Today

Finish Euclid.

Today

Finish Euclid.

Bijection/CRT/Isomorphism.

Today

Finish Euclid.

Bijection/CRT/Isomorphism.

Review for Midterm.

Finding an inverse?

We showed how to efficiently tell if there is an inverse.

Finding an inverse?

We showed how to efficiently tell if there is an inverse.

Extend euclid to find inverse.

Euclid's GCD algorithm.

```
(define (euclid x y)
 (if (= y 0)
     x
    (euclid  y (mod x y))))
```

Euclid's GCD algorithm.

```
(define (euclid x y)
 (if (= y 0)
     x
    (euclid  y (mod x y))))
```

Computes the $\gcd(x, y)$ in $O(n)$ divisions.

Euclid's GCD algorithm.

```
(define (euclid x y)
 (if (= y 0)
     x
    (euclid  y (mod x y))))
```

Computes the $\gcd(x, y)$ in $O(n)$ divisions.

For x and m, if $\gcd(x, m) = 1$ then x has an inverse modulo m.

Multiplicative Inverse.

GCD algorithm used to tell **if** there is a multiplicative inverse.

Multiplicative Inverse.

GCD algorithm used to tell **if** there is a multiplicative inverse.

How do we **find** a multiplicative inverse?

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that

$$ax + by$$

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that
$ax + by = d$ where $d = gcd(x, y)$.

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that

$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that

$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that

$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

By extended GCD theorem, when $\gcd(x, m) = 1$.

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that
$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

By extended GCD theorem, when $\gcd(x, m) = 1$.

$$ax + bm = 1$$

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that

$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

By extended GCD theorem, when $\gcd(x, m) = 1$.

$$ax + bm = 1$$
$$ax \equiv 1 - bm \equiv 1 \pmod{m}.$$

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that

$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

By extended GCD theorem, when $\gcd(x, m) = 1$.

$$\begin{aligned} ax + bm &= 1 \\ ax \equiv 1 - bm &\equiv 1 \pmod{m}. \end{aligned}$$

So a multiplicative inverse of $x \pmod{m}$!!

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that
$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

By extended GCD theorem, when $\gcd(x, m) = 1$.

$$\begin{aligned} ax + bm &= 1 \\ ax \equiv 1 - bm &\equiv 1 \pmod{m}. \end{aligned}$$

So a multiplicative inverse of $x \pmod{m}$!!
Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that
$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

By extended GCD theorem, when $\gcd(x, m) = 1$.

$$ax + bm = 1$$
$$ax \equiv 1 - bm \equiv 1 \pmod{m}.$$

So a multiplicative inverse of $x \pmod{m}$!!
Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$$(3)12 + (-1)35 = 1.$$

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that

$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

By extended GCD theorem, when $\gcd(x, m) = 1$.

$$\begin{aligned} ax + bm &= 1 \\ ax &\equiv 1 - bm \equiv 1 \pmod{m}. \end{aligned}$$

So a multiplicative inverse of $x \pmod{m}$!!
Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$$(3)12 + (-1)35 = 1.$$

$a = 3$ and $b = -1$.

Extended GCD

Euclid's Extended GCD Theorem: For any x, y there are integers a, b such that

$$ax + by = d \qquad \text{where } d = gcd(x, y).$$

"Make d out of sum of multiples of x and y."

What is multiplicative inverse of x modulo m?

By extended GCD theorem, when $\gcd(x, m) = 1$.

$$ax + bm = 1$$
$$ax \equiv 1 - bm \equiv 1 \pmod{m}.$$

So a multiplicative inverse of $x \pmod{m}$!!
Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$$(3)12 + (-1)35 = 1.$$

$a = 3$ and $b = -1$.
The multiplicative inverse of $12 \pmod{35}$ is 3.

Make *d* out of *x* and *y*..?

```
gcd(35,12)
```

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
```

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
```

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Algorithm finally returns 1.

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Algorithm finally returns 1.

But we want 1 from sum of multiples of 35 and 12?

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Algorithm finally returns 1.

But we want 1 from sum of multiples of 35 and 12?

Get 1 from 12 and 11.

Make d out of x and y..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Algorithm finally returns 1.

But we want 1 from sum of multiples of 35 and 12?

Get 1 from 12 and 11.
$1 = 12 - (1)11$

Make d out of x and y..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Algorithm finally returns 1.

But we want 1 from sum of multiples of 35 and 12?

Get 1 from 12 and 11.
$1 = 12 - (1)11 = 12 - (1)(35 - (2)12)$
Get 11 from 35 and 12 and plugin....

Make d out of x and y..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Algorithm finally returns 1.

But we want 1 from sum of multiples of 35 and 12?

Get 1 from 12 and 11.
$1 = 12 - (1)11 = 12 - (1)(35 - (2)12) = (3)12 + (-1)35$
Get 11 from 35 and 12 and plugin.... Simplify.

Make *d* out of *x* and *y*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Algorithm finally returns 1.

But we want 1 from sum of multiples of 35 and 12?

Get 1 from 12 and 11.
$1 = 12 - (1)11 = 12 - (1)(35 - (2)12) = (3)12 + (-1)35$
Get 11 from 35 and 12 and plugin.... Simplify.

Make d out of x and y..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  gcd(11, 12%11)
      gcd(1,0)
        1
```

How did gcd get 11 from 35 and 12?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

How does gcd get 1 from 12 and 11?
$12 - \lfloor \frac{12}{11} \rfloor 11 = 12 - (1)11 = 1$

Algorithm finally returns 1.

But we want 1 from sum of multiples of 35 and 12?

Get 1 from 12 and 11.
$1 = 12 - (1)11 = 12 - (1)(35 - (2)12) = (3)12 + (-1)35$
Get 11 from 35 and 12 and plugin.... Simplify. $a = 3$ and $b = -1$.

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d, a, b): $d = gcd(a, b)$ and $d = ax + by$.

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d, a, b): $d = gcd(a, b)$ and $d = ax + by$.
Example:

```
ext-gcd(35,12)
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d,a,b): $d = gcd(a,b)$ and $d = ax + by$.
Example:

```
ext-gcd(35,12)
  ext-gcd(12, 11)
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d,a,b): $d = gcd(a,b)$ and $d = ax + by$.
Example:

```
ext-gcd(35,12)
  ext-gcd(12, 11)
    ext-gcd(11, 1)
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d, a, b): $d = gcd(a, b)$ and $d = ax + by$.
Example:

```
ext-gcd(35,12)
  ext-gcd(12, 11)
    ext-gcd(11, 1)
      ext-gcd(1,0)
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d,a,b): $d = gcd(a,b)$ and $d = ax + by$.
Example: $a - \lfloor x/y \rfloor \cdot b =$

```
ext-gcd(35,12)
  ext-gcd(12, 11)
    ext-gcd(11, 1)
      ext-gcd(1,0)
      return (1,1,0) ;; 1 = (1)1 + (0) 0
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d,a,b): $d = gcd(a,b)$ and $d = ax + by$.
Example: $a - \lfloor x/y \rfloor \cdot b = 1 - \lfloor 11/1 \rfloor \cdot 0 = 1$

```
ext-gcd(35,12)
  ext-gcd(12, 11)
    ext-gcd(11, 1)
      ext-gcd(1,0)
      return (1,1,0) ;; 1 = (1)1 + (0) 0
    return (1,0,1)   ;; 1 = (0)11 + (1)1
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d,a,b): $d = gcd(a,b)$ and $d = ax + by$.
Example: $a - \lfloor x/y \rfloor \cdot b = 0 - \lfloor 12/11 \rfloor \cdot 1 = -1$

```
ext-gcd(35,12)
  ext-gcd(12, 11)
    ext-gcd(11, 1)
      ext-gcd(1,0)
      return (1,1,0) ;; 1 = (1)1 + (0) 0
    return (1,0,1)   ;; 1 = (0)11 + (1)1
  return (1,1,-1)    ;; 1 = (1)12 + (-1)11
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d,a,b): $d = gcd(a,b)$ and $d = ax + by$.
Example: $a - \lfloor x/y \rfloor \cdot b1 = \lfloor 35/12 \rfloor \cdot (-1) = 3$

```
    ext-gcd(35,12)
      ext-gcd(12, 11)
        ext-gcd(11, 1)
          ext-gcd(1,0)
          return (1,1,0) ;; 1 = (1)1 + (0) 0
        return (1,0,1)   ;; 1 = (0)11 + (1)1
      return (1,1,-1)    ;; 1 = (1)12 + (-1)11
   return (1,-1, 3)      ;; 1 = (-1)35 +(3)12
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Claim: Returns (d, a, b): $d = gcd(a, b)$ and $d = ax + by$.
Example:

```
    ext-gcd(35,12)
      ext-gcd(12, 11)
        ext-gcd(11, 1)
          ext-gcd(1,0)
          return (1,1,0) ;; 1 = (1)1 + (0) 0
        return (1,0,1)   ;; 1 = (0)11 + (1)1
      return (1,1,-1)    ;; 1 = (1)12 + (-1)11
   return (1,-1, 3)      ;; 1 = (-1)35 +(3)12
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Extended GCD Algorithm.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Theorem: Returns (d,a,b), where $d = gcd(a,b)$ and

$$d = ax + by.$$

Correctness.

Proof: Strong Induction.[1]

[1] Assume d is $gcd(x, y)$ by previous proof.

Correctness.

Proof: Strong Induction.[1]
Base: ext-gcd$(x, 0)$ returns $(d = x, 1, 0)$ with $x = (1)x + (0)y$.

[1] Assume d is $gcd(x, y)$ by previous proof.

Correctness.

Proof: Strong Induction.[1]
Base: ext-gcd$(x, 0)$ returns $(d = x, 1, 0)$ with $x = (1)x + (0)y$.

Induction Step: Returns (d, A, B) with $d = Ax + By$
Ind hyp: **ext-gcd**$(y, \mod (x, y))$ returns (d, a, b) with
$d = ay + b(\mod (x, y))$

[1] Assume d is $gcd(x, y)$ by previous proof.

Correctness.

Proof: Strong Induction.[1]
Base: ext-gcd($x,0$) returns ($d = x, 1, 0$) with $x = (1)x + (0)y$.

Induction Step: Returns (d, A, B) with $d = Ax + By$
Ind hyp: **ext-gcd**(y, $\mod(x,y)$) returns (d, a, b) with
$d = ay + b(\mod(x,y))$

ext-gcd(x,y) calls **ext-gcd**(y, $\mod(x,y)$) so

[1] Assume d is $gcd(x,y)$ by previous proof.

Correctness.

Proof: Strong Induction.[1]
Base: ext-gcd$(x, 0)$ returns $(d = x, 1, 0)$ with $x = (1)x + (0)y$.

Induction Step: Returns (d, A, B) with $d = Ax + By$
Ind hyp: **ext-gcd**$(y, \mod(x, y))$ returns (d, a, b) with
$d = ay + b(\mod(x, y))$

ext-gcd(x, y) calls **ext-gcd**$(y, \mod(x, y))$ so

$$d = ay + b \cdot (\mod(x, y))$$

[1] Assume d is $gcd(x, y)$ by previous proof.

Correctness.

Proof: Strong Induction.[1]
Base: ext-gcd$(x,0)$ returns $(d=x,1,0)$ with $x=(1)x+(0)y$.

Induction Step: Returns (d,A,B) with $d=Ax+By$
Ind hyp: **ext-gcd**$(y,\ \bmod(x,y))$ returns (d,a,b) with
$d=ay+b(\bmod(x,y))$

ext-gcd(x,y) calls **ext-gcd**$(y,\ \bmod(x,y))$ so

$$\begin{aligned} d &= ay+b\cdot(\bmod(x,y)) \\ &= ay+b\cdot(x-\lfloor\frac{x}{y}\rfloor y) \end{aligned}$$

[1] Assume d is $gcd(x,y)$ by previous proof.

Correctness.

Proof: Strong Induction.[1]
Base: ext-gcd$(x,0)$ returns $(d=x,1,0)$ with $x=(1)x+(0)y$.

Induction Step: Returns (d,A,B) with $d=Ax+By$
Ind hyp: **ext-gcd**$(y, \mod(x,y))$ returns (d,a,b) with
$d=ay+b(\mod(x,y))$

ext-gcd(x,y) calls **ext-gcd**$(y, \mod(x,y))$ so

$$\begin{aligned} d &= ay+b\cdot(\mod(x,y)) \\ &= ay+b\cdot(x-\lfloor\frac{x}{y}\rfloor y) \\ &= bx+(a-\lfloor\frac{x}{y}\rfloor\cdot b)y \end{aligned}$$

[1] Assume d is $gcd(x,y)$ by previous proof.

Correctness.

Proof: Strong Induction.[1]
Base: ext-gcd$(x,0)$ returns $(d = x, 1, 0)$ with $x = (1)x + (0)y$.

Induction Step: Returns (d, A, B) with $d = Ax + By$
Ind hyp: **ext-gcd**$(y, \mod(x,y))$ returns (d,a,b) with
$d = ay + b(\mod(x,y))$

ext-gcd(x,y) calls **ext-gcd**$(y, \mod(x,y))$ so

$$\begin{aligned} d &= ay + b \cdot (\mod(x,y)) \\ &= ay + b \cdot (x - \lfloor \frac{x}{y} \rfloor y) \\ &= bx + (a - \lfloor \frac{x}{y} \rfloor \cdot b)y \end{aligned}$$

And ext-gcd returns $(d, b, (a - \lfloor \frac{x}{y} \rfloor \cdot b))$ so theorem holds!

[1] Assume d is $gcd(x,y)$ by previous proof.

Correctness.

Proof: Strong Induction.[1]
Base: ext-gcd$(x, 0)$ returns $(d = x, 1, 0)$ with $x = (1)x + (0)y$.

Induction Step: Returns (d, A, B) with $d = Ax + By$
Ind hyp: **ext-gcd**$(y, \mod(x, y))$ returns (d, a, b) with
$d = ay + b(\mod(x, y))$

ext-gcd(x, y) calls **ext-gcd**$(y, \mod(x, y))$ so

$$\begin{aligned} d &= ay + b \cdot (\mod(x, y)) \\ &= ay + b \cdot (x - \lfloor \frac{x}{y} \rfloor y) \\ &= bx + (a - \lfloor \frac{x}{y} \rfloor \cdot b)y \end{aligned}$$

And ext-gcd returns $(d, b, (a - \lfloor \frac{x}{y} \rfloor \cdot b))$ so theorem holds! □

[1] Assume d is $gcd(x, y)$ by previous proof.

Review Proof: step.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Review Proof: step.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Recursively: $d = ay + b(x - \lfloor \frac{x}{y} \rfloor \cdot y)$

Review Proof: step.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Recursively: $d = ay + b(x - \lfloor \frac{x}{y} \rfloor \cdot y) \implies d = bx - (a - \lfloor \frac{x}{y} \rfloor b)y$

Review Proof: step.

```
ext-gcd(x,y)
  if y = 0 then return(x, 1, 0)
     else
         (d, a, b) := ext-gcd(y, mod(x,y))
         return (d, b, a - floor(x/y)  * b)
```

Recursively: $d = ay + b(x - \lfloor \frac{x}{y} \rfloor \cdot y) \implies d = bx - (a - \lfloor \frac{x}{y} \rfloor b)y$

Returns $(d, b, (a - \lfloor \frac{x}{y} \rfloor \cdot b))$.

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?

≤ 80 divisions.

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?
≤ 80 divisions.
versus $1,000,000$

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?
≤ 80 divisions.
versus $1,000,000$

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?
≤ 80 divisions.
versus $1,000,000$

Internet Security.

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?
≤ 80 divisions.
versus $1,000,000$

Internet Security.
Public Key Cryptography: 512 digits.

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?
≤ 80 divisions.
versus $1,000,000$

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?
≤ 80 divisions.
versus $1,000,000$

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.
$(10000000000000000000000000000000000000000000)^5$ divisions.

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?
≤ 80 divisions.
versus $1,000,000$

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.
$(10000000000000000000000000000000000000000000)^5$ divisions.

Internet Security:

Wrap-up

Conclusion: Can find multiplicative inverses in $O(n)$ time!

Very different from elementary school: try 1, try 2, try 3...

$2^{n/2}$

Inverse of $500,000,357$ modulo $1,000,000,000,000$?
≤ 80 divisions.
versus $1,000,000$

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.
$(10000000000000000000000000000000000000000000)^5$ divisions.

Internet Security: Next Week!

Bijections

Bijection is **one to one** and **onto.**
Bijection:

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \rightarrow B$.
Domain: A, Co-Domain: B.
Versus Range.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \rightarrow B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \rightarrow B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \rightarrow B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1, 1]$.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1, 1]$. Onto: $[-1, 1]$.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1, 1]$. Onto: $[-1, 1]$.
Not one-to-one.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. $\mathbf{sin}\ (x)$.
$A = B =$ reals.
Range is $[-1, 1]$. Onto: $[-1, 1]$.
Not one-to-one. $\mathbf{sin}\ (\pi) = \mathbf{sin}\ (0) = 0$.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1, 1]$. Onto: $[-1, 1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** $(0) = 0$.

Range Definition always is onto.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. $\textbf{sin}\,(x)$.
$A = B =$ reals.
Range is $[-1, 1]$. Onto: $[-1, 1]$.
Not one-to-one. $\textbf{sin}\,(\pi) = \textbf{sin}\,(0) = 0$.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1,1]$. Onto: $[-1,1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** (0) = 0.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.
$f : \{0,\ldots,m-1\} \to \{0,\ldots,m-1\}$.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1,1]$. Onto: $[-1,1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** $(0) = 0$.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.
$f : \{0,\ldots,m-1\} \to \{0,\ldots,m-1\}$.
Domain/Co-Domain: $\{0,\ldots,m-1\}$.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1,1]$. Onto: $[-1,1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** (0) = 0.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.
$f : \{0,\ldots,m-1\} \to \{0,\ldots,m-1\}$.
Domain/Co-Domain: $\{0,\ldots,m-1\}$.

When is it a bijection?

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1, 1]$. Onto: $[-1, 1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** (0) = 0.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.
$f : \{0, \ldots, m-1\} \to \{0, \ldots, m-1\}$.
Domain/Co-Domain: $\{0, \ldots, m-1\}$.

When is it a bijection?
When $gcd(a, m)$ is ....

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1,1]$. Onto: $[-1,1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** (0) = 0.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.
$f : \{0,\ldots,m-1\} \to \{0,\ldots,m-1\}$.
Domain/Co-Domain: $\{0,\ldots,m-1\}$.

When is it a bijection?
When $gcd(a,m)$ is ....?

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1,1]$. Onto: $[-1,1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** (0) = 0.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.
$f : \{0,\ldots,m-1\} \to \{0,\ldots,m-1\}$.
Domain/Co-Domain: $\{0,\ldots,m-1\}$.

When is it a bijection?
When $gcd(a,m)$ is ....? ... 1.

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1,1]$. Onto: $[-1,1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** (0) = 0.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.
$f : \{0,\ldots,m-1\} \to \{0,\ldots,m-1\}$.
Domain/Co-Domain: $\{0,\ldots,m-1\}$.

When is it a bijection?
When $gcd(a,m)$ is ....? ... 1.

Not Example: $a = 2$, $m = 4$,

Bijections

Bijection is **one to one** and **onto.**
Bijection:
$f : A \to B$.
Domain: A, Co-Domain: B.
Versus Range.
E.g. **sin** (x).
$A = B =$ reals.
Range is $[-1, 1]$. Onto: $[-1, 1]$.
Not one-to-one. **sin** $(\pi) =$ **sin** (0) = 0.

Range Definition always is onto.
Consider $f(x) = ax \mod m$.
$f : \{0, \ldots, m-1\} \to \{0, \ldots, m-1\}$.
Domain/Co-Domain: $\{0, \ldots, m-1\}$.

When is it a bijection?
When $gcd(a, m)$ is ....? ... 1.

Not Example: $a = 2$, $m = 4$, $f(0) = f(2) = 0 \pmod 4$.

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

Lots of Mods

$x = 5 \pmod{7}$ and $x = 3 \pmod{5}$.

What is $x \pmod{35}$?

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5.

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3.

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

If $x = 6 \pmod 7$

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

If $x = 6 \pmod 7$
then x is in $\{5, 12, 19, 26, 33\}$.

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

If $x = 6 \pmod 7$
then x is in $\{5, 12, 19, 26, 33\}$.

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

If $x = 6 \pmod 7$
then x is in $\{5, 12, 19, 26, 33\}$.

Oh, only 33 is $3 \pmod 5$.

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

If $x = 6 \pmod 7$
then x is in $\{5, 12, 19, 26, 33\}$.

Oh, only 33 is $3 \pmod 5$.

Hmmm...

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

If $x = 6 \pmod 7$
then x is in $\{5, 12, 19, 26, 33\}$.

Oh, only 33 is $3 \pmod 5$.

Hmmm... only one solution.

Lots of Mods

$x = 5 \pmod 7$ and $x = 3 \pmod 5$.

What is $x \pmod{35}$?

Let's try 5. Not $3 \pmod 5$!
Let's try 3. Not $5 \pmod 7$!

If $x = 6 \pmod 7$
then x is in $\{5, 12, 19, 26, 33\}$.

Oh, only 33 is $3 \pmod 5$.

Hmmm... only one solution.

A bit slow for large values.

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: Unique solution $\pmod{mn}$.

Simple Chinese Remainder Theorem.

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

CRT Thm: Unique solution $\pmod{mn}$.
Proof:

Simple Chinese Remainder Theorem.

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

CRT Thm: Unique solution $\pmod{mn}$.
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: Unique solution $\pmod{mn}$.
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: Unique solution $\pmod{mn}$.
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: Unique solution $\pmod{mn}$.
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: Unique solution $\pmod{mn}$.
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: Unique solution $\pmod{mn}$.

Proof:

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

$u = 0 \pmod{n}$ $\qquad u = 1 \pmod{m}$

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

$v = 1 \pmod{n}$ $\qquad v = 0 \pmod{m}$

Simple Chinese Remainder Theorem.

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

CRT Thm: Unique solution $\pmod{mn}$.
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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.

Simple Chinese Remainder Theorem.

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

CRT Thm: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$

Simple Chinese Remainder Theorem.

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

CRT Thm: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$

Simple Chinese Remainder Theorem.

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

CRT Thm: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod m$ since $bv = 0 \pmod m$ and $au = a \pmod m$
$x = b \pmod n$ since $au = 0 \pmod n$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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.
$\implies x - y \geq mn \implies x, y \not\in \{0, \ldots, mn-1\}$.

Simple Chinese Remainder Theorem.

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

CRT Thm: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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: Unique solution $\pmod{mn}$.
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 $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ 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. $\square$

Midterm Review

Now...

First there was logic...

A statement is a true or false.

First there was logic...

A statement is a true or false.
Statements?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

First there was logic...

A statement is a true or false.
Statements?
$3 = 4 - 1$? Statement!
$3 = 5$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

$n = 3$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

First there was logic...

A statement is a true or false.
Statements?

$3 = 4 - 1$? Statement!
$3 = 5$? Statement!
3 ? Not a statement!
$n = 3$? Not a statement...but a predicate.

Predicate: Statement with free variable(s).

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

$n > 3$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

$x = y$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

$x + y$?

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

$x + y$? No.

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

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

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

First there was logic...

A statement is a true or false.

Statements?

$3 = 4 - 1$? Statement!

$3 = 5$? Statement!

3 ? Not a statement!

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

Predicate: Statement with free variable(s).

Example: $x = 3$

Given a value for x, becomes a statement.

Predicate?

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

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

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

Quantifiers:

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

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

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

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

Connecting Statements

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

Connecting Statements

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

You got this!

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Proofs: truth table or manipulation of known formulas.

Connecting Statements

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

You got this!

Propositional Expressions and Logical Equivalence

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

Proofs: truth table or manipulation of known formulas.

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

..and then proofs...

Direct: $P \implies Q$

..and then proofs...

Direct: $P \implies Q$

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

..and then proofs...

Direct: $P \implies Q$

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

Approach: What is even?

..and then proofs...

Direct: $P \implies Q$

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

Contrapositive: $P \implies Q$

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

Example: finite set of primes

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

Example: finite set of primes does not exist.

..and then proofs...

Direct: $P \implies Q$

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

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

$a^2 = 4k^2$.

What is even?

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

Integers closed under multiplication!

a^2 is even.

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

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

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

Contradiction: P

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

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

Useful for prove something does not exist:

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

Example: finite set of primes does not exist.

Example: rogue couple does not exist.

...jumping forward..

Contradiction in induction:

...jumping forward..

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

...jumping forward..

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

Well Ordering Principle.

...jumping forward..

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

Well Ordering Principle.
 Stable Marriage:

...jumping forward..

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

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

...jumping forward..

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

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

...jumping forward..

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

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

...jumping forward..

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

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

...jumping forward..

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

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

...and then induction...

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

...and then induction...

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

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

...and then induction...

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

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

Induction on n.

...and then induction...

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

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

Induction on n.

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

...and then induction...

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

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

Induction on n.

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

...and then induction...

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

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

Induction on n.

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

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

...and then induction...

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

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

Induction on n.

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...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 $P(n+1)$

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

...and then induction...

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

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

Induction on n.

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

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

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

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

...and then induction...

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

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

Induction on n.

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

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

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

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

Divisible by 8.

...and then induction...

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

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

Induction on n.

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

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

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

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

Divisible by 8.

□

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

Each person has completely ordered preference list

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

Pairing.

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Pairing.

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

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

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

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

Stable Pairing.
Pairing with no rogue couples.

Stable Marriage: a study in definitions and WOP.

n-men, *n*-women.

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

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

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

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

Stable Marriage: a study in definitions and WOP.

n-men, n-women.

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

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

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

Stable Pairing.
Pairing with no rogue couples.

Does stable pairing exist?

No, for roommates problem.

TMA.

Traditional Marriage Algorithm:

TMA.

Traditional Marriage Algorithm:

Each Day:

TMA.

Traditional Marriage Algorithm:

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

TMA.

Traditional Marriage Algorithm:

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

TMA.

Traditional Marriage Algorithm:

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

Useful Algorithmic Definitions:

TMA.

Traditional Marriage Algorithm:

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

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

TMA.

Traditional Marriage Algorithm:

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

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

TMA.

Traditional Marriage Algorithm:

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

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

"Propose and Reject."

TMA.

Traditional Marriage Algorithm:

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

Key Property: Improvement Lemma:

TMA.

Traditional Marriage Algorithm:

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

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

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

Stability:

TMA.

Traditional Marriage Algorithm:

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

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

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

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

Stability: No rogue couple.

TMA.

Traditional Marriage Algorithm:

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

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

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

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

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

TMA.

Traditional Marriage Algorithm:

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

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

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

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

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

Optimality/Pessimal

Optimal partner if best partner in any stable pairing.

Optimality/Pessimal

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

Optimality/Pessimal

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

Optimality/Pessimal

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

Optimality/Pessimal

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

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

Optimality/Pessimal

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

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

Thm: TMA produces male optimal pairing, S.

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Optimality/Pessimal

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

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

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

Thm: woman pessimal.

Optimality/Pessimal

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

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

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

Thm: woman pessimal.

Man optimal $\implies$ Woman pessimal.

Optimality/Pessimal

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

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

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

Thm: woman pessimal.

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

...Graphs...

$G = (V, E)$

...Graphs...

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

...Graphs...

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

...Graphs...

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

...Graphs...

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

Directed: ordered pair of vertices.

...Graphs...

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

Directed: ordered pair of vertices.

...Graphs...

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

Directed: ordered pair of vertices.

Adjacent, Incident, Degree.

...Graphs...

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

Directed: ordered pair of vertices.

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

Pair of Vertices are Connected:

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

Connected Component: maximal set of connected vertices.

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

Connected Component: maximal set of connected vertices.

...Graphs...

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

Directed: ordered pair of vertices.

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

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

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

Connected Component: maximal set of connected vertices.

Connected Graph: one connected component.

Graph Algorithm: Eulerian Tour

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

Graph Algorithm: Eulerian Tour

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

Algorithm:

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Graph Algorithm: Eulerian Tour

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

Algorithm:
Take a walk using each edge at most once.
Property: return to starting point.
Proof Idea: Even degree.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Graph Algorithm: Eulerian Tour

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

Algorithm:

Take a walk using each edge at most once.

Property: return to starting point.

Proof Idea: Even degree.

Recurse on connected components.

Put together.

Property: walk visits every component.

Proof Idea: Original graph connected.

Graph Algorithm: Eulerian Tour

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

Algorithm:
Take a walk using each edge at most once.
Property: return to starting point.
Proof Idea: Even degree.

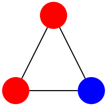
Recurse on connected components.
Put together.
Property: walk visits every component.
Proof Idea: Original graph connected.

Graph Coloring.

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

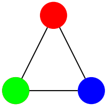
Graph Coloring.

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



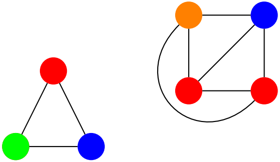
Graph Coloring.

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



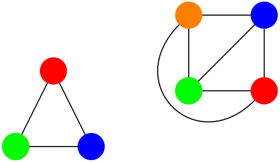
Graph Coloring.

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



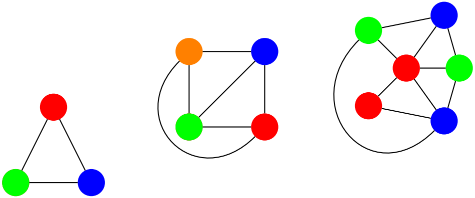
Graph Coloring.

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



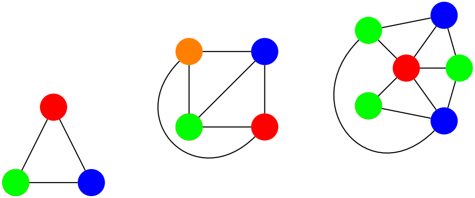
Graph Coloring.

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



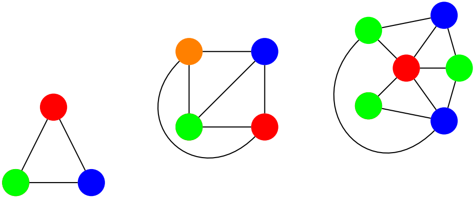
Graph Coloring.

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



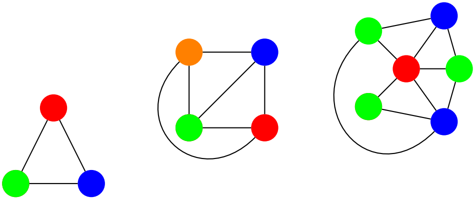
Graph Coloring.

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



Graph Coloring.

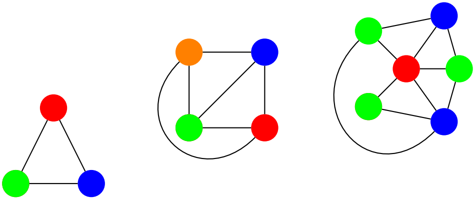
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



Notice that the last one, has one three colors.

Graph Coloring.

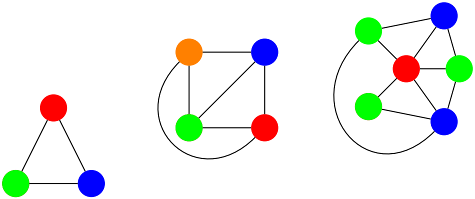
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



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

Graph Coloring.

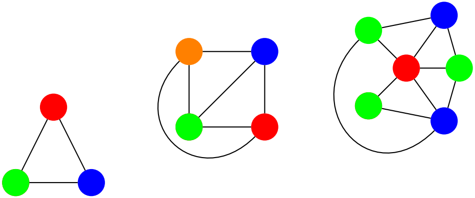
Given $G = (V, E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



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

Graph Coloring.

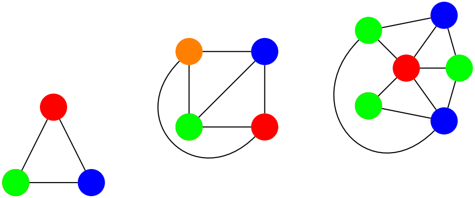
Given $G = (V,E)$, a coloring of a G assigns colors to vertices V where for each edge the endpoints have different colors.



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

Graph Coloring.

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



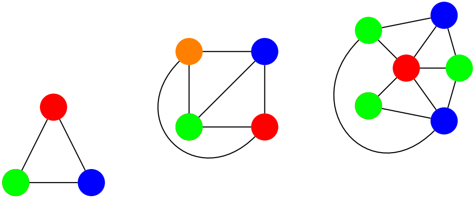
Notice that the last one, has one three colors.

- Fewer colors than number of vertices.
- Fewer colors than max degree node.

Interesting things to do.

Graph Coloring.

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

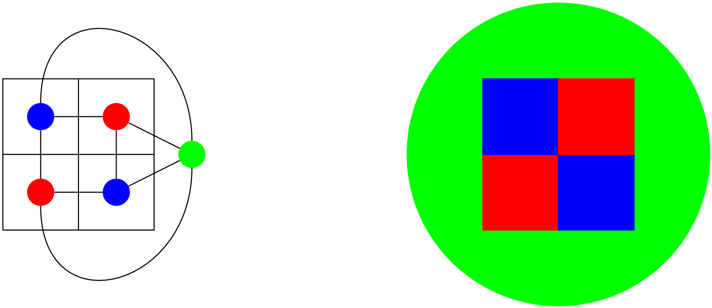


Notice that the last one, has one three colors.
  Fewer colors than number of vertices.
  Fewer colors than max degree node.

Interesting things to do. Algorithm!

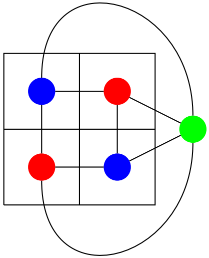
Planar graphs and maps.

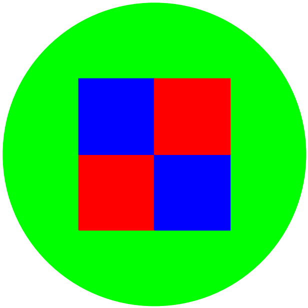
Planar graph coloring $\equiv$ map coloring.



Planar graphs and maps.

Planar graph coloring $\equiv$ map coloring.





Four color theorem is about planar graphs!

Six color theorem.

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

Six color theorem.

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

Proof:

Six color theorem.

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

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

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.

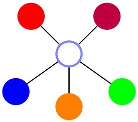
Proof: Again with the degree 5 vertex. Again recurse.

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.



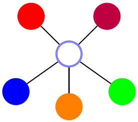
Either switch green.

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.



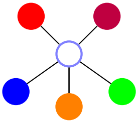
Either switch green.
Or try switching orange.

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.

Four Color Theorem

Four Color Theorem

Theorem: Any planar graph can be colored with four colors.

Four Color Theorem

Theorem: Any planar graph can be colored with four colors.

Proof:

Four Color Theorem

Theorem: Any planar graph can be colored with four colors.

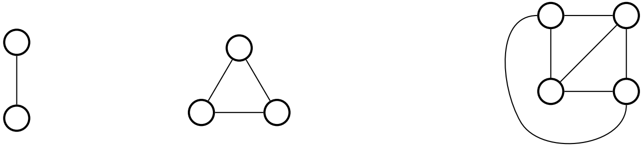
Proof: Not Today!

Four Color Theorem

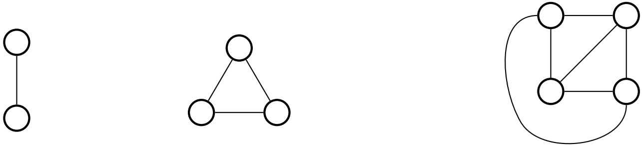
Theorem: Any planar graph can be colored with four colors.

Proof: Not Today!

Graph Types: Complete Graph.

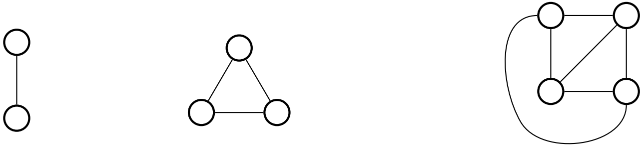


Graph Types: Complete Graph.



K_n, $|V| = n$

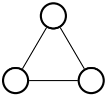
Graph Types: Complete Graph.

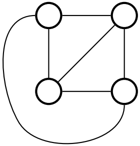


K_n, $|V| = n$

every edge present.

Graph Types: Complete Graph.

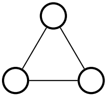
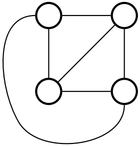




K_n, $|V| = n$

every edge present.
degree of vertex?

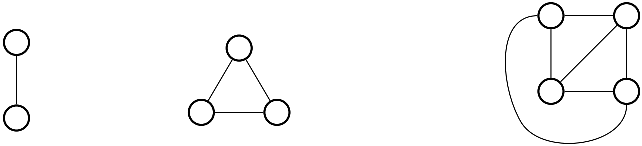
Graph Types: Complete Graph.

K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Graph Types: Complete Graph.

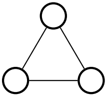


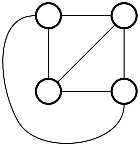
K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Very connected.

Graph Types: Complete Graph.



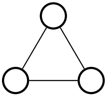
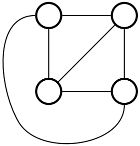


K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

Very connected.
Lots of edges:

Graph Types: Complete Graph.

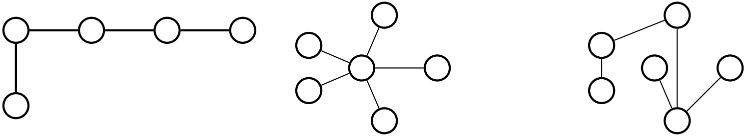




K_n, $|V| = n$

every edge present.
degree of vertex? $|V| - 1$.

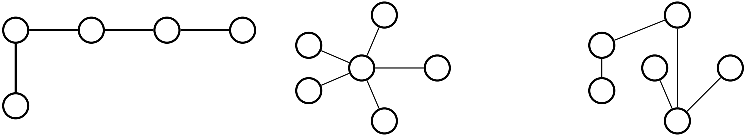
Very connected.
Lots of edges: $n(n-1)/2$.

Trees.



Definitions:

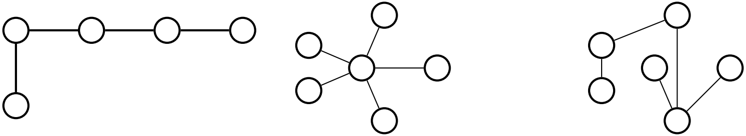
Trees.



Definitions:

A connected graph without a cycle.

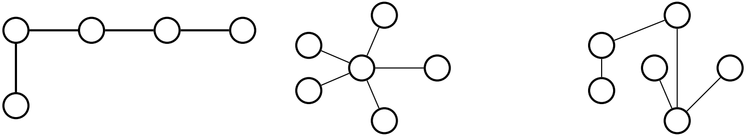
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.

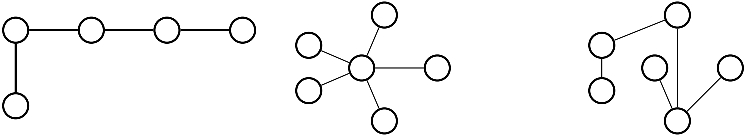
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V|-1$ edges.
A connected graph where any edge removal disconnects it.

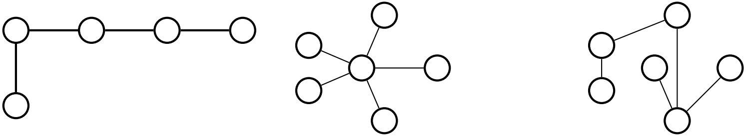
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

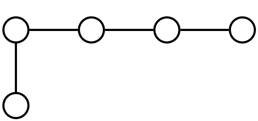
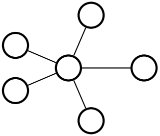
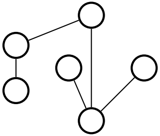
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

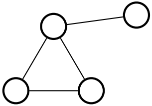
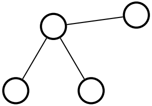
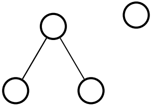
Trees.

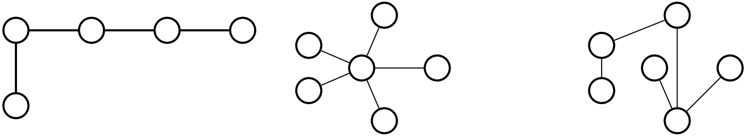
Definitions:

A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!

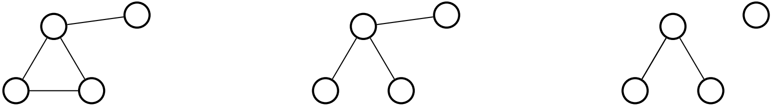
Trees.



Definitions:

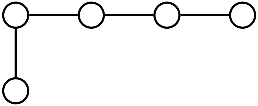
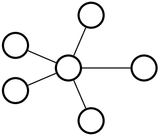
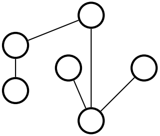
A connected graph without a cycle.
A connected graph with $|V| - 1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



Minimally connected, minimum number of edges to connect.

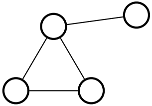
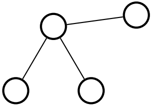
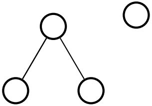
Trees.

Definitions:

A connected graph without a cycle.
A connected graph with $|V|-1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

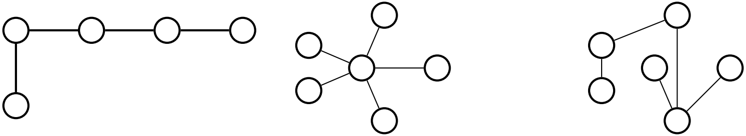
To tree or not to tree!

Minimally connected, minimum number of edges to connect.

Property:

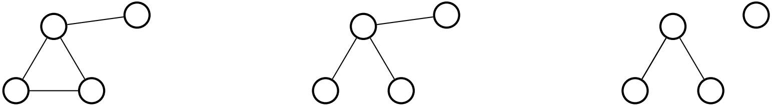
Trees.



Definitions:

A connected graph without a cycle.
A connected graph with $|V|-1$ edges.
A connected graph where any edge removal disconnects it.
An acyclic graph where any edge addition creates a cycle.

To tree or not to tree!



Minimally connected, minimum number of edges to connect.

Property:
Can remove a single node and break into components of size at most $|V|/2$.

Hypercube

Hypercubes.

Hypercube

Hypercubes. Really connected.

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$

Hypercube

Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0, 1\}^n$,

Hypercube

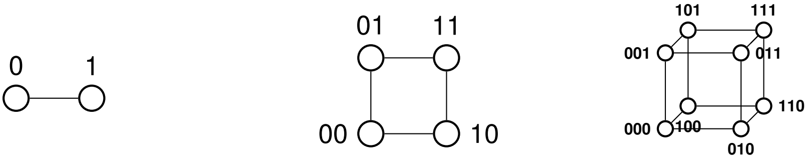
Hypercubes. Really connected. $|V| \log |V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0, 1\}^n$,
$|E| = \{(x, y) | x$ and y differ in one bit position.$\}$

Hypercube

Hypercubes. Really connected. $|V|\log|V|$ edges!
Also represents bit-strings nicely.

$G = (V, E)$
$|V| = \{0,1\}^n$,
$|E| = \{(x,y) | x$ and y differ in one bit position.$\}$



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

Recursive Definition.

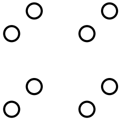
A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.

Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

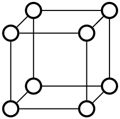
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

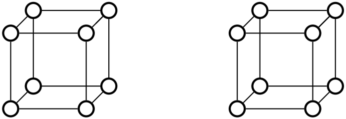
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

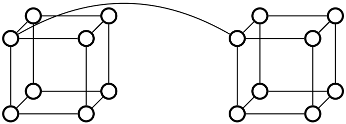
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

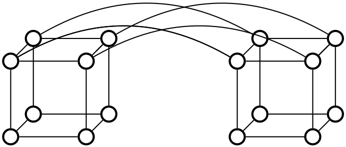
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

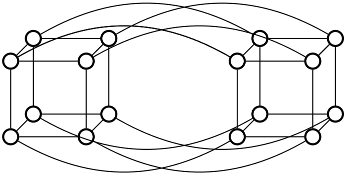
An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Recursive Definition.

A 0-dimensional hypercube is a node labelled with the empty string of bits.

An n-dimensional hypercube consists of a 0-subcube (1-subcube) which is a $n-1$-dimensional hypercube with nodes labelled $0x$ ($1x$) with the additional edges $(0x, 1x)$.



Hypercube:properties

Rudrata Cycle: cycle that visits every node.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian?

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut?

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
 Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
 Best cut? Cut apart subcubes:

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI:

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
000100 $\rightarrow$ 100100 $\rightarrow$ 101100 $\rightarrow$ 101000

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
000100 $\rightarrow$ 100100 $\rightarrow$ 101100 $\rightarrow$ 101000
Correct bits in string, moves along path in hypercube!

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
$000100 \rightarrow 100100 \rightarrow 101100 \rightarrow 101000$
Correct bits in string, moves along path in hypercube!

Hypercube:properties

Rudrata Cycle: cycle that visits every node.
Eulerian? If n is even.

Large Cuts: Cutting off k nodes needs $\geq k$ edges.
Best cut? Cut apart subcubes: cuts off 2^n nodes with 2^{n-1} edges.

FYI: Also cuts represent boolean functions.

Nice Paths between nodes.
Get from 000100 to 101000.
$000100 \to 100100 \to 101100 \to 101000$
Correct bits in string, moves along path in hypercube!

Good communication network!

...Modular Arithmetic...

Arithmetic modulo m.
 Elements of equivalence classes of integers.

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0,\ldots,m-1\}$
    and integer $i \equiv a \pmod{m}$

...Modular Arithmetic...

Arithmetic modulo m.

Elements of equivalence classes of integers.

$\{0, \ldots, m-1\}$

and integer $i \equiv a \pmod{m}$

if $i = a + km$ for integer k.

...Modular Arithmetic...

Arithmetic modulo m.
  Elements of equivalence classes of integers.
    $\{0, \ldots, m-1\}$
    and integer $i \equiv a \pmod{m}$
      if $i = a + km$ for integer k.
       or if the remainder of i divided by m is a.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle

...Modular Arithmetic...

Arithmetic modulo m.
 Elements of equivalence classes of integers.
 $\{0, \ldots, m-1\}$
 and integer $i \equiv a \pmod{m}$
 if $i = a + km$ for integer k.
 or if the remainder of i divided by m is a.

Can do calculations by taking remainders
 at the beginning,
 in the middle
 or at the end.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58 + 32 = 90 = 6 \pmod{7}$
$58 + 32 = 2 + 4 = 6 \pmod{7}$
$58 + 32 = 2 + -3 = -1 = 6 \pmod{7}$

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0,\ldots,m-1\}$
and integer $i \equiv a \pmod{m}$
if $i = a + km$ for integer k.
or if the remainder of i divided by m is a.

Can do calculations by taking remainders
at the beginning,
in the middle
or at the end.

$58+32=90=6 \pmod{7}$
$58+32=2+4=6 \pmod{7}$
$58+32=2+-3=-1=6 \pmod{7}$

Negative numbers work the way you are used to.

...Modular Arithmetic...

Arithmetic modulo m.
Elements of equivalence classes of integers.
$\{0, \ldots, m-1\}$
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 Inverses and GCD

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

Finding gcd.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm!

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case?

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y)

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

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

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax+by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse!

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm$

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod m$.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod m$.

Idea: egcd.
gcd produces 1

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.
gcd produces 1
by adding and subtracting multiples of x and y

Modular Arithmetic Inverses and GCD

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

Group structures more generally.

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

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

Give recursive Algorithm! Base Case? $gcd(x,0) = x$.

Extended-gcd(x,y) returns (d,a,b)
$d = gcd(x,y)$ and $d = ax + by$

Multiplicative inverse of (x,m).
egcd$(x,m) = (1,a,b)$
a is inverse! $1 = ax + bm = ax \pmod{m}$.

Idea: egcd.
gcd produces 1
by adding and subtracting multiples of x and y

Example: $p = 7$, $q = 11$.

Example: $p = 7$, $q = 11$.

$N = 77$.

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7, 60) = 1$.

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$7(0) + 60(1) = 60$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7, 60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \\ 7(-8) + 60(1) &= 4 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \\ 7(-8) + 60(1) &= 4 \\ 7(9) + 60(-1) &= 3 \end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned}
7(0)+60(1) &= 60 \\
7(1)+60(0) &= 7 \\
7(-8)+60(1) &= 4 \\
7(9)+60(-1) &= 3 \\
7(-17)+60(2) &= 1
\end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}$$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}$$

Confirm:

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}
$$

Confirm: $-119 + 120 = 1$

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}$$

Confirm: $-119 + 120 = 1$

$d = e^{-1} = -17 = 43 = \pmod{60}$

Midterm format

Time: 120 minutes.

Midterm format

Time: 120 minutes.

Some short answers.

Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.

Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
Know material well:

Midterm format

Time: 120 minutes.

Some short answers.

Get at ideas that you learned.

Know material well: fast,

Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
Know material well: fast, correct.

Midterm format

Time: 120 minutes.

Some short answers.

Get at ideas that you learned.

Know material well: fast, correct.

Know material medium:

Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.
Know material well: fast, correct.
Know material medium: slower,

Midterm format

Time: 120 minutes.

Some short answers.
Get at ideas that you learned.

Know material well:	fast, correct.
Know material medium:	slower, less correct.

Midterm format

Time: 120 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: 120 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: 120 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: 120 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: 120 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: 120 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: 120 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: 120 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: 120 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: 120 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....
admin@eecs70.org

Wrapup.

Other issues....

admin@eecs70.org

Private message on piazza.

Wrapup.

Other issues....

admin@eecs70.org

Private message on piazza.

Wrapup.

Other issues....

admin@eecs70.org

Private message on piazza.

Good (sort of last minute) Studying!

Wrapup.

Other issues....
admin@eecs70.org
Private message on piazza.

Good (sort of last minute)
Studying!!

Wrapup.

Other issues....
admin@eecs70.org
Private message on piazza.

Good (sort of last minute)
Studying!!!

Wrapup.

Other issues....

admin@eecs70.org

Private message on piazza.

Good (sort of last minute)

Studying!!!!

Wrapup.

Other issues....
admin@eecs70.org
Private message on piazza.

Good (sort of last minute)
Studying!!!!!

Wrapup.

Other issues....
  admin@eecs70.org
  Private message on piazza.

Good (sort of last minute)
Studying!!!!!!

Wrapup.

Other issues....
  admin@eecs70.org
  Private message on piazza.

Good (sort of last minute) Studying!!!!!!!

Wrapup.

Other issues....
admin@eecs70.org
Private message on piazza.

Good (sort of last minute)
Studying!!!!!!!!

Wrapup.

Other issues....

admin@eecs70.org

Private message on piazza.

Good (sort of last minute)

Studying!!!!!!!!!

Wrapup.

Other issues....
admin@eecs70.org
Private message on piazza.

Good (sort of last minute)
Studying!!!!!!!!!!

Wrapup.

Other issues....

admin@eecs70.org

Private message on piazza.

Good (sort of last minute)

Studying!!!!!!!!!!!!

Wrapup.

Other issues....
  admin@eecs70.org
  Private message on piazza.

Good (sort of last minute)
Studying!!!!!!!!!!!!

Wrapup.

Other issues....
  admin@eecs70.org
  Private message on piazza.

Good (sort of last minute)
Studying!!!!!!!!!!!!

Wrapup.

Other issues....
  admin@eecs70.org
  Private message on piazza.

Good (sort of last minute) Studying!!!!!!!!!!!!!

Wrapup.

Other issues....
admin@eecs70.org
Private message on piazza.

Good (sort of last minute)
Studying!!!!!!!!!!!!!!!

Wrapup.

Other issues....

admin@eecs70.org

Private message on piazza.

Good (sort of last minute)

Studying!!!!!!!!!!!!!!!!

Wrapup.

Other issues....
admin@eecs70.org
Private message on piazza.

Good (sort of last minute)
Studying!!!!!!!!!!!!!!!!!!