subtracting multiples of x and y

Hand calculation: egcd.

Extended GCD: egcd$(7,60) = 1$.

Hand calculation: egcd.

Extended GCD: egcd$(7, 60) = 1$.
egcd(7,60).

Hand calculation: egcd.

Extended GCD: egcd$(7,60) = 1$.
egcd(7,60).

$$7(0) + 60(1) = 60$$

Hand calculation: egcd.

Extended GCD: egcd$(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \end{aligned}$$

Hand calculation: egcd.

Extended GCD: $\text{egcd}(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4
\end{aligned}
$$

Hand calculation: egcd.

Extended GCD: $\text{egcd}(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3
\end{aligned}
$$

Hand calculation: egcd.

Extended GCD: $\text{egcd}(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}
$$

Hand calculation: egcd.

Extended GCD: $\text{egcd}(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \\ 7(9)+60(-1) &= 3 \\ 7(-17)+60(2) &= 1 \end{aligned}$$

Hand calculation: egcd.

Extended GCD: $\text{egcd}(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0)+60(1) &= 60\\
7(1)+60(0) &= 7\\
7(-8)+60(1) &= 4\\
7(9)+60(-1) &= 3\\
7(-17)+60(2) &= 1
\end{aligned}
$$

Confirm:

Hand calculation: egcd.

Extended GCD: $\text{egcd}(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0)+60(1) &= 60\\
7(1)+60(0) &= 7\\
7(-8)+60(1) &= 4\\
7(9)+60(-1) &= 3\\
7(-17)+60(2) &= 1
\end{aligned}
$$

Confirm: $-119+120=1$

Hand calculation: egcd.

Extended GCD: $\text{egcd}(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0)+60(1) &= 60 \\
7(1)+60(0) &= 7 \\
7(-8)+60(1) &= 4 \\
7(9)+60(-1) &= 3 \\
7(-17)+60(2) &= 1
\end{aligned}
$$

Confirm: $-119+120=1$

$d = e^{-1} = -17 = 43 = \pmod{60}$

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.
Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.

Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.

Same as domain.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.

Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.

Same as domain. Product of both same.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.

Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.

Same as domain. Product of both same.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod p$ if $a \neq \pmod p$.

Proof: Look at range of $f(x) = ax \pmod p$ on $\{1, \ldots, p-1\}$.

Same as domain. Product of both same.

Extra factor of a^{p-1} on one side.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.

Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.

Same as domain. Product of both same.

Extra factor of a^{p-1} on one side. Must be 1.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.

Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.

Same as domain. Product of both same.

Extra factor of a^{p-1} on one side. Must be 1.

CRT:

Unique $x \pmod{mn}$ where $a \pmod{m}$ and $b \pmod{n}$ $gcd(x, m) = 1$.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.

Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.

Same as domain. Product of both same.

Extra factor of a^{p-1} on one side. Must be 1.

CRT:

Unique $x \pmod{mn}$ where $a \pmod{m}$ and $b \pmod{n}$ $gcd(x, m) = 1$.

Proof: Zero and One. Make u which is $0 \pmod{m}$ and $1 \pmod{n}$.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.
Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.
Same as domain. Product of both same.
Extra factor of a^{p-1} on one side. Must be 1.

CRT:
Unique $x \pmod{mn}$ where $a \pmod{m}$ and $b \pmod{n}$ $gcd(x,m) = 1$.
Proof: Zero and One. Make u which is $0 \pmod{m}$ and $1 \pmod{n}$.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.
Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.
Same as domain. Product of both same.
Extra factor of a^{p-1} on one side. Must be 1.

CRT:
Unique $x \pmod{mn}$ where $a \pmod{m}$ and $b \pmod{n}$ $gcd(x,m) = 1$.
Proof: Zero and One. Make u which is $0 \pmod{m}$ and $1 \pmod{n}$.

Repeated Squaring Idea for computing x^a efficiently,

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod p$ if $a \neq \pmod p$.
Proof: Look at range of $f(x) = ax \pmod p$ on $\{1, \ldots, p-1\}$.
Same as domain. Product of both same.
Extra factor of a^{p-1} on one side. Must be 1.

CRT:
Unique $x \pmod{mn}$ where $a \pmod m$ and $b \pmod n$ $gcd(x, m) = 1$.
Proof: Zero and One. Make u which is $0 \pmod m$ and $1 \pmod n$.

Repeated Squaring Idea for computing x^a efficiently,
Idea: Squaring builds big exponents that are powers of two.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.
Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.
Same as domain. Product of both same.
Extra factor of a^{p-1} on one side. Must be 1.

CRT:
Unique $x \pmod{mn}$ where $a \pmod{m}$ and $b \pmod{n}$ $gcd(x, m) = 1$.
Proof: Zero and One. Make u which is $0 \pmod{m}$ and $1 \pmod{n}$.

Repeated Squaring Idea for computing x^a efficiently,
Idea: Squaring builds big exponents that are powers of two.
Write exponent in binary.

Fermat/CRT/RSA

Fermat: $a^{p-1} = 1 \pmod{p}$ if $a \neq \pmod{p}$.
  Proof: Look at range of $f(x) = ax \pmod{p}$ on $\{1, \ldots, p-1\}$.
    Same as domain. Product of both same.
    Extra factor of a^{p-1} on one side. Must be 1.

CRT:
Unique $x \pmod{mn}$ where $a \pmod{m}$ and $b \pmod{n}$ $gcd(x, m) = 1$.
  Proof: Zero and One. Make u which is $0 \pmod{m}$ and $1 \pmod{n}$.

Repeated Squaring Idea for computing x^a efficiently,
  Idea: Squaring builds big exponents that are powers of two.
    Write exponent in binary.
  $O(n^3)$ time.

Midterm format

Time: 110 minutes.

Midterm format

Time: 110 minutes.

Some short answers.

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.

Midterm format

Time: 110 minutes.

Some short answers.
  Get at ideas that you learned.
   Know material well:

Midterm format

Time: 110 minutes.

Some short answers.

Get at ideas that you learned.

Know material well: fast,

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.
Know material well: fast, correct.

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.
Know material well: fast, correct.
Know material medium:

Midterm format

Time: 110 minutes.

Some short answers.

Get at ideas that you learned.

Know material well: fast, correct.

Know material medium: slower,

Midterm format

Time: 110 minutes.

Some short answers.

Get at ideas that you learned.

Know material well: fast, correct.

Know material medium: slower, less correct.

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.
Know material well: fast, correct.
Know material medium: slower, less correct.
Know material not so well:

Midterm format

Time: 110 minutes.

Some short answers.
  Get at ideas that you learned.
   Know material well: fast, correct.
   Know material medium: slower, less correct.
   Know material not so well: Uh oh.

Midterm format

Time: 110 minutes.

Some short answers.

Get at ideas that you learned.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.

Midterm format

Time: 110 minutes.

Some short answers.
 Get at ideas that you learned.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.
 Proofs,

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.
Proofs, algorithms,

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.
Proofs, algorithms, properties.

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.
Proofs, algorithms, properties.
Not so much calculation.

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.
Proofs, algorithms, properties.
Not so much calculation.

Midterm format

Time: 110 minutes.

Some short answers.
Get at ideas that you learned.

Know material well:	fast, correct.
Know material medium:	slower, less correct.
Know material not so well:	Uh oh.

Some longer questions.
Proofs, algorithms, properties.
Not so much calculation.

See piazza for more resources.

Midterm format

Time: 110 minutes.

Some short answers.
- Get at ideas that you learned.
  - Know material well: fast, correct.
  - Know material medium: slower, less correct.
  - Know material not so well: Uh oh.

Some longer questions.
- Proofs, algorithms, properties.
- Not so much calculation.

See piazza for more resources.
- E.g., TA videos for past exams.

Wrapup.

Wrapup.

Other issues....

Wrapup.

Other issues....
sp19@eecs70.org

Wrapup.

Other issues....
  sp19@eecs70.org

Wrapup.

Other issues....
sp19@eecs70.org

Wrapup.

Other issues....
sp19@eecs70.org

Good Studying!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!!!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!!!!!!

Wrapup.

Other issues....
sp19@eecs70.org

Good Studying!!!!!!!!!

Wrapup.

Other issues....
sp19@eecs70.org

Good Studying!!!!!!!!!!

Wrapup.

Other issues....
sp19@eecs70.org

Good Studying!!!!!!!!!!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!!!!!!!!!!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!!!!!!!!!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!!!!!!!!!!!!!

Wrapup.

Other issues....

sp19@eecs70.org

Good Studying!!!!!!!!!!!!!!

Wrapup.

Other issues....
sp19@eecs70.org

Good Studying!!!!!!!!!!!!!!!

Wrapup.

Other issues....
  sp19@eecs70.org

Good Studying!!!!!!!!!!!!!!!!

Wrapup.

Other issues....
sp19@eecs70.org

Good Studying!!!!!!!!!!!!!!!!!