Today

Finish Euclid.

Bijection/CRT/Isomorphism.

Fermat's Little Theorem.
Today

Finish Euclid.
Today

Finish Euclid.

Bijection/CRT/Isomorphism.
Today

Finish Euclid.

Bijection/CRT/Isomorphism.

Fermat’s Little Theorem.
More divisibility

**Notation:** $d|x$ means “$d$ divides $x$” or
More divisibility

**Notation:** \( d \mid x \) means “\( d \) divides \( x \)” or \( x = kd \) for some integer \( k \).
More divisibility

**Notation:** $d | x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d | x$ and $d | y$ then $d | y$ and $d | \text{mod} (x, y)$.
More divisibility

**Notation:** $d \mid x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d \mid x$ and $d \mid y$ then $d \mid y$ and $d \mid \text{mod}(x, y)$.

**Proof:**
\[
\text{mod}(x, y) = x - \lfloor x/y \rfloor \cdot y
\]
More divisibility

**Notation:** $d|x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d|x$ and $d|y$ then $d|y$ and $d|\text{mod}(x, y)$.

**Proof:**
\[
\text{mod}(x, y) = x - \lfloor x/y \rfloor \cdot y \\
= x - \lfloor s \rfloor \cdot y \quad \text{for integer } s
\]
More divisibility

**Notation:** $d|x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d|x$ and $d|y$ then $d|y$ and $d|\text{mod}(x, y)$.

**Proof:**
\[
\text{mod}(x, y) = x - \lfloor x/y \rfloor \cdot y \\
= x - \lfloor s \rfloor \cdot y \quad \text{for integer } s \\
= kd - sld \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = ld
\]
More divisibility

**Notation:** $d|x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d|x$ and $d|y$ then $d|y$ and $d|\text{mod}(x, y)$.

**Proof:**

\[
\text{mod}(x, y) = x - \lfloor x/y \rfloor \cdot y = x - \lfloor s \rfloor \cdot y \quad \text{for integer } s
\]

\[
= kd - s\ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d
\]

\[
= (k - s\ell)d
\]
More divisibility

**Notation:** \( d \mid x \) means “\( d \) divides \( x \)” or \( x = kd \) for some integer \( k \).

**Lemma 1:** If \( d \mid x \) and \( d \mid y \) then \( d \mid y \) and \( d \mid \text{mod}(x, y) \).

**Proof:**
\[
\text{mod}(x, y) = x - \lfloor x/y \rfloor \cdot y \\
= x - \lfloor s \rfloor \cdot y \quad \text{for integer } s \\
= kd - s \ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d \\
= (k - s \ell)d
\]
Therefore \( d \mid \text{mod}(x, y) \).
More divisibility

**Notation:** $d | x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d | x$ and $d | y$ then $d | y$ and $d | \mod (x, y)$.

**Proof:**
\[
\mod (x, y) = x - \lfloor x/y \rfloor \cdot y = x - s \cdot y \quad \text{for integer } s = kd - s \ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d = (k - s \ell) d
\]

Therefore $d | \mod (x, y)$. And $d | y$ since it is in condition.
More divisibility

**Notation:** \(d | x\) means “\(d\) divides \(x\)” or \(x = kd\) for some integer \(k\).

**Lemma 1:** If \(d | x\) and \(d | y\) then \(d | y\) and \(d \mid \text{mod} (x, y)\).

**Proof:**
\[
\text{mod} (x, y) = x - \lfloor x/y \rfloor \cdot y = x - \lfloor s \rfloor \cdot y \quad \text{for integer } s
\]
\[
= kd - s\ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d
\]
\[
= (k - s\ell)d
\]

Therefore \(d \mid \text{mod} (x, y)\). And \(d | y\) since it is in condition.

\(\square\)
More divisibility

**Notation:** $d|x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d|x$ and $d|y$ then $d|y$ and $d|\text{mod}(x,y)$.

**Proof:**
\[
\begin{align*}
\text{mod} (x,y) &= x - \lfloor x/y \rfloor \cdot y \\
&= x - \lfloor s \rfloor \cdot y \quad \text{for integer } s \\
&= kd - s\ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d \\
&= (k - s\ell)d
\end{align*}
\]
Therefore $d|\text{mod} (x,y)$. And $d|y$ since it is in condition.

**Lemma 2:** If $d|y$ and $d|\text{mod}(x,y)$ then $d|y$ and $d|x$.

**Proof...:** Similar.

---

GCD Mod Corollary:

\[\gcd(x,y) = \gcd(y,\text{mod}(x,y))\]

**Proof:** $x$ and $y$ have the same set of common divisors as $x$ and $\text{mod}(x,y)$ by Lemma 1 and 2. The same common divisors $\implies$ largest is the same.
More divisibility

**Notation:** $d | x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d | x$ and $d | y$ then $d | y$ and $d | \text{mod} (x, y)$.

**Proof:**
\[
\text{mod} (x, y) = x - \lfloor x/y \rfloor \cdot y \\
= x - \lfloor s \rfloor \cdot y \quad \text{for integer } s \\
= kd - s \ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d \\
= (k - s \ell )d
\]

Therefore $d | \text{mod} (x, y)$. And $d | y$ since it is in condition.

**Lemma 2:** If $d | y$ and $d | \text{mod} (x, y)$ then $d | y$ and $d | x$.

**Proof...:** Similar. Try this at home.
More divisibility

**Notation:** $d | x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d | x$ and $d | y$ then $d | y$ and $d | \text{mod}(x, y)$.  

**Proof:**

$$\text{mod}(x, y) = x - \lfloor x/y \rfloor \cdot y$$

$$= x - \lfloor s \rfloor \cdot y \quad \text{for integer } s$$

$$= kd - s\ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d$$

$$= (k - s\ell)d$$

Therefore $d | \text{mod}(x, y)$. And $d | y$ since it is in condition.  

**Lemma 2:** If $d | y$ and $d | \text{mod}(x, y)$ then $d | y$ and $d | x$.

**Proof...:** Similar. Try this at home.

ish.
More divisibility

**Notation:** \( d|\ x \) means “\( d \) divides \( x \)” or \( x = kd \) for some integer \( k \).

**Lemma 1:** If \( d|\ x \) and \( d|\ y \) then \( d|\ y \) and \( d|\ \text{mod}\ (x, y) \).

**Proof:**

\[
\text{mod}\ (x, y) = x - \left\lfloor \frac{x}{y} \right\rfloor \cdot y \\
= x - \lfloor s \rfloor \cdot y \quad \text{for integer } s \\
= kd - s\ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d \\
= (k - s\ell)d
\]

Therefore \( d|\ \text{mod}\ (x, y) \). And \( d|\ y \) since it is in condition. \( \square \)

**Lemma 2:** If \( d|\ y \) and \( d|\ \text{mod}\ (x, y) \) then \( d|\ y \) and \( d|\ x \).

**Proof...:** Similar. Try this at home. \( \square \text{ish.} \)

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \text{ mod } (x, y)) \).
More divisibility

**Notation:** $d | x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d | x$ and $d | y$ then $d | y$ and $d | \text{mod} (x, y)$.

**Proof:**
\[
\text{mod} (x, y) = x - \lfloor x/y \rfloor \cdot y \\
= x - [s] \cdot y \quad \text{for integer } s \\
= kd - s\ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d \\
= (k - s\ell)d
\]

Therefore $d | \text{mod} (x, y)$. And $d | y$ since it is in condition.

**Lemma 2:** If $d | y$ and $d | \text{mod} (x, y)$ then $d | y$ and $d | x$.

**Proof...:** Similar. Try this at home.

**GCD Mod Corollary:** $\gcd(x, y) = \gcd(y, \text{mod} (x, y))$.

**Proof:** $x$ and $y$ have **same** set of common divisors as $x$ and mod $(x, y)$ by Lemma 1 and 2.
More divisibility

**Notation:** $d|x$ means “$d$ divides $x$” or $x = kd$ for some integer $k$.

**Lemma 1:** If $d|x$ and $d|y$ then $d|y$ and $d|\text{mod} \ (x, y)$.

**Proof:**
\[
\begin{align*}
\text{mod} \ (x, y) &= x - \lfloor x/y \rfloor \cdot y \\
&= x - \lfloor s \rfloor \cdot y \\
&= kd - sld \\
&= (k - sl)d
\end{align*}
\]
Therefore $d|\text{mod} \ (x, y)$. And $d|y$ since it is in condition.

**Lemma 2:** If $d|y$ and $d|\text{mod} \ (x, y)$ then $d|y$ and $d|x$.

**Proof...:** Similar. Try this at home.

**GCD Mod Corollary:** $\gcd(x, y) = \gcd(y, \ \text{mod} \ (x, y))$.

**Proof:** $x$ and $y$ have **same** set of common divisors as $x$ and $\text{mod} \ (x, y)$ by Lemma 1 and 2.

Same common divisors $\implies$ largest is the same.
More divisibility

**Notation:** \( d \mid x \) means “\( d \) divides \( x \)” or \( x = kd \) for some integer \( k \).

**Lemma 1:** If \( d \mid x \) and \( d \mid y \) then \( d \mid y \) and \( d \mid \text{mod} \ (x, y) \).

**Proof:**
\[
\text{mod} \ (x, y) = x - \lfloor x/y \rfloor \cdot y \\
= x - s \cdot y \quad \text{for integer } s \\
= kd - s\ell d \quad \text{for integers } k, \ell \text{ where } x = kd \text{ and } y = \ell d \\
= (k - s\ell)d
\]
Therefore \( d \mid \text{mod} \ (x, y) \). And \( d \mid y \) since it is in condition.

**Lemma 2:** If \( d \mid y \) and \( d \mid \text{mod} \ (x, y) \) then \( d \mid y \) and \( d \mid x \).

**Proof...:** Similar. Try this at home.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \text{mod} \ (x, y)) \).

**Proof:** \( x \) and \( y \) have same set of common divisors as \( x \) and \( \text{mod} \ (x, y) \) by Lemma 1 and 2.

Same common divisors \( \implies \) largest is the same.
Euclid’s algorithm.

GCD Mod Corollary: \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).
Euclid’s algorithm.

**GCD Mod Corollary:** $\gcd(x, y) = \gcd(y, \mod(x, y))$.

Hey, what’s $\gcd(7, 0)$?
Euclid’s algorithm.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0) \)? 7
Euclid’s algorithm.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
Euclid’s algorithm.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \gcd(x, 0) \)?
Euclid’s algorithm.

**GCD Mod Corollary:** \( \text{gcd}(x, y) = \text{gcd}(y, \mod(x, y)) \).

Hey, what’s \( \text{gcd}(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \text{gcd}(x, 0) \)? \( x \)
Euclid’s algorithm.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \gcd(x, 0) \)?  

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

Theorem: \( \text{(euclid } x y \text{)} = \gcd(x, y) \) if \( x \geq y \).

Proof: Use Strong Induction.
Base Case: \( y = 0 \), "x divides y and x is common divisor and clearly largest."
Induction Step: \( \mod(x, y) < y \leq x \) when \( x \geq y \) call in line (*** meets conditions plus arguments "smaller" and by strong induction hypothesis computes \( \gcd(y, \mod(x, y)) \) which is \( \gcd(x, y) \) by GCD Mod Corollary.
Euclid’s algorithm.

**GCD Mod Corollary:** $\gcd(x, y) = \gcd(y, \mod(x, y))$.

Hey, what’s $\gcd(7, 0)$? 7 since 7 divides 7 and 7 divides 0
What’s $\gcd(x, 0)$? x

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

**Theorem:** $(\text{euclid } x y) = \gcd(x, y)$ if $x \geq y$. 

Euclid’s algorithm.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0) \)? 7 since 7 divides 7 and 7 divides 0

What’s \( \gcd(x, 0) \)? \( x \)

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

**Theorem:** \( (\text{euclid } x \ y) = \gcd(x, y) \) if \( x \geq y \).

**Proof:** Use Strong Induction.
Euclid’s algorithm.

**GCD Mod Corollary:** \( \text{gcd}(x, y) = \text{gcd}(y, \mod(x, y)) \).

Hey, what’s \( \text{gcd}(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \text{gcd}(x, 0) \)? \( x \)

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

**Theorem:** \( (\text{euclid } x \ y) = \text{gcd}(x, y) \) if \( x \geq y \).

**Proof:** Use Strong Induction.
**Base Case:** \( y = 0 \), “\( x \) divides \( y \) and \( x \)”
Euclid’s algorithm.

**GCD Mod Corollary:** \( \text{gcd}(x, y) = \text{gcd}(y, \mod (x, y)) \).

Hey, what’s \( \text{gcd}(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \text{gcd}(x, 0) \)? \( x \)

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

**Theorem:** \( (\text{euclid} \ x \ y) = \text{gcd}(x, y) \) if \( x \geq y \).

**Proof:** Use Strong Induction.

**Base Case:** \( y = 0 \), “\( x \) divides \( y \) and \( x \)”
 \( \implies \) “\( x \) is common divisor and clearly largest.”
Euclid’s algorithm.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \gcd(x, 0) \)? \( x \)

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

**Theorem:** \( (\text{euclid } x \ y) = \gcd(x, y) \) if \( x \geq y \).

**Proof:** Use Strong Induction.

**Base Case:** \( y = 0 \), “\( x \) divides \( y \) and \( x \)”
\[ \implies \text{ “} x \text{ is common divisor and clearly largest.”} \]

**Induction Step:** \( \mod(x, y) < y \leq x \) when \( x \geq y \)
Euclid’s algorithm.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \gcd(x, 0) \)? \( x \)

\[
\text{(define (euclid x y)} \n  \text{(if (= y 0)} \n    x \n    (euclid y (mod x y)))) \) ***
\]

**Theorem:** \( (\text{euclid } x \ y) = \gcd(x, y) \) if \( x \geq y \).

**Proof:** Use Strong Induction.

**Base Case:** \( y = 0 \), “\( x \) divides \( y \) and \( x \)”
\( \implies \) “\( x \) is common divisor and clearly largest.”

**Induction Step:** \( \mod(x, y) < y \leq x \) when \( x \geq y \)

call in line (***') meets conditions plus arguments “smaller”
Euclid’s algorithm.

GCD Mod Corollary: \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0)? \) 7 since 7 divides 7 and 7 divides 0
What’s \( \gcd(x, 0)? \)  \( \text{x} \)

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

Theorem: \( (\text{euclid } x \ y) = \gcd(x, y) \) if \( x \geq y \).

Proof: Use Strong Induction.
Base Case: \( y = 0, \text{ “x divides y and x”} \)
  \( \implies \text{ “x is common divisor and clearly largest.”} \)
Induction Step: \( \mod(x, y) < y \leq x \) when \( x \geq y \)

call in line (*** ) meets conditions plus arguments “smaller”
  and by strong induction hypothesis
Euclid’s algorithm.

**GCD Mod Corollary:** \( \gcd(x, y) = \gcd(y, \mod(x, y)) \).

Hey, what’s \( \gcd(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \gcd(x, 0) \)? \( x \)

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

**Theorem:** \( (euclid x y) = \gcd(x, y) \) if \( x \geq y \).

**Proof:** Use Strong Induction.

**Base Case:** \( y = 0 \), “\( x \) divides \( y \) and \( x \)”

\[ \implies \text{“}\( x \) is common divisor and clearly largest.” \]

**Induction Step:** \( \mod(x, y) < y \leq x \) when \( x \geq y \)

call in line (***), meets conditions plus arguments “smaller”
and by strong induction hypothesis
computes \( \gcd(y, \mod(x, y)) \)
Euclid’s algorithm.

**GCD Mod Corollary:** \( \text{gcd}(x, y) = \text{gcd}(y, \mod(x, y)) \).

Hey, what’s \( \text{gcd}(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \text{gcd}(x, 0) \)? \( x \)

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

**Theorem:** \( \text{(euclid x y)} = \text{gcd}(x, y) \) if \( x \geq y \).

**Proof:** Use Strong Induction.

**Base Case:** \( y = 0 \), “\( x \) divides \( y \) and \( x \)”

\[ \implies \] “\( x \) is common divisor and clearly largest.”

**Induction Step:** \( \mod(x, y) < y \leq x \) when \( x \geq y \)

call in line (***)) meets conditions plus arguments “smaller”
and by strong induction hypothesis
computes \( \text{gcd}(y, \mod(x, y)) \)
which is \( \text{gcd}(x, y) \) by GCD Mod Corollary.
Euclid’s algorithm.

**GCD Mod Corollary:** \( \text{gcd}(x, y) = \text{gcd}(y, \mod(x, y)) \).

Hey, what’s \( \text{gcd}(7, 0) \)? 7 since 7 divides 7 and 7 divides 0
What’s \( \text{gcd}(x, 0) \)?  \( x \)

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

**Theorem:** \( (\text{euclid } x \ y) = \text{gcd}(x, y) \) if \( x \geq y \).

**Proof:** Use Strong Induction.

**Base Case:** \( y = 0 \), “\( x \) divides \( y \) and \( x \)”

\( \implies \) “\( x \) is common divisor and clearly largest.”

**Induction Step:** \( \mod(x, y) < y \leq x \) when \( x \geq y \)

call in line (*** ) meets conditions plus arguments “smaller”
and by strong induction hypothesis
computes \( \text{gcd}(y, \mod(x, y)) \)
which is \( \text{gcd}(x, y) \) by GCD Mod Corollary.
Excursion: Value and Size.

Before discussing running time of gcd procedure...
Excursion: Value and Size.

Before discussing running time of gcd procedure...
What is the value of 1,000,000?
Excursion: Value and Size.

Before discussing running time of gcd procedure...

What is the value of 1,000,000?

One million or 1,000,000!
Excursion: Value and Size.

Before discussing running time of gcd procedure...

What is the value of 1,000,000?

one million or 1,000,000!

What is the “size” of 1,000,000?
Excursion: Value and Size.

Before discussing running time of gcd procedure...

What is the value of 1,000,000?
one million or 1,000,000!

What is the “size” of 1,000,000?
Number of digits in base 10: 7.
Excursion: Value and Size.

Before discussing running time of gcd procedure...

What is the value of 1,000,000?
one million or 1,000,000!

What is the “size” of 1,000,000?
Number of digits in base 10: 7.
Number of bits (a digit in base 2): 21.
Excursion: Value and Size.

Before discussing running time of gcd procedure...
What is the value of 1,000,000?
one million or 1,000,000!
What is the “size” of 1,000,000?
Number of digits in base 10: 7.
Number of bits (a digit in base 2): 21.
For a number $x$, what is its size in bits?
Excursion: Value and Size.

Before discussing running time of gcd procedure...
What is the value of 1,000,000?
one million or 1,000,000!
What is the “size” of 1,000,000?
Number of digits in base 10: 7.
Number of bits (a digit in base 2): 21.
For a number $x$, what is its size in bits?

$$n = b(x) \approx \log_2 x$$
Excursion: Value and Size.

Before discussing running time of gcd procedure...

What is the value of 1,000,000?
one million or 1,000,000!

What is the “size” of 1,000,000?
Number of digits in base 10: 7.
Number of bits (a digit in base 2): 21.

For a number \( x \), what is its size in bits?

\[ n = b(x) \approx \log_2 x \]
Euclid procedure is fast.

**Theorem:** (euclid $x$ $y$) uses $2n$ ”divisions” where $n = b(x) \approx \log_2 x$. 

---

$2^n$ is much faster! roughly 200 divisions.

$2^{100} \approx 10^{30} = \text{"million, trillion, trillion" divisions!}$
Euclid procedure is fast.

**Theorem:** \((\text{euclid } x \ y)\) uses \(2n\) ”divisions” where \(n = b(x) \approx \log_2 x\).

Is this good?
Euclid procedure is fast.

**Theorem:** $(\text{euclid } x \ y)$ uses $2n$ "divisions" where $n = b(x) \approx \log_2 x$.

Is this good? Better than trying all numbers in $\{2, \ldots y/2\}$?
Euclid procedure is fast.

**Theorem:** \((\text{euclid } x \ y)\) uses \(2n\) "divisions" where \(n = b(x) \approx \log_2 x\).

Is this good? Better than trying all numbers in \(\{2, \ldots y/2\}\)?

Check 2,
Euclid procedure is fast.

**Theorem:** $(\text{euclid } x \ y)$ uses $2n$ "divisions" where $n = b(x) \approx \log_2 x$. Is this good? Better than trying all numbers in $\{2, \ldots, y/2\}$?

Check 2, check 3,
Euclid procedure is fast.

**Theorem:** \((\text{euclid } x \ y)\) uses \(2n\) "divisions" where \(n = b(x) \approx \log_2 x\).

Is this good? Better than trying all numbers in \(\{2, \ldots y/2\}\)?

Check 2, check 3, check 4,
Euclid procedure is fast.

**Theorem:** \((\text{euclid } x \ y)\) uses \(2n\) ”divisions” where \(n = b(x) \approx \log_2 x\).

Is this good? Better than trying all numbers in \(\{2, \ldots, y/2\}\)?

Check 2, check 3, check 4, check 5 . . . , check \(y/2\).
Euclid procedure is fast.

**Theorem:** $(\text{euclid } x y)$ uses $2n$ ”divisions” where $n = b(x) \approx \log_2 x$.

Is this good? Better than trying all numbers in $\{2, \ldots, y/2\}$?

Check 2, check 3, check 4, check 5 . . . , check $y/2$. 

$2^n$ is much faster!

... roughly $2^{100} \approx 10^{30} =$ "million, trillion, trillion" divisions!
Euclid procedure is fast.

**Theorem:** \((\text{euclid } x \ y)\) uses \(2n\) ”divisions” where \(n = b(x) \approx \log_2 x\).

Is this good? Better than trying all numbers in \(\{2, \ldots y/2\}\)?

Check 2, check 3, check 4, check 5 \ldots, check \(y/2\).

If \(y \approx x\)
Euclid procedure is fast.

**Theorem:** $(\text{euclid } x \ y)$ uses $2n$ ”divisions” where $n = b(x) \approx \log_2 x$.

Is this good? Better than trying all numbers in $\{2, \ldots, y/2\}$?

Check 2, check 3, check 4, check 5 . . . , check $y/2$.

If $y \approx x$ roughly $y$ uses $n$ bits
Euclid procedure is fast.

**Theorem:** \((\text{euclid } x \ y)\) uses \(2n\) ”divisions” where \(n = b(x) \approx \log_2 x\).

Is this good? Better than trying all numbers in \(\{2, \ldots y/2\}\)?

Check 2, check 3, check 4, check 5 . . . , check \(y/2\).

If \(y \approx x\) roughly \(y\) uses \(n\) bits ...

\(2^{n-1}\) divisions! Exponential dependence on size!
Euclid procedure is fast.

**Theorem:** \((\text{euclid } x \ y)\) uses \(2n\) ”divisions” where \(n = b(x) \approx \log_2 x\).

Is this good? Better than trying all numbers in \(\{2, \ldots y/2\}\)?

Check 2, check 3, check 4, check 5 . . . , check \(y/2\).

If \(y \approx x\) roughly \(y\) uses \(n\) bits ...

\[2^{n-1}\] divisions! Exponential dependence on size!

101 bit number.
Euclid procedure is fast.

**Theorem:** $(euclid \ x \ y)$ uses $2n$ ”divisions” where $n = b(x) \approx \log_2 x$.

Is this good? Better than trying all numbers in $\{2, \ldots y/2\}$?

Check 2, check 3, check 4, check 5 . . . , check $y/2$.

If $y \approx x$ roughly $y$ uses $n$ bits . . .

$2^{n-1}$ divisions! Exponential dependence on size!

101 bit number. $2^{100} \approx 10^{30} = \text{“million, trillion, trillion” divisions}!}$
Euclid procedure is fast.

**Theorem:** (euclid x y) uses $2n$ ”divisions” where $n = b(x) \approx \log_2 x$.

Is this good? Better than trying all numbers in \{2,\ldots,y/2\}?

Check 2, check 3, check 4, check 5 \ldots , check $y/2$.

If $y \approx x$ roughly $y$ uses $n$ bits ...

$2^{n-1}$ divisions! Exponential dependence on size!

101 bit number. $2^{100} \approx 10^{30} =$ “million, trillion, trillion” divisions!

$2n$ is much faster!
Euclid procedure is fast.

**Theorem:** $(\text{euclid } x \ y)$ uses $2n$ "divisions" where $n = b(x) \approx \log_2 x$.

Is this good? Better than trying all numbers in \{2, \ldots y/2\}?

Check 2, check 3, check 4, check 5 . . . , check $y/2$.

If $y \approx x$ roughly $y$ uses $n$ bits . . .  
$2^{n-1}$ divisions! Exponential dependence on size!

101 bit number. $2^{100} \approx 10^{30} = \text{“million, trillion, trillion” divisions!}$

$2n$ is much faster! . . roughly 200 divisions.
Algorithms at work.

Trying everything

```
(x, y) = euclid(700, 568)
(x, y) = euclid(568, 132)
(x, y) = euclid(132, 40)
(x, y) = euclid(40, 12)
(x, y) = euclid(12, 4)
```

Notice: The first argument decreases rapidly. At least a factor of 2 in two recursive calls. (The second is less than the first.)
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check $y/2$. 
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check $y/2$.
“(gcd x y)” at work.
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check \( y/2 \).
“(gcd x y)” at work.

euclid(700, 568)
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check \(y/2\).
“(gcd x y)” at work.

\[
\text{euclid}(700, 568) \\
\text{euclid}(568, 132)
\]
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check $y/2$.
“(gcd $x$ $y$)” at work.

\[
\begin{align*}
euclid(700, 568) \\
euclid(568, 132) \\
euclid(132, 40)
\end{align*}
\]
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check y/2.
“(gcd x y)” at work.

euclid(700, 568)
euclid(568, 132)
euclid(132, 40)
euclid(40, 12)
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 \ldots, check \( y/2 \).

“(gcd x y)” at work.

\[
\begin{align*}
euclid(700, 568) \\
euclid(568, 132) \\
euclid(132, 40) \\
euclid(40, 12) \\
euclid(12, 4)
\end{align*}
\]
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check \( y/2 \).
“(gcd x y)” at work.

\[
euclid(700, 568) \\
euclid(568, 132) \\
euclid(132, 40) \\
euclid(40, 12) \\
euclid(12, 4) \\
euclid(4, 0)
\]
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check \( y/2 \).
“(gcd x y)” at work.

\[
\text{euclid}(700, 568) \\
\text{euclid}(568, 132) \\
\text{euclid}(132, 40) \\
\text{euclid}(40, 12) \\
\text{euclid}(12, 4) \\
\text{euclid}(4, 0) \\
4
\]
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 \ldots, check \( y/2 \).
“(gcd x y)” at work.

\begin{verbatim}
euclid(700, 568)
euclid(568, 132)
euclid(132, 40)
euclid(40, 12)
euclid(12, 4)
euclid(4, 0)
4
\end{verbatim}

Notice: The first argument decreases rapidly.
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check $y/2$.
“(gcd x y)” at work.

\[
\begin{align*}
euclid(700, 568) \\
euclid(568, 132) \\
euclid(132, 40) \\
euclid(40, 12) \\
euclid(12, 4) \\
euclid(4, 0) \Rightarrow 4
\end{align*}
\]

Notice: The first argument decreases rapidly.
At least a factor of 2 in two recursive calls.
Algorithms at work.

Trying everything
Check 2, check 3, check 4, check 5 . . . , check \( y/2 \).
“(gcd x y)” at work.

\[
\begin{align*}
\text{euclid}(700, 568) \\
\text{euclid}(568, 132) \\
\text{euclid}(132, 40) \\
\text{euclid}(40, 12) \\
\text{euclid}(12, 4) \\
\text{euclid}(4, 0) \\
&= 4
\end{align*}
\]

Notice: The first argument decreases rapidly.
At least a factor of 2 in two recursive calls.
(The second is less than the first.)
Poll.
(define (euclid x y)
  (if (= y 0)
    x
    (euclid y (mod x y))))

**Theorem:** (euclid x y) uses $O(n)$ ”divisions” where $n = b(x)$. 
(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y))))

Theorem: (euclid x y) uses $O(n)$ ”divisions” where $n = b(x)$.

Proof:

Fact:
First arg decreases by at least factor of two in two recursive calls.
(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y))))

**Theorem:** \(\text{(euclid } x \ y)\) uses \(O(n)\) "divisions" where \(n = b(x)\).

**Proof:**

**Fact:**
First arg decreases by at least factor of two in two recursive calls.

After \(2\log_2 x = O(n)\) recursive calls, argument \(x\) is 1 bit number.
(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y))))

**Theorem:** (euclid x y) uses $O(n)$ ”divisions” where $n = b(x)$.

**Proof:**

**Fact:**
First arg decreases by at least factor of two in two recursive calls.

After $2\log_2 x = O(n)$ recursive calls, argument $x$ is 1 bit number.
One more recursive call to finish.
(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y)))))

**Theorem:** (euclid x y) uses $O(n)$ ”divisions” where $n = b(x)$.

**Proof:**

**Fact:**
First arg decreases by at least factor of two in two recursive calls.
After $2\log_2 x = O(n)$ recursive calls, argument $x$ is 1 bit number.
One more recursive call to finish.
1 division per recursive call.
Runtime Proof.

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

**Theorem:** (euclid x y) uses $O(n)$ ”divisions” where $n = b(x)$.

**Proof:**

**Fact:**
First arg decreases by at least factor of two in two recursive calls.

After $2\log_2 x = O(n)$ recursive calls, argument $x$ is 1 bit number.
One more recursive call to finish.
1 division per recursive call.
$O(n)$ divisions.
Runtime Proof (continued.)

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

**Fact:**
First arg decreases by at least factor of two in two recursive calls.
Runtime Proof (continued.)

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

**Fact:**
First arg decreases by at least factor of two in two recursive calls.

**Proof of Fact:** Recall that first argument decreases every call.
Runtime Proof (continued.)

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

Fact:
First arg decreases by at least factor of two in two recursive calls.

Proof of Fact: Recall that first argument decreases every call.

Case 1: $y < x/2$, first argument is $y$
  $\implies$ true in one recursive call;
Runtime Proof (continued.)

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

Fact:
First arg decreases by at least factor of two in two recursive calls.

Proof of Fact: Recall that first argument decreases every call.

Case 1: $y < x/2$, first argument is $y$
  $\implies$ true in one recursive call;
Runtime Proof (continued.)

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

Fact:
First arg decreases by at least factor of two in two recursive calls.

Proof of Fact: Recall that first argument decreases every call.

Case 1: \( y < \frac{x}{2} \), first argument is \( y \)
\[ \implies \text{true in one recursive call;} \]

Case 2: Will show \( y \geq \frac{x}{2} \) \( \implies \) \( \text{mod}(x, y) \leq \frac{x}{2}. \)
Runtime Proof (continued.)

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

Fact:
First arg decreases by at least factor of two in two recursive calls.

Proof of Fact: Recall that first argument decreases every call.
Case 1: \( y < \frac{x}{2} \), first argument is \( y \)
  \( \implies \) true in one recursive call;

Case 2: Will show \( \frac{y}{2} \) \( \implies \) \( \text{mod}(x, y) \leq \frac{x}{2} \).
  \( \text{mod} (x, y) \) is second argument in next recursive call,
Runtime Proof (continued.)

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

Fact:
First arg decreases by at least factor of two in two recursive calls.

Proof of Fact: Recall that first argument decreases every call.

Case 1: \( y < \frac{x}{2} \), first argument is \( y \)
  \( \Rightarrow \) true in one recursive call;

Case 2: Will show \( y \geq \frac{x}{2} \) \( \Rightarrow \) \( \text{mod}(x, y) \leq \frac{x}{2} \).

\( \text{mod} (x, y) \) is second argument in next recursive call, and becomes the first argument in the next one.
(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y))))

Fact:
First arg decreases by at least factor of two in two recursive calls.

Proof of Fact: Recall that first argument decreases every call.

Case 1: \( y < x/2 \), first argument is \( y \)
\[ \implies \text{true in one recursive call;} \]

Case 2: Will show “\( y \geq x/2 \) \( \implies \text{mod}(x, y) \leq x/2. \)”
\[ \text{mod} (x, y) \text{ is second argument in next recursive call,} \]
\[ \text{and becomes the first argument in the next one.} \]
When \( y \geq x/2 \), then
Runtime Proof (continued.)

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

Fact:
First arg decreases by at least factor of two in two recursive calls.

Proof of Fact: Recall that first argument decreases every call.

Case 1: $y < \frac{x}{2}$, first argument is $y$
  $\implies$ true in one recursive call;

Case 2: Will show “$y \geq \frac{x}{2}$” $\implies$ “$\text{mod}(x, y) \leq \frac{x}{2}$.”
  $\text{mod}(x, y)$ is second argument in next recursive call, and becomes the first argument in the next one.
When $y \geq \frac{x}{2}$, then
  \[ \left\lfloor \frac{x}{y} \right\rfloor = 1, \]
(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y))))

**Fact:**
First arg decreases by at least factor of two in two recursive calls.

**Proof of Fact:** Recall that first argument decreases every call.

Case 1: $y < x/2$, first argument is $y$

$$\implies \text{true in one recursive call;}$$

Case 2: Will show “$y \geq x/2$$ \implies “\mod(x,y) \leq x/2.”$$

$\mod(x,y)$ is second argument in next recursive call, and becomes the first argument in the next one.

When $y \geq x/2$, then

$$\lfloor \frac{x}{y} \rfloor = 1,$$

$$\mod(x,y) = x - y \lfloor \frac{x}{y} \rfloor =$$
Runtime Proof (continued.)

\[
\text{(define (euclid x y)}
\text{
  (if (= y 0)
    x
    \text{(euclid y (mod x y)})}))
\]

**Fact:**
First arg decreases by at least factor of two in two recursive calls.

**Proof of Fact:** Recall that first argument decreases every call.

Case 1: \( y < \frac{x}{2} \), first argument is \( y \)

\( \implies \) true in one recursive call;

Case 2: Will show “\( y \geq \frac{x}{2} \)” \( \implies \) “\( \text{mod}(x,y) \leq \frac{x}{2} \).”

\( \text{mod} (x,y) \) is second argument in next recursive call, and becomes the first argument in the next one.

When \( y \geq \frac{x}{2} \), then

\[
\left\lfloor \frac{x}{y} \right\rfloor = 1,
\]

\[
\text{mod} (x,y) = x - y \left\lfloor \frac{x}{y} \right\rfloor = x - y \leq x - \frac{x}{2}
\]
(define (euclid x y)
  (if (= y 0)
      x
      (euclid y (mod x y))))

**Fact:**
First arg decreases by at least factor of two in two recursive calls.

**Proof of Fact:** Recall that first argument decreases every call.

Case 1: \( y < \frac{x}{2} \), first argument is \( y \)
\[ \implies \text{true in one recursive call}; \]

Case 2: Will show \( y \geq \frac{x}{2} \) \( \implies \text{``mod}(x, y) \leq x/2.$$ \)
\[ \text{mod} (x, y) \text{ is second argument in next recursive call,} \]
\[ \text{and becomes the first argument in the next one.} \]

When \( y \geq \frac{x}{2} \), then
\[ \left\lfloor \frac{x}{y} \right\rfloor = 1, \]
\[ \text{mod} (x, y) = x - y \left\lfloor \frac{x}{y} \right\rfloor = x - y \leq x - \frac{x}{2} = \frac{x}{2} \]
Runtime Proof (continued.)

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

Fact:
First arg decreases by at least factor of two in two recursive calls.

Proof of Fact: Recall that first argument decreases every call.

Case 1: \( y < \frac{x}{2} \), first argument is \( y \)
\( \implies \) true in one recursive call;

Case 2: Will show \( y \geq \frac{x}{2} \) \( \implies \) \( \text{mod}(x, y) \leq x/2. \)

\( \text{mod} (x, y) \) is second argument in next recursive call, and becomes the first argument in the next one.

When \( y \geq \frac{x}{2} \), then
\[
\left\lfloor \frac{x}{y} \right\rfloor = 1,
\]
\[
\text{mod} (x, y) = x - y \left\lfloor \frac{x}{y} \right\rfloor = x - y \leq x - \frac{x}{2} = \frac{x}{2}
\]
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.

\[
\text{(define (euclid } x \ y) \\
\text{(if } (= \ y 0) \\
\quad x \\
\quad (euclid \ y \ (\text{mod } x \ y)))
\]

Computes the gcd\((x, y)\) in \(O(n)\) divisions. (Remember \(n = \log_2 x\).)
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. (Remember $n = \log_2 x$.)
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

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

$$ax + by = d$$

where $d = \gcd(x, y)$.

“Make $d$ out of sum of multiples of $x$ and $y$.”
Euclid’s Extended GCD Theorem: For any $x, y$ there are integers $a, b$ such that

$$ax + by = d$$

where $d = \gcd(x, y)$.

“Make $d$ out of sum of multiples of $x$ and $y$.”

What is multiplicative inverse of $x$ modulo $m$?
Euclid’s Extended GCD Theorem: For any $x, y$ there are integers $a, b$ such that
\[ ax + by = d \quad \text{where } d = \gcd(x, y). \]

“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

$$ax + by = d$$

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

$$ax + by = d$$

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
\[ ax + by = d \quad \text{where } d = \gcd(x, y). \]

“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 $x \pmod{m}$!!
**Extended GCD**

**Euclid’s Extended GCD Theorem:** For any \( x, y \) there are integers \( a, b \) such that

\[
ax + by = d
\]

where \( d = \gcd(x, y) \).

“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 \( x \) \((\mod m)!!

Example: For \( x = 12 \) and \( y = 35 \), \( \gcd(12, 35) = 1 \).
Euclid’s Extended GCD Theorem: For any $x, y$ there are integers $a, b$ such that
$$ax + by = d$$
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 $x$ (mod $m$)!!

Example: For $x = 12$ and $y = 35$, $\gcd(12, 35) = 1$.

$$(3)12 + (-1)35 = 1.$$
Euclid’s Extended GCD Theorem: For any $x, y$ there are integers $a, b$ such that
\[ ax + by = d \quad \text{where} \quad d = \gcd(x, y). \]

“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 $b = -1$. 
Euclid’s Extended GCD Theorem: For any $x, y$ there are integers $a, b$ such that

$$ax + by = d$$

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 $\gcd(x, m) = 1$.

$$ax + bm = 1$$

$$ax \equiv 1 \equiv bm \equiv 1 \pmod{m}.$$ 

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 $3$. 


Extended GCD

Euclid’s Extended GCD Theorem: For any \( x, y \) there are integers \( a, b \) such that

\[
ax + by = d \quad \text{where} \quad d = \gcd(x, y).
\]

“Make \( d \) out of sum of multiples of \( x \) and \( y \).”

What is multiplicative inverse of \( x \) modulo \( m \)?

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

\[
ax + bm = 1
\]

\[
ax \equiv 1 \quad \text{and} \quad bm \equiv 1 \quad \pmod{m}.
\]

So \( a \) multiplicative inverse of \( x \) \((\text{mod } m)\)!!

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 \((\text{mod } 35)\) is 3.

Check: 3(12)
Extended GCD

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

“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 \( y = 35 \), \( \gcd(12, 35) = 1 \).
\[
(3)12 + (-1)35 = 1.
\]
\[
a = 3 \text{ and } b = -1.
\]

The multiplicative inverse of 12 (mod 35) is 3.

Check: \( 3(12) = 36 \)
Extended GCD

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

$$ax + by = d$$

where $d = \gcd(x, y)$.

“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 $y = 35$, $\gcd(12, 35) = 1$.

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

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

The multiplicative inverse of 12 (mod 35) is 3.

Check: $3(12) = 36 \equiv 1 \pmod{35}$. 
Make $d$ out of multiples of $x$ and $y$..?

\[ \text{gcd}(35, 12) \]
Make $d$ out of multiples of $x$ and $y$..?

\[
gcd(35, 12) \\
gcd(12, 11) ;; gcd(12, 35 \% 12)
\]
Make $d$ out of multiples of $x$ and $y$..?

\[
\text{gcd}(35, 12) \\
\text{gcd}(12, 11) ;; \text{gcd}(12, 35\%12) \\
\text{gcd}(11, 1) ;; \text{gcd}(11, 12\%11)
\]
Make $d$ out of multiples 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 multiples of \( x \) and \( y \)?

\[
\begin{align*}
gcd(35, 12) \\
gcd(12, 11) ;; gcd(12, 35 \mod 12) \\
gcd(11, 1) ;; gcd(11, 12 \mod 11) \\
gcd(1, 0) \\
1
\end{align*}
\]

How did \( \text{gcd} \) get 11 from 35 and 12?
Make $d$ out of multiples 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?
\[
35 - \left\lfloor \frac{35}{12} \right\rfloor \times 12 = 35 - (2)12 = 11
\]
Make \( d \) out of multiples of \( x \) and \( y \)?

\[
\text{gcd}(35, 12) \\
\text{gcd}(12, 11) \quad \text{;;} \quad \text{gcd}(12, 35 \% 12) \\
\text{gcd}(11, 1) \quad \text{;;} \quad \text{gcd}(11, 12 \% 11) \\
\text{gcd}(1, 0) \\
1
\]

How did \( \text{gcd} \) get 11 from 35 and 12?
\[
35 - \left\lfloor \frac{35}{12} \right\rfloor 12 = 35 - (2)12 = 11
\]

How does \( \text{gcd} \) get 1 from 12 and 11?
Make $d$ out of multiples of $x$ and $y$..?

```plaintext
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 multiples of $x$ and $y$..?

\[
\text{gcd}(35, 12) \\
\text{gcd}(12, 11) ;; \text{gcd}(12, 35 \mod 12) \\
\text{gcd}(11, 1) ;; \text{gcd}(11, 12 \mod 11) \\
\text{gcd}(1, 0) \\
1
\]

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

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

Algorithm finally returns 1.
Make $d$ out of multiples of $x$ and $y$..?

$$\text{gcd}(35, 12)$$
$$\text{gcd}(12, 11) ;; \text{gcd}(12, 35 \mod 12)$$
$$\text{gcd}(11, 1) ;; \text{gcd}(11, 12 \mod 11)$$
$$\text{gcd}(1, 0)$$
$$1$$

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 multiples of $x$ and $y$..?

\[
\begin{align*}
gcd(35, 12) \\
gcd(12, 11) ;;& \quad gcd(12, 35 \% 12) \\
gcd(11, 1) ;;& \quad gcd(11, 12 \% 11) \\
gcd(1, 0) \\
1
\end{align*}
\]

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

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

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 multiples 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?
$$35 - \left\lfloor \frac{35}{12} \right\rfloor 12 = 35 - (2)12 = 11$$

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

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 multiples of $x$ and $y$..?

\[
gcd(35, 12) \\
gcd(12, 11) ;; \ gcd(12, 35 \mod 12) \\
gcd(11, 1) ;; \ gcd(11, 12 \mod 11) \\
gcd(1, 0) \\
1
\]

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

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

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 multiples 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?
$$35 - \left\lfloor \frac{35}{12} \right\rfloor 12 = 35 - (2)12 = 11$$

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

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 multiples of \( x \) and \( y \) ..?

\[
\begin{align*}
gcd(35, 12) \\
gcd(12, 11) ;; gcd(12, 35 \% 12) \\
gcd(11, 1) ;; gcd(11, 12 \% 11) \\
gcd(1, 0) \\
1
\end{align*}
\]

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 multiples of $x$ and $y$..?

\[
\begin{align*}
gcd(35, 12) \\
gcd(12, 11) \quad ;; \quad gcd(12, 35 \% 12) \\
gcd(11, 1) \quad ;; \quad gcd(11, 12 \% 11) \\
gcd(1, 0) \\
1
\end{align*}
\]

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

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

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.

\[
\text{ext-gcd}(x,y)
\]

if \( y = 0 \) then return \((x, 1, 0)\)
else

\((d, a, b) := \text{ext-gcd}(y, \text{mod}(x,y))\)
return \((d, b, a - \text{floor}(x/y) \times b)\)

Claim: Returns \((d, a, b)\): \(d = \gcd(a, b)\) and \(d = ax + by\).
Extended GCD Algorithm.

\[
\text{ext-gcd}(x, y) \\
\quad \text{if } y = 0 \text{ then return } (x, 1, 0) \\
\quad \text{else} \\
\quad \quad (d, a, b) := \text{ext-gcd}(y, \mod(x, y)) \\
\quad \quad \text{return } (d, b, a - \text{floor}(x/y) \times b)
\]

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

Example:

\[
\text{ext-gcd}(35, 12)
\]
Extended GCD Algorithm.

\[
\text{ext-gcd}(x, y)
\begin{align*}
\text{if } y & = 0 \text{ then return } (x, 1, 0) \\
\text{else} \\
(d, a, b) & := \text{ext-gcd}(y, \text{mod}(x, y)) \\
\text{return } (d, b, a - \text{floor}(x/y) \times b)
\end{align*}
\]

Claim: Returns \((d, a, b)\): \(d = \text{gcd}(a, b)\) and \(d = ax + by\).

Example:

\[
\begin{align*}
\text{ext-gcd}(35, 12) \\
\text{ext-gcd}(12, 11)
\end{align*}
\]
Extended GCD Algorithm.

```
text-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)
```

15 / 27
Extended GCD Algorithm.

\[
\text{ext-gcd}(x, y) \\
\quad \text{if } y = 0 \text{ then return } (x, 1, 0) \\
\quad \text{else} \\
\quad \quad (d, a, b) := \text{ext-gcd}(y, \mod(x, y)) \\
\quad \quad \text{return } (d, b, a - \text{floor}(x/y) \times b)
\]

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

\[
\begin{align*}
\text{ext-gcd}(35, 12) \\
\text{ext-gcd}(12, 11) \\
\text{ext-gcd}(11, 1) \\
\text{ext-gcd}(1, 0)
\end{align*}
\]
Extended GCD Algorithm.

```plaintext
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: \(a - \lceil x/y \rceil \cdot b = \)

```plaintext
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.

\[
\text{ext-gcd}(x, y) \\
\text{if } y = 0 \text{ then return } (x, 1, 0) \\
\text{else} \\
(\text{d, a, b}) := \text{ext-gcd}(y, \text{mod}(x, y)) \\
\text{return } (\text{d, b, a - floor}(x/y) \times b)
\]

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

\[
\text{ext-gcd}(35, 12) \\
\text{ext-gcd}(12, 11) \\
\quad \text{ext-gcd}(11, 1) \\
\quad \text{ext-gcd}(1, 0) \\
\text{return } (1, 1, 0) ;; 1 = (1)1 + (0)0 \\
\text{return } (1, 0, 1) ;; 1 = (0)11 + (1)1
\]
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: \[ a - \left\lfloor \frac{x}{y} \right\rfloor \cdot b = 0 - \left\lfloor \frac{12}{11} \right\rfloor \cdot 1 = -1 \]

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.

```plaintext
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: \(a - \lfloor x/y \rfloor \cdot b = 1 - \lfloor 35/12 \rfloor \cdot (-1) = 3\)

```
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.

\[
\text{ext-gcd}(x, y)
\]

if \( y = 0 \) then return \((x, 1, 0)\)
else

\[
(d, a, b) := \text{ext-gcd}(y, \text{mod}(x, y))
\]

return \((d, b, a - \text{floor}(x/y) \times b)\)

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

Example:

\[
\text{ext-gcd}(35, 12)
\]

\[
\text{ext-gcd}(12, 11)
\]

\[
\text{ext-gcd}(11, 1)
\]

\[
\text{ext-gcd}(1, 0)
\]

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.

```plaintext
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 \(d = ax + by\).
Extended GCD Algorithm.

```plaintext
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 \(d = \text{gcd}(x, y)\) by previous proof.
Correctness.

Proof: Strong Induction.\footnote{Assume $d$ is $gcd(x, y)$ by previous proof.}

Base: ext-gcd($x, 0$) returns ($d = x, 1, 0$) with $x = (1)x + (0)y$. 
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 \(d = Ax + By\)

¹Assume \(d\) 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

\[ d = ay + b(\text{mod} (x, y)) \]

¹Assume d is gcd(x, y) by previous proof.
Correctness.

Proof: Strong Induction.\(^1\)
Base: \(\text{ext-gcd}(x, 0)\) returns \((d = x, 1, 0)\) with \(x = (1)x + (0)y\).

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

\(\text{ext-gcd}(x, y)\) calls \(\text{ext-gcd}(y, \text{mod } (x, y))\) so

\(^1\)Assume \(d\) is \(gcd(x, y)\) by previous proof.
Correctness.

**Proof:** Strong Induction.\(^1\)

**Base:** \( \text{ext-gcd}(x,0) \) returns \((d = x, 1, 0)\) with \(x = (1)x + (0)y\).

**Induction Step:** Returns \((d, A, B)\) with \(d = Ax + By\)

Ind hyp: \( \text{ext-gcd}(y, \mod (x, y)) \) returns \((d, a, b)\) with \(d = ay + b(\mod (x, y))\)

\( \text{ext-gcd}(x, y) \) calls \( \text{ext-gcd}(y, \mod (x, y)) \) so

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

---

\(^1\)Assume \(d\) is \(\text{gcd}(x, y)\) 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

\[
    d = ay + b(\mod (x, y))
\]
\[
    = ay + b(x - \lfloor \frac{x}{y} \rfloor y)
\]

\(^1\)Assume \(d\) is \(gcd(x, y)\) by previous proof.
Correctness.

**Proof:** Strong Induction.¹

**Base:** \text{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: \textbf{ext-gcd}(y, \mod (x, y)) returns \((d, a, b)\) with
\[
d = ay + b \mod (x, y)
\]

\text{ext-gcd}(x, y) calls \textbf{ext-gcd}(y, \mod (x, y)) so
\[
d = ay + b \cdot (\mod (x, y))
\]
\[
= ay + b \cdot (x - \left\lfloor \frac{x}{y} \right\rfloor y)
\]
\[
= bx + (a - \left\lfloor \frac{x}{y} \right\rfloor \cdot b)y
\]

¹Assume \(d\) is \(gcd(x, y)\) by previous proof.
Correctness.

Proof: Strong Induction.\(^1\)

**Base:** \(\text{ext-gcd}(x, 0)\) returns \((d = x, 1, 0)\) with \(x = (1)x + (0)y\).

**Induction Step:** Returns \((d, A, B)\) with \(d = Ax + By\)

Ind hyp: \(\text{ext-gcd}(y, \mod (x, y))\) returns \((d, a, b)\) with
\[
d = ay + b(\mod (x, y))
\]

\(\text{ext-gcd}(x, y)\) calls \(\text{ext-gcd}(y, \mod (x, y))\) so
\[
d = ay + b(\mod (x, y))
\]
\[
= ay + b(x - \frac{x}{y} y)
\]
\[
= bx + (a - \lfloor \frac{x}{y} \rfloor b)y
\]

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

\(^1\)Assume \(d\) is \(gcd(x, y)\) by previous proof.
Correctness.

Proof: Strong Induction.\(^1\)

Base: \(\text{ext-gcd}(x, 0)\) returns \((d = x, 1, 0)\) with \(x = (1)x + (0)y\).

Induction Step: Returns \((d, A, B)\) with \(d = Ax + By\)

Ind hyp: \(\text{ext-gcd}(y, \text{mod}(x, y))\) returns \((d, a, b)\) with \(d = ay + b(\text{mod}(x, y))\)

\(\text{ext-gcd}(x, y)\) calls \(\text{ext-gcd}(y, \text{mod}(x, y))\) so

\[
d = ay + b \cdot (\text{mod}(x, y))
\]

\[
= ay + b \cdot (x - \lfloor \frac{x}{y} \rfloor y)
\]

\[
= bx + (a - \lfloor \frac{x}{y} \rfloor \cdot b)y
\]

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

---

\(^1\)Assume \(d\) is \(\text{gcd}(x, y)\) by previous proof.

\[
\text{ext-gcd}(x, y)
\]
\[
\begin{align*}
& \text{if } y = 0 \text{ then return } (x, 1, 0) \\
& \text{else} \\
& \quad (d, a, b) := \text{ext-gcd}(y, \text{mod}(x, y)) \\
& \quad \text{return } (d, b, a - \text{floor}(x/y) \times b)
\end{align*}
\]

\[
\text{ext-gcd}(x, y) \\
\begin{align*}
\text{if } y &= 0 \text{ then return } (x, 1, 0) \\
\text{else} & \\
& (d, a, b) := \text{ext-gcd}(y, \text{mod}(x, y)) \\
& \text{return } (d, b, a - \text{floor}(x/y) \times b)
\end{align*}
\]

Recursively: \[d = ay + b(x - \lceil \frac{x}{y} \rceil \cdot y)\]

\[
\text{ext-gcd}(x,y) \\
\quad \begin{cases} 
\text{if } y = 0 \text{ then return } (x, 1, 0) \\
\text{else} \\
\quad \quad (d, a, b) := \text{ext-gcd}(y, \text{mod}(x,y)) \\
\quad \quad \text{return } (d, b, a - \text{floor}(x/y) \times b) 
\end{cases}
\]

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

\[
\text{ext-gcd}(x, y) \\
\quad \text{if } y = 0 \text{ then return } (x, 1, 0) \\
\quad \text{else} \\
\quad \quad (d, a, b) := \text{ext-gcd}(y, \mod(x, y)) \\
\quad \quad \text{return } (d, b, a - \text{floor}(x/y) \times b)
\]

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

Returns \((d, b, (a - \lfloor \frac{x}{y} \rfloor \cdot b))\).
Hand Calculation Method for Inverses.

Example: \( \gcd(7, 60) = 1. \)
Hand Calculation Method for Inverses.

Example: \( \gcd(7, 60) = 1. \)
\[ \text{egcd}(7,60). \]
Example: \( \text{gcd}(7,60) = 1 \).
\[ \text{egcd}(7,60). \]

\[ 7(0) + 60(1) = 60 \]

Confirm:
\[ -119 + 120 = 1 \]

Note: an "iterative" version of the e-gcd algorithm.
Hand Calculation Method for Inverses.

Example: $\text{gcd}(7, 60) = 1$.

$\text{egcd}(7, 60)$.

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7
\end{align*}
\]
Hand Calculation Method for Inverses.

Example: \( \gcd(7, 60) = 1 \).
\[ \text{egcd}(7, 60). \]

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4
\end{align*}
\]
Hand Calculation Method for Inverses.

Example: \(\gcd(7, 60) = 1.\)

\[\text{egcd}(7, 60).\]

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3
\end{align*}
\]
Example: \( \gcd(7, 60) = 1 \).
\( \text{egcd}(7, 60) \).

\[
egin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{align*}
\]
Hand Calculation Method for Inverses.

Example: \( \text{gcd}(7, 60) = 1 \).

\[
\text{egcd}(7, 60).
\]

\[
7(0) + 60(1) = 60
\]
\[
7(1) + 60(0) = 7
\]
\[
7(-8) + 60(1) = 4
\]
\[
7(9) + 60(-1) = 3
\]
\[
7(-17) + 60(2) = 1
\]
Hand Calculation Method for Inverses.

Example: \( \gcd(7, 60) = 1 \).
\[ \text{egcd}(7, 60). \]

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{align*}
\]

Confirm:
Hand Calculation Method for Inverses.

Example: \( \text{gcd}(7, 60) = 1 \).
\[
\text{egcd}(7, 60).
\]

\[
7(0) + 60(1) = 60
\]
\[
7(1) + 60(0) = 7
\]
\[
7(-8) + 60(1) = 4
\]
\[
7(9) + 60(-1) = 3
\]
\[
7(-17) + 60(2) = 1
\]

Confirm: \(-119 + 120 = 1\)
Hand Calculation Method for Inverses.

Example: \( \text{gcd}(7, 60) = 1 \).

\[
\text{egcd}(7, 60).
\]

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{align*}
\]

Confirm: \(-119 + 120 = 1\)

Note: an “iterative” version of the e-gcd algorithm.
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...
$2^{n/2}$

Internet Security.

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

Internet Security: Soon.
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 $1,000,000,000,000,000$?
$\leq 80$ divisions.
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?

$\leq 80$ 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 1,000,000,000,000,000?

$\leq 80$ divisions.

versus 1,000,000
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?

$\leq 80$ 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 $1,000,000,000,000,000$?
$\leq 80$ divisions.
versus $1,000,000$

Internet Security.
Public Key Cryptography: 512 digits.
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 1,000,000,000,000,000?
$\leq 80$ 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?

$\leq 80$ divisions.

versus 1,000,000

Internet Security.

Public Key Cryptography: 512 digits.

512 divisions vs.

$(100000000000000000000000000000000000000000000000000000000)^5$ 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?

$\leq 80$ divisions.

versus 1,000,000

Internet Security.

Public Key Cryptography: 512 digits.

512 divisions vs.

$(100000000000000000000000000000000000000000)^5$ divisions.

Internet Security:
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?

$\leq 80$ divisions.

versus 1,000,000

Internet Security.

Public Key Cryptography: 512 digits.

512 divisions vs.

$(10000000000000000000000000000000000000000000) ^ 5$ divisions.

Internet Security: Soon.
Bijections

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

Bijection:

\[
\begin{align*}
\text{Domain: } & \mathcal{A}, \quad \text{Co-Domain: } \mathcal{B} \\
\text{Versus Range.} & \\
E.g. \sin(x) \\
A = B = \text{reals.} \\
\text{Range is } & \left[ -1, 1 \right] \\
\text{Onto: } & \left[ -1, 1 \right] \\
\text{Not one-to-one.} & \\
\text{Range Definition always is onto.} \\
\text{Consider } & f(x) = ax \mod m. \\
\text{When is it a bijection?} & \\
\text{When } \gcd(a, m) & = 1. \\
\text{Not Example: } a = 2, m = 4, \\
f(0) = f(2) = 0 \mod 4.
\end{align*}
\]
Bijections

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

**Bijection:**

\[ f : A \rightarrow B. \]
Bijections

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

Bijection:
\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).
Bijections

**Bijection** is **one to one and onto**.
Bijection:
   \[ f : A \to B. \]
Domain: \( A \), Co-Domain: \( B \).
   Versus Range.

E.g. \( \sin(x) \). \( A = B = \text{reals.} \)
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 \( \gcd(a, m) = 1. \)
Not Example: \( a = 2, m = 4, f(0) = f(2) = 0 \text{ mod } 4. \)
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. \( \sin (x) \).

\[ A = B = \text{reals}. \]
Bijections

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

**Bijection:**
\[ f : A \rightarrow B. \]

**Domain:** \( A \), **Co-Domain:** \( B \).

**Versus Range.**

E.g. **sin** (\( x \)).

\[ A = B = \text{reals}. \]

**Range** is \([-1, 1]\).
Bijections

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

Bijection:

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin(x) \).

\[ A = B = \text{reals.} \]

Range is \([-1, 1]\). Onto: \([-1, 1]\).
Bijections

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

**Bijection:**

\[ f : A \rightarrow B. \]

**Domain:** \( A \), **Co-Domain:** \( B \).

**Versus Range.**

E.g. **\( \sin \ (x) \).**

\[ A = B = \text{reals}. \]

Range is \([-1, 1]\). **Onto:** \([-1, 1]\).

Not one-to-one.
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 \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \( \sin (\pi) = \sin (0) = 0. \)
Bijections

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

Bijection:

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin (x) \).

\( A = B = \text{reals}. \)

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:

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin (x) \).

\[ A = B = \text{reals}. \]

Range is \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \( \sin (\pi) = \sin (0) = 0. \)

Range Definition always is onto.

Consider \( f(x) = ax \mod m. \)
Bijections

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

Bijection:

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin (x) \).

\( A = B = \text{reals}. \)

Range is \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \( \sin (\pi) = \sin (0) = 0. \)

Range Definition always is onto.

Consider \( f(x) = ax \mod m. \)

\( f : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, m-1\}. \)
Bijections

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

Bijection:

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin(x) \).

\( A = B = \text{reals}. \)

Range is \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \( \sin(\pi) = \sin(0) = 0. \)

Range Definition always is onto.

Consider \( f(x) = ax \mod m. \)

\( f: \{0,\ldots,m-1\} \rightarrow \{0,\ldots,m-1\}. \)

Domain/Co-Domain: \( \{0,\ldots,m-1\}. \)
**Bijections**

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

Bijection:

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin (x) \).

\( A = B = \text{reals.} \)

Range is \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \( \sin (\pi) = \sin (0) = 0. \)

Range Definition always is onto.

Consider \( f(x) = ax \mod m. \)

\( f : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, m-1\}. \)

Domain/Co-Domain: \( \{0, \ldots, m-1\}. \)

When is it a bijection?
Bijections

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

Bijection:

\[ f : A \rightarrow B. \]

Domain: \(A\), Co-Domain: \(B\).

Versus Range.

E.g. \(\sin(x)\).

\(A = B = \text{reals}.\)

Range is \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \(\sin(\pi) = \sin(0) = 0.\)

Range Definition always is onto.

Consider \(f(x) = ax \mod m.\)

\(f : \{0, \ldots, m - 1\} \rightarrow \{0, \ldots, m - 1\}.\)

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:

\[ f : A \rightarrow B. \]

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 \( f(x) = ax \mod m \).

\[ f : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, m-1\}. \]

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:

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin(x) \).

\( A = B = \text{reals} \).

Range is \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \( \sin(\pi) = \sin(0) = 0 \).

Range Definition always is onto.

Consider \( f(x) = ax \mod m \).

\[ f : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, m-1\}. \]

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:

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin(x) \).

\[ A = B = \text{reals}. \]

Range is \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \( \sin(\pi) = \sin(0) = 0 \).

Range Definition always is onto.

Consider \( f(x) = ax \mod m \).

\[ f : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, m-1\}. \]

Domain/Co-Domain: \( \{0, \ldots, m-1\} \).

When is it a bijection?

When \( \gcd(a, m) \) is ....? ... 1.

Not Example: \( a = 2, m = 4 \),
Bijectivity

Bijectivity is one to one and onto.

Bijectivity: 

\[ f : A \rightarrow B. \]

Domain: \( A \), Co-Domain: \( B \).

Versus Range.

E.g. \( \sin(x) \).

\( A = B = \text{reals}. \)

Range is \([-1, 1]\). Onto: \([-1, 1]\).

Not one-to-one. \( \sin(\pi) = \sin(0) = 0. \)

Range Definition always is onto.

Consider \( f(x) = ax \mod m. \)

\[ f : \{0, \ldots, m-1\} \rightarrow \{0, \ldots, m-1\}. \]

Domain/Co-Domain: \( \{0, \ldots, m-1\}. \)

When is it a bijection?

When \( \gcd(a, m) \) is ....? ... 1.

Not Example: \( a = 2, m = 4, f(0) = f(2) = 0 \mod 4. \)
Lots of Mods

\[ x = 5 \pmod{7} \] and \[ x = 3 \pmod{5} \].
Lots of Mods

\[ x = 5 \pmod{7} \text{ and } x = 3 \pmod{5}. \]

What is \( x \pmod{35} \)?
$x = 5 \pmod{7}$ 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

\[ x = 5 \pmod{7} \text{ and } x = 3 \pmod{5}. \]

What is \( x \pmod{35} \)?

Let’s try 5. Not 3 \( \pmod{5} \)!

Let’s try 3.
What is $x \pmod{35}$?

Let’s try 5. Not 3 (mod 5)!
Let’s try 3. Not 5 (mod 7)!
$x = 5 \pmod{7}$ and $x = 3 \pmod{5}$.

What is $x \pmod{35}$?

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

Let's try 3. Not $5 \pmod{7}$!
$x = 5 \pmod{7}$ 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 $x = 5 \pmod{7}$
Lots of Mods

\[ x = 5 \pmod{7} \text{ and } x = 3 \pmod{5}. \]

What is \( x \pmod{35} \)?

Let’s try 5. Not 3 \((\text{mod } 5)!\)
Let’s try 3. Not 5 \((\text{mod } 7)!\)

If \( x = 5 \pmod{7} \)
   then \( x \) is in \( \{5, 12, 19, 26, 33\} \).
Lots of Mods

\[ x = 5 \pmod{7} \text{ and } x = 3 \pmod{5}. \]

What is \( x \pmod{35} \)?

Let’s try 5. Not 3 \( \pmod{5} \)!
Let’s try 3. Not 5 \( \pmod{7} \)!

If \( x = 5 \pmod{7} \)
then \( x \) is in \( \{5, 12, 19, 26, 33\} \).
Lots of Mods

\[ x = 5 \pmod{7} \text{ and } x = 3 \pmod{5}. \]

What is \( x \pmod{35} \)?

Let’s try 5. Not \( 3 \pmod{5} \)!

Let’s try 3. Not \( 5 \pmod{7} \)!

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

Oh, only 33 is 3 $(\pmod{5})$.

Hmmm...
Lots of Mods

\[ x = 5 \pmod{7} \text{ and } x = 3 \pmod{5}. \]

What is \( x \pmod{35} \)?

Let’s try 5. Not 3 \( \pmod{5} \)!

Let’s try 3. Not 5 \( \pmod{7} \)!

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

Oh, only 33 is 3 \( \pmod{5} \).

Hmmmm... only one solution.
$x = 5 \pmod{7}$ 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 $x = 5 \pmod{7}$
  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.
Simple Chinese Remainder Theorem.

My love is won.
Simple Chinese Remainder Theorem.

My love is won. Zero and One.
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$.
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

---

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

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

$u = 0 \pmod{n}$

$u = 1 \pmod{m}$

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

$v = 1 \pmod{n}$

$v = 0 \pmod{m}$

Let $x = au + bv$.

$x = a \pmod{m}$ 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 $(x-y) \equiv 0 \pmod{n}$.

$\Rightarrow (x-y)$ is multiple of $m$ and $n$ since $\gcd(m, n)=1$.

$\Rightarrow x-y \geq mn$ since $x, y \notin \{0, \ldots, mn-1\}$.

Thus, only one solution modulo $mn$. 

23 / 27
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution $x \pmod{mn}$. 
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1} \pmod{m}) \).
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

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

**Proof:**
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1} \pmod{m}) \).
\[
\begin{align*}
u & = 0 \pmod{n} \quad u = 1 \pmod{m}
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

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

**Proof:**
Consider $u = n(n^{-1} \pmod{m})$.
\[ u = 0 \pmod{n} \quad u = 1 \pmod{m} \]

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

My love is won. Zero and One. Nothing and nothing done.

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

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

**Proof:**

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

$u = 0 \pmod{n}$ \hspace{1cm} $u = 1 \pmod{m}$

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

$v = 1 \pmod{n}$
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**

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

\[
\begin{align*}
    u &= 0 \pmod{n} & u &= 1 \pmod{m} \\
    \end{align*}
\]

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

\[
\begin{align*}
    v &= 1 \pmod{n} & v &= 0 \pmod{m} \\
    \end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**

Consider \( u = n(n^{-1} \pmod{m}) \).
- \( u = 0 \pmod{n} \quad \text{and} \quad u = 1 \pmod{m} \)

Consider \( v = m(m^{-1} \pmod{n}) \).
- \( v = 1 \pmod{n} \quad \text{and} \quad v = 0 \pmod{m} \)
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

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

**Proof:**
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ \quad $u = 1 \pmod{m}$

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

Let $x = au + bv$. 

Simple Chinese Remainder Theorem.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1}) \pmod{m} \).
\[
\begin{align*}
  u &= 0 \pmod{n} & u &= 1 \pmod{m} \\
\end{align*}
\]

Consider \( v = m(m^{-1}) \pmod{n} \).
\[
\begin{align*}
  v &= 1 \pmod{n} & v &= 0 \pmod{m} \\
\end{align*}
\]

Let \( x = au + bv \).
\[
\begin{align*}
  x &= a \pmod{m} \\
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

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

**Proof:**
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$  \hspace{1cm} $u = 1 \pmod{m}$

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

Let $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1}) \pmod{m} \).
\[
\begin{align*}
u &= 0 \pmod{n} & u &= 1 \pmod{m} \\
\end{align*}
\]
Consider \( v = m(m^{-1}) \pmod{n} \).
\[
\begin{align*}
v &= 1 \pmod{n} & v &= 0 \pmod{m} \\
\end{align*}
\]
Let \( x = au + bv \).
\[
\begin{align*}
x &= a \pmod{m} \text{ since } bv = 0 \pmod{m} \text{ and } au = a \pmod{m} \\
\end{align*}
\]
Simple Chinese Remainder Theorem.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1} \pmod{m}) \).
\[
\begin{align*}
    u &= 0 \pmod{n} \\
    u &= 1 \pmod{m}
\end{align*}
\]
Consider \( v = m(m^{-1} \pmod{n}) \).
\[
\begin{align*}
    v &= 1 \pmod{n} \\
    v &= 0 \pmod{m}
\end{align*}
\]
Let \( x = au + bv \).
\[
\begin{align*}
    x &= a \pmod{m} \quad \text{since } bv = 0 \pmod{m} \text{ and } au = a \pmod{m} \\
    x &= b \pmod{n}
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1}) \pmod{m} \).
\[
\begin{align*}
  u &= 0 \pmod{n} \\
  u &= 1 \pmod{m}
\end{align*}
\]
Consider \( v = m(m^{-1}) \pmod{n} \).
\[
\begin{align*}
  v &= 1 \pmod{n} \\
  v &= 0 \pmod{m}
\end{align*}
\]
Let \( x = au + bv \).
\[
\begin{align*}
  x &= a \pmod{m} \quad \text{since } bv = 0 \pmod{m} \text{ and } au = a \pmod{m} \\
  x &= b \pmod{n} \quad \text{since } au = 0 \pmod{n} \text{ and } bv = b \pmod{n}
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1} \pmod{m}) \).
\[
u = 0 \pmod{n} \quad u = 1 \pmod{m}
\]

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

Let \( x = au + bv \).
\[
x = a \pmod{m} \quad \text{since } bv = 0 \pmod{m} \text{ and } au = a \pmod{m}
\]
\[
x = b \pmod{n} \quad \text{since } au = 0 \pmod{n} \text{ and } bv = b \pmod{n}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1} \pmod{m}) \).
\[
\begin{align*}
    u &= 0 \pmod{n} & u &= 1 \pmod{m}
\end{align*}
\]

Consider \( v = m(m^{-1} \pmod{n}) \).
\[
\begin{align*}
    v &= 1 \pmod{n} & v &= 0 \pmod{m}
\end{align*}
\]

Let \( x = au + bv \).
\[
\begin{align*}
    x &= a \pmod{m} & \text{since } bv &= 0 \pmod{m} \text{ and } au = a \pmod{m} \\
    x &= b \pmod{n} & \text{since } au &= 0 \pmod{n} \text{ and } bv = b \pmod{n}
\end{align*}
\]

Only solution?
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**

Consider \( u = n(n^{-1} \pmod{m}) \).
\[
\begin{align*}
u & = 0 \pmod{n} & u & = 1 \pmod{m}
\end{align*}
\]

Consider \( v = m(m^{-1} \pmod{n}) \).
\[
\begin{align*}
v & = 1 \pmod{n} & v & = 0 \pmod{m}
\end{align*}
\]

Let \( x = au + bv \).
\[
\begin{align*}
x & = a \pmod{m} \text{ since } bv = 0 \pmod{m} \text{ and } au = a \pmod{m} \\
x & = b \pmod{n} \text{ since } au = 0 \pmod{n} \text{ and } bv = b \pmod{n}
\end{align*}
\]

Only solution? If not, two solutions, \( x \) and \( y \).
Simple Chinese Remainder Theorem.

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

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

**Proof:**
Consider $u = n(n^{-1} \pmod{m})$.
\[
\begin{align*}
u &= 0 \pmod{n} \quad u &= 1 \pmod{m}
\end{align*}
\]
Consider $v = m(m^{-1} \pmod{n})$.
\[
\begin{align*}
v &= 1 \pmod{n} \quad v &= 0 \pmod{m}
\end{align*}
\]
Let $x = au + bv$.
\[
\begin{align*}
x &= a \pmod{m} \quad &\text{since} \quad bv = 0 \pmod{m} \quad \text{and} \quad au = a \pmod{m} \\
x &= b \pmod{n} \quad &\text{since} \quad au = 0 \pmod{n} \quad \text{and} \quad bv = b \pmod{n}
\end{align*}
\]
Only solution? If not, two solutions, $x$ and $y$.
\[
(x - y) \equiv 0 \pmod{m} \quad \text{and} \quad (x - y) \equiv 0 \pmod{n}.
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1} \pmod{m}) \).
\[
u = 0 \pmod{n} \quad u = 1 \pmod{m}\]

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

Let \( x = au + bv \).
\[
x = a \pmod{m} \] 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 \( (x - y) \equiv 0 \pmod{n} \).
\[
\implies (x - y) \text{ is multiple of } m \text{ and } n \text{ since } \gcd(m, n) = 1.\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**
Consider \( u = n(n^{-1}) \pmod{m} \).
\[
\begin{align*}
u &= 0 \pmod{n} && u = 1 \pmod{m} \\
\end{align*}
\]
Consider \( v = m(m^{-1}) \pmod{n} \).
\[
\begin{align*}
v &= 1 \pmod{n} && v = 0 \pmod{m} \\
\end{align*}
\]
Let \( x = au + bv \).
\[
\begin{align*}x &= a \pmod{m} \quad \text{since } bv = 0 \pmod{m} \text{ and } au = a \pmod{m} \\
x &= b \pmod{n} \quad \text{since } au = 0 \pmod{n} \text{ and } bv = b \pmod{n} \\
\end{align*}
\]
Only solution? If not, two solutions, \( x \) and \( y \).
\[
\begin{align*}(x - y) &\equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}. \\
\implies (x - y) \text{ is multiple of } m \text{ and } n \text{ since } \gcd(m, n) = 1. \\
\implies x - y \geq mn
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**

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

- \( u = 0 \pmod{n} \)
- \( u = 1 \pmod{m} \)

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

- \( v = 1 \pmod{n} \)
- \( v = 0 \pmod{m} \)

Let \( x = au + bv \).

- \( x = a \pmod{m} \) 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.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**

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

\[
\begin{align*}
u & = 0 \pmod{n} \quad u = 1 \pmod{m} \\
u & = 1 \pmod{n} \\
u & = 0 \pmod{m}
\end{align*}
\]

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

\[
\begin{align*}
v & = 1 \pmod{n} \quad v = 0 \pmod{m} \\
v & = 1 \pmod{n} \\
v & = 0 \pmod{m}
\end{align*}
\]

Let \( x = au + bv \).

\[
\begin{align*}
x & = a \pmod{m} \quad \text{since } bv = 0 \pmod{m} \text{ and } au = a \pmod{m} \\
x & = b \pmod{n} \quad \text{since } au = 0 \pmod{n} \text{ and } bv = b \pmod{n}
\end{align*}
\]

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

\[
(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.
\]

\[\implies (x - y) \text{ is multiple of } m \text{ and } n \text{ since } \gcd(m, n) = 1.\]

\[\implies x - y \geq mn \implies x, y \not\in \{0, \ldots, mn - 1\}.\]

Thus, only one solution modulo \( mn \).
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

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

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof:**

Consider \( u = n(n^{-1} \pmod{m}) \).
\[
\begin{align*}
u &= 0 \pmod{n} \\
u &= 1 \pmod{m}
\end{align*}
\]

Consider \( v = m(m^{-1} \pmod{n}) \).
\[
\begin{align*}
v &= 1 \pmod{n} \\
v &= 0 \pmod{m}
\end{align*}
\]

Let \( x = au + bv \).
\[
\begin{align*}
x &= a \pmod{m} \text{ since } bv = 0 \pmod{m} \text{ and } au = a \pmod{m} \\
x &= b \pmod{n} \text{ since } au = 0 \pmod{n} \text{ and } bv = b \pmod{n}
\end{align*}
\]

Only solution? If not, two solutions, \( x \) and \( y \).
\[
\begin{align*}(x - y) &\equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.
\end{align*}
\]
\[
\Rightarrow (x - y) \text{ is multiple of } m \text{ and } n \text{ since } \gcd(m, n) = 1.
\]
\[
\Rightarrow x - y \geq mn \Rightarrow x, y \not\in \{0, \ldots, mn - 1\}.
\]

Thus, only one solution modulo \( mn \).
CRT: isomorphism.

For $m, n$, $\gcd(m, n) = 1$. 

Also, true that $x \mod mn \leftrightarrow x = a \mod m$ and $x = b \mod n$. 

$y \mod mn \leftrightarrow y = c \mod m$ and $y = d \mod n$. 

Mapping is “isomorphic”: corresponding addition (and multiplication) operations consistent with mapping.
CRT: isomorphism.

For $m, n$, $\gcd(m, n) = 1$.

$x \mod mn \leftrightarrow x = a \mod m \text{ and } x = b \mod n$
CRT: isomorphism.

For $m, n$, $\gcd(m, n) = 1$.

$x \mod mn \iff x = a \mod m$ and $x = b \mod n$

$y \mod mn \iff y = c \mod m$ and $y = d \mod n$
For $m, n$, $\gcd(m, n) = 1$.

$x \mod mn \leftrightarrow x = a \mod m$ and $x = b \mod n$

$y \mod mn \leftrightarrow y = c \mod m$ and $y = d \mod n$

Also, true that $x + y \mod mn \leftrightarrow a + c \mod m$ and $b + d \mod n$. 

CRT: isomorphism.
CRT: isomorphism.

For $m, n$, $\gcd(m, n) = 1$.

$x \mod mn \leftrightarrow x = a \mod m$ and $x = b \mod n$

$y \mod mn \leftrightarrow y = c \mod m$ and $y = d \mod n$

Also, true that $x + y \mod mn \leftrightarrow a + c \mod m$ and $b + d \mod n$.

Mapping is “isomorphic”: corresponding addition (and multiplication) operations consistent with mapping.
Fermat’s Theorem: Reducing Exponents.

Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$,
Fermat’s Theorem: Reducing Exponents.

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

$$a^{p-1} \equiv 1 \pmod{p}.$$
Fermat’s Theorem: Reducing Exponents.

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

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

**Proof:**

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

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

$S$ contains representative of $\{1, \ldots, p-1\}$ modulo $p$.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},$$ since multiplication is commutative.

$a \cdot (p-1) \cdot 1 \cdots (p-1) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}$.

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

Solve to get...

$$a^{p-1} \equiv 1 \pmod{p}.$$
Fermat’s Theorem: Reducing Exponents.

Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$, $a^{p-1} \equiv 1 \pmod{p}$.

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

Fermat’s Theorem: Reducing Exponents.

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

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

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p - 1)\}$. All different modulo $p$ since $a$ has an inverse modulo $p$. 
Fermat’s Theorem: Reducing Exponents.

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

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

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

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

$S$ contains representative of $\{1, \ldots, p-1\}$ modulo $p$. 
Fermat’s Theorem: Reducing Exponents.

**Fermat’s Little Theorem:** For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),
\[ a^{p-1} \equiv 1 \pmod{p}. \]

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

All different modulo \( p \) since \( a \) has an inverse modulo \( p \).

\( S \) contains representative of \( \{1, \ldots, p-1\} \) modulo \( p \).

\[
(a \cdot 1)(a \cdot 2)\cdots(a \cdot (p-1)) \equiv 1 \cdot 2 \cdot \cdots (p-1) \pmod{p},
\]
Fermat’s Theorem: Reducing Exponents.

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

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

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

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

$S$ contains representative of $\{1, \ldots, p-1\}$ modulo $p$.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},$$

Since multiplication is commutative.
Fermat’s Theorem: Reducing Exponents.

**Fermat’s Little Theorem:** For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),
\[
a^{p-1} \equiv 1 \pmod{p}.
\]

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

All different modulo \( p \) since \( a \) has an inverse modulo \( p \).

\( S \) contains representative of \( \{1, \ldots, p - 1\} \) modulo \( p \).

\[
(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p - 1)) \equiv 1 \cdot 2 \cdots (p - 1) \pmod{p},
\]

Since multiplication is commutative.

\[
a^{p-1}(1 \cdots (p - 1)) \equiv (1 \cdots (p - 1)) \pmod{p}.
\]
Fermat’s Theorem: Reducing Exponents.

**Fermat’s Little Theorem:** For prime $p$, and $a \equiv 0 \pmod{p}$, $a^{p-1} \equiv 1 \pmod{p}$.

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

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

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},$$

Since multiplication is commutative.

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

Each of $2, \ldots (p-1)$ has an inverse modulo $p$,
**Fermat’s Theorem: Reducing Exponents.**

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

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

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

$S$ contains representative of $\{1, \ldots, p-1\}$ modulo $p$.

\[ (a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}, \]

Since multiplication is commutative.

\[ a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \pmod{p}. \]

Each of $2, \ldots (p-1)$ has an inverse modulo $p$, solve to get...
Fermat’s Theorem: Reducing Exponents.

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

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

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

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

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p - 1)) \equiv 1 \cdot 2 \cdots (p - 1) \pmod{p},$$

Since multiplication is commutative.

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

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

$$a^{(p-1)} \equiv 1 \pmod{p}.$$
Fermat’s Theorem: Reducing Exponents.

**Fermat’s Little Theorem:** For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),
\[
a^{p-1} \equiv 1 \pmod{p}.
\]

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

All different modulo \( p \) since \( a \) has an inverse modulo \( p \).

\( S \) contains representative of \( \{1, \ldots, p-1\} \) modulo \( p \).

\[
(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},
\]

Since multiplication is commutative.

\[
a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \pmod{p}.
\]

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

\[
a^{(p-1)} \equiv 1 \pmod{p}.
\]
Fermat and Exponent reducing.

**Fermat’s Little Theorem:** For prime $p$, and $a \not\equiv 0 \pmod{p}$,
Fermat and Exponent reducing.

**Fermat’s Little Theorem**: For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),

\[ a^{p-1} \equiv 1 \pmod{p}. \]
Fermat and Exponent reducing.

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

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

What is $2^{101} \pmod{7}$?
Fermat and Exponent reducing.

**Fermat’s Little Theorem:** For prime \( p \), and \( a \not\equiv 0 \pmod{p} \),

\[ a^{p-1} \equiv 1 \pmod{p}. \]

What is \( 2^{101} \pmod{7} \)?

**Wrong:** \( 2^{101} = 2^{7 	imes 14 + 3} = 2^3 \pmod{7} \)

**Correct:** \( 2^{101} = 2^{100+1} = (2^{100}) \cdot 2 = 1^2 \cdot 2 = 2 \pmod{7} \)
Fermat and Exponent reducing.

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

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

What is $2^{101} \pmod{7}$?

Wrong: $2^{101} = 2^{7*14+3} = 2^3 \pmod{7}$

Fermat: 2 is relatively prime to 7.  \[\implies 2^6 = 1 \pmod{7}.\]
Fermat and Exponent reducing.

Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

What is $2^{101} \pmod{7}$?

Wrong: $2^{101} = 2^{7\times14+3} = 2^3 \pmod{7}$

Fermat: 2 is relatively prime to 7. $\implies 2^6 = 1 \pmod{7}$.

Correct: $2^{101} = 2^{6\times16+5} = 2^5 = 32 = 4 \pmod{7}$. 
Fermat and Exponent reducing.

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

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

What is $2^{101} \pmod{7}$?

Wrong: $2^{101} = 2^{7 \times 14 + 3} = 2^3 \pmod{7}$

Fermat: 2 is relatively prime to 7. \implies 2^6 \equiv 1 \pmod{7}.

Correct: $2^{101} = 2^{6 \times 16 + 5} = 2^5 = 32 = 4 \pmod{7}$.

For a prime modulus, we can reduce exponents modulo $p - 1$!
Lecture in a minute.

Euclid’s Alg:
\[ \text{gcd}(x, y) = \text{gcd}(y, x \mod y) \]
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \text{gcd}(x, y) \).
Idea: compute \( a, b \) recursively (euclid), or iteratively.

Inverse:
\[ ax + by = ax = \text{gcd}(x, y) \mod y. \]
If \( \text{gcd}(x, y) = 1 \), we have \( ax = 1 \mod y \rightarrow a = x - 1 \mod y \).

Chinese Remainder Theorem:
If \( \text{gcd}(n, m) = 1 \), \( x = a \mod n \), \( x = b \mod m \) unique sol.
Proof: Find \( u = 1 \mod n \), \( u = 0 \mod m \), and \( v = 0 \mod n \), \( v = 1 \mod m \).
Then:
\[ x = au + bv = a \mod n \ldots \]
\( u = m(m - 1)(\mod n) \) works!

Fermat: Prime \( p \),
\[ a^p - 1 = 1 \mod p. \]
Proof Idea:
\( f(x) = a(x) \mod p \): bijection on \( S = \{1, \ldots, p-1\} \).

Product of elts == for range/domain:
\[ a^{p-1} \text{ factor in range.} \]
Lecture in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Lecture in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$
Fast cuz value drops by a factor of two every two recursive calls.
Lecture in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$. 

Chinese Remainder Theorem:
If $gcd(x, y) = 1$, we have $ax = 1 \mod y$.

Fermat: Prime $p$, $a^p - 1 = 1 \mod p$. 
Proof Idea: $f(x) = a^x (\mod p)$ is a bijection on $S = \{1, \ldots, p-1\}$. 
Product of elts == for range/domain: $a^{p-1}$ factor in range.
Lecture in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$

Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.

Idea: compute $a, b$ recursively (euclid), or iteratively.
Lecture in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
Idea: compute \( a, b \) recursively (euclid), or iteratively.
Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
Lecture in a minute.

Euclid’s Alg: \[ \text{gcd}(x, y) = \text{gcd}(y, x \mod y) \]
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \text{gcd}(x, y) \).

Idea: compute \( a, b \) recursively (euclid), or iteratively.

Inverse: \( ax + by = ax = \text{gcd}(x, y) \mod y \).
If \( \text{gcd}(x, y) = 1 \), we have \( ax = 1 \mod y \)
Lecture in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.
   Idea: compute $a, b$ recursively (euclid), or iteratively.
Inverse: $ax + by = ax = gcd(x, y) \mod y$.
   If $gcd(x, y) = 1$, we have $ax = 1 \mod y$
   $\rightarrow a = x^{-1} \mod y$. 
Lecture in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$
  Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.
  Idea: compute $a, b$ recursively (euclid), or iteratively.
  Inverse: $ax + by = ax = gcd(x, y) \mod y$.
  If $gcd(x, y) = 1$, we have $ax = 1 \mod y$
  $\rightarrow a = x^{-1} mody$. 
Lecture in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
Idea: compute \( a, b \) recursively (euclid), or iteratively.
Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
\( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
If \( \gcd(n, m) = 1 \), \( x = a \mod n \), \( x = b \mod m \) unique sol.
Lecture in a minute.

Euclid’s Alg: \( \text{gcd}(x, y) = \text{gcd}(y, x \mod y) \)

Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \text{gcd}(x, y) \).

Idea: compute \( a, b \) recursively (euclid), or iteratively.

Inverse: \( ax + by = ax = \text{gcd}(x, y) \mod y \).

If \( \text{gcd}(x, y) = 1 \), we have \( ax = 1 \mod y \)

\( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:

If \( \text{gcd}(n, m) = 1 \), \( x = a \mod n \), \( x = b \mod m \) unique sol.

Proof: Find \( u = 1 \mod n \), \( u = 0 \mod m \),
Lecture in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)

Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).

Idea: compute \( a, b \) recursively (euclid), or iteratively.

Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).

If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)

\[ a = x^{-1} \mod y. \]

Chinese Remainder Theorem:

If \( \gcd(n, m) = 1 \), \( x = a \) (mod \( n \)), \( x = b \) (mod \( m \)) unique sol.

Proof: Find \( u = 1 \) (mod \( n \)), \( u = 0 \) (mod \( m \)),

and \( v = 0 \) (mod \( n \)), \( v = 1 \) (mod \( m \)).
Lecture in a minute.

Euclid’s Alg: $gcd(x, y) = gcd(y, x \mod y)$
   Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find $a, b$ where $ax + by = gcd(x, y)$.
   Idea: compute $a, b$ recursively (euclid), or iteratively.
   Inverse: $ax + by = ax = gcd(x, y) \mod y$.
   If $gcd(x, y) = 1$, we have $ax = 1 \mod y$
   $\rightarrow a = x^{-1} \mod y$.

Chinese Remainder Theorem:
   If $gcd(n, m) = 1$, $x = a \pmod{n}, x = b \pmod{m}$ unique sol.
   Proof: Find $u = 1 \pmod{n}, u = 0 \pmod{m}$,
   and $v = 0 \pmod{n}, v = 1 \pmod{m}$.
   Then: $x = au + bv = a \pmod{n}...$
Lecture in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
   Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
   Idea: compute \( a, b \) recursively (euclid), or iteratively.
   Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
      If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
      \( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
   If \( \gcd(n, m) = 1 \), \( x = a \mod n \), \( x = b \mod m \) unique sol.

   Proof: Find \( u = 1 \mod n \), \( u = 0 \mod m \),
      and \( v = 0 \mod n \), \( v = 1 \mod m \).
      Then: \( x = au + bv = a \mod n \)...
      \( u = m(m^{-1} \mod n)) \mod n \) works!

Lecture in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)

Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).

Idea: compute \( a, b \) recursively (euclid), or iteratively.

Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).

If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)

\( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:

If \( \gcd(n, m) = 1 \), \( x = a \mod n \), \( x = b \mod m \) unique sol.

Proof: Find \( u = 1 \mod n \), \( u = 0 \mod m \),

and \( v = 0 \mod n \), \( v = 1 \mod m \).

Then: \( x = au + bv = a \mod n \)...

\( u = m(m^{-1} \mod n)) \mod n \) works!

Fermat: Prime \( p \), \( a^{p-1} = 1 \mod p \).
Lecture in a minute.

Euclid’s Alg: \( \gcd(x, y) = \gcd(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \gcd(x, y) \).
Idea: compute \( a, b \) recursively (euclid), or iteratively.
Inverse: \( ax + by = ax = \gcd(x, y) \mod y \).
If \( \gcd(x, y) = 1 \), we have \( ax = 1 \mod y \)
\( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
If \( \gcd(n, m) = 1 \), \( x = a \pmod{n} \), \( x = b \pmod{m} \) unique sol.

Proof: Find \( u = 1 \pmod{n} \), \( u = 0 \pmod{m} \),
and \( v = 0 \pmod{n} \), \( v = 1 \pmod{m} \).
Then: \( x = au + bv = a \pmod{n} \)... \( u = m(m^{-1} \pmod{n} ) \pmod{n} \) works!

Fermat: Prime \( p \), \( a^{p-1} = 1 \pmod{p} \).
Proof Idea: \( f(x) = a(x) \pmod{p} \): bijection on \( S = \{1, \ldots, p-1\} \).
Lecture in a minute.

Euclid’s Alg: \( \text{gcd}(x, y) = \text{gcd}(y, x \mod y) \)
Fast cuz value drops by a factor of two every two recursive calls.

Extended Euclid: Find \( a, b \) where \( ax + by = \text{gcd}(x, y) \).
   Idea: compute \( a, b \) recursively (euclid), or iteratively.
   Inverse: \( ax + by = ax = \text{gcd}(x, y) \mod y \).
   If \( \text{gcd}(x, y) = 1 \), we have \( ax = 1 \mod y \)
   \( \rightarrow a = x^{-1} \mod y \).

Chinese Remainder Theorem:
If \( \text{gcd}(n, m) = 1 \), \( x = a \mod n, x = b \mod m \) unique sol.
   Proof: Find \( u = 1 \mod n, u = 0 \mod m \),
   and \( v = 0 \mod n, v = 1 \mod m \).
   Then: \( x = au + bv = a \mod n \)...
   \( u = m(m^{-1} \mod n)) \mod n \) works!

Fermat: Prime \( p \), \( a^{p-1} = 1 \mod p \).
   Proof Idea: \( f(x) = a(x) \mod p \): bijection on \( S = \{1, \ldots, p-1\} \).
   Product of elts == for range/domain: \( a^{p-1} \) factor in range.