Today

Homework/No-Homework option.

Today

Homework/No-Homework option.
  Deadline is set for after Midterm.
  Data on Thursday.

Today

Homework/No-Homework option.

- Deadline is set for after Midterm.
- Data on Thursday.

Finish Euclid.

Today

Homework/No-Homework option.
  Deadline is set for after Midterm.
  Data on Thursday.

Finish Euclid.

Bijection/CRT/Isomorphism.

Today

Homework/No-Homework option.
  Deadline is set for after Midterm.
  Data on Thursday.

Finish Euclid.

Bijection/CRT/Isomorphism.

Fermat's Little Theorem.

Today

Homework/No-Homework option.
- Deadline is set for after Midterm.
- Data on Thursday.

Finish Euclid.

Bijection/CRT/Isomorphism.

Fermat's Little Theorem.

Quick review

Review runtime proof.

Runtime Proof.

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

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

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.

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.

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.

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.

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: $y < x/2$, first argument is y
$\implies$ true in one recursive call;

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

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: $y < x/2$, first argument is y
$\Longrightarrow$ true in one recursive call;

Case 2: Will show "$y \geq x/2$" $\Longrightarrow$ "$mod(x, y) \leq x/2$."

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: $y < x/2$, first argument is y
$\implies$ 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.

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;

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

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;

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

$\text{mod}\,(x, y)$ is second argument in next recursive call,
and becomes the first argument in the next one.
When $y \geq x/2$, then

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

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;

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

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

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

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;

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

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

$$\text{mod}\,(x, y) = x - y\lfloor \frac{x}{y} \rfloor = x - y \leq x - x/2$$

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: $y < x/2$, first argument is y
$\implies$ 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 $(x, y) = x - y\lfloor \frac{x}{y} \rfloor = x - y \leq x - x/2 = x/2$

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: $y < x/2$, first argument is y
$\implies$ 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

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

$\text{mod}\,(x, y) = x - y\lfloor \frac{x}{y} \rfloor = x - y \leq x - x/2 = x/2$

$\square$

Poll

Mark correct answers.
Note: Mod(x,y) is the remainder of x divided by y and $y < x$.

Poll

Mark correct answers.
Note: Mod(x,y) is the remainder of x divided by y and $y < x$.

(A) $\mod(x, y) < y$
(B) If euclid(x,y) calls euclid(u,v) calls euclid (a,b) then $a <= x/2$.
(C) euclid(x,y) calls euclid (u,v) means u = y.
(D) if $y > x/2$, $\mod(x, y) = (x - y)$
(E) if $y > x/2$, $\mod(x, y) < x/2$

Poll

Mark correct answers.
Note: Mod(x,y) is the remainder of x divided by y and $y < x$.

(A) $\mod(x, y) < y$
(B) If euclid(x,y) calls euclid(u,v) calls euclid (a,b) then $a <= x/2$.
(C) euclid(x,y) calls euclid (u,v) means u = y.
(D) if $y > x/2$, $\mod(x, y) = (x - y)$
(E) if $y > x/2$, $\mod(x, y) < x/2$

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.

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

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

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
$ax + by = d$ where $d = gcd(x, y)$.

Extended GCD

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

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

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

Extended GCD

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

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

What is multiplicative inverse of x modulo m?

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

Extended GCD

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

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

What is multiplicative inverse of x modulo m?

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

$$ax + bm = 1$$

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

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

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

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

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

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

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

What is multiplicative inverse of x modulo m?

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

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

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

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

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

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

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

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

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

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

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

$$a = 3 \text{ and } b = -1.$$

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

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

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

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

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

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

The multiplicative inverse of $12 \pmod{35}$ is 3.

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

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

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

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

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

Check: 3(12)

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

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

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

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

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

Check: $3(12) = 36$

Extended GCD

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

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

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

What is multiplicative inverse of x modulo m?

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

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

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

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

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

Check: $3(12) = 36 = 1 \pmod{35}$.

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

```
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*..?

```
gcd(35,12)
  gcd(12, 11)  ;;  gcd(12, 35%12)
    gcd(11, 1)  ;;  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*..?

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

How did gcd get 11 from 35 and 12?

Make *d* out of 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?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

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
```

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

How does gcd get 1 from 12 and 11?

Make *d* out of 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?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

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*..?

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

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

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

Algorithm finally returns 1.

Make d out of 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?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

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..?

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

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

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

Algorithm finally returns 1.

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

Get 1 from 12 and 11.

Make *d* out of 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?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

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

Algorithm finally returns 1.

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

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

Make *d* out of 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?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

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

Algorithm finally returns 1.

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

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

Make d out of 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?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

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..?

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

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

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

Algorithm finally returns 1.

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

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

Make *d* out of 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?
$35 - \lfloor \frac{35}{12} \rfloor 12 = 35 - (2)12 = 11$

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

Algorithm finally returns 1.

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

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

Extended GCD Algorithm.

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

Extended GCD Algorithm.

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

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

Extended GCD Algorithm.

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

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

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

Extended GCD Algorithm.

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

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

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

Extended GCD Algorithm.

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

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

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

Extended GCD Algorithm.

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

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

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

Extended GCD Algorithm.

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

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

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

Extended GCD Algorithm.

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

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

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

Extended GCD Algorithm.

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

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

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

Extended GCD Algorithm.

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

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

```
    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.

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

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

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

Extended GCD Algorithm.

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

Extended GCD Algorithm.

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

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

$$d = ax + by.$$

Correctness.

Proof: Strong Induction.[1]

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

Correctness.

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

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

Correctness.

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

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

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

Correctness.

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

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

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

Correctness.

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

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

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

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

Correctness.

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

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

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

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

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

Correctness.

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

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

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

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

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

Correctness.

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

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

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

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

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

Correctness.

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

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

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

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

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

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

Correctness.

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

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

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

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

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

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

Review Proof: step.

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

Review Proof: step.

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

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

Review Proof: step.

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

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

Review Proof: step.

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

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

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

Hand Calculation Method for Inverses.

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

Hand Calculation Method for Inverses.

Example: gcd$(7,60) = 1$.
egcd(7,60).

Hand Calculation Method for Inverses.

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

$$7(0) + 60(1) = 60$$

Hand Calculation Method for Inverses.

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

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \end{aligned}$$

Hand Calculation Method for Inverses.

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

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \end{aligned}$$

Hand Calculation Method for Inverses.

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

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \\ 7(-8) + 60(1) &= 4 \\ 7(9) + 60(-1) &= 3 \end{aligned}$$

Hand Calculation Method for Inverses.

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

$$
\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}
$$

Hand Calculation Method for Inverses.

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

$$\begin{aligned}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{aligned}$$

Hand Calculation Method for Inverses.

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

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \\ 7(9)+60(-1) &= 3 \\ 7(-17)+60(2) &= 1 \end{aligned}$$

Confirm:

Hand Calculation Method for Inverses.

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

$$\begin{aligned} 7(0) + 60(1) &= 60 \\ 7(1) + 60(0) &= 7 \\ 7(-8) + 60(1) &= 4 \\ 7(9) + 60(-1) &= 3 \\ 7(-17) + 60(2) &= 1 \end{aligned}$$

Confirm: $-119 + 120 = 1$

Hand Calculation Method for Inverses.

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

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \\ 7(9)+60(-1) &= 3 \\ 7(-17)+60(2) &= 1 \end{aligned}$$

Confirm: $-119+120=1$

Note: an “iterative” version of the e-gcd algorithm.

Fundamental Theorem of Arithmetic.

Thm: Every natural number can be written as the product of primes.

Fundamental Theorem of Arithmetic.

Thm: Every natural number can be written as the product of primes.

Proof: n is either prime (base cases)
or $n = a \times b$ and a and b can be written as product of primes.

Fundamental Theorem of Arithmetic.

Thm: Every natural number can be written as the product of primes.

Proof: n is either prime (base cases)
or $n = a \times b$ and a and b can be written as product of primes.

Thm: The prime factorization of n is unique up to reordering.

Fundamental Theorem of Arithmetic.

Thm: Every natural number can be written as the product of primes.

Proof: n is either prime (base cases)
or $n = a \times b$ and a and b can be written as product of primes.

Thm: The prime factorization of n is unique up to reordering.

Fundamental Theorem of Arithmetic: Every natural number can be written as the a unique (up to reordering) product of primes.

Fundamental Theorem of Arithmetic.

Thm: Every natural number can be written as the product of primes.

Proof: n is either prime (base cases)
or $n = a \times b$ and a and b can be written as product of primes.

Thm: The prime factorization of n is unique up to reordering.

Fundamental Theorem of Arithmetic: Every natural number can be written as the a unique (up to reordering) product of primes.

Generalization: things with a “division algorithm”.

Fundamental Theorem of Arithmetic.

Thm: Every natural number can be written as the product of primes.

Proof: n is either prime (base cases)
or $n = a \times b$ and a and b can be written as product of primes.

Thm: The prime factorization of n is unique up to reordering.

Fundamental Theorem of Arithmetic: Every natural number can be written as the a unique (up to reordering) product of primes.

Generalization: things with a “division algorithm”.

One example: polynomial division.

No shared common factors, and products.

Claim: For $x, y, z \in \mathbb{Z}^+$ with $gcd(x, y) = 1$ and $x|yz$ then $x|z$.

No shared common factors, and products.

Claim: For $x, y, z \in \mathbb{Z}^+$ with $gcd(x, y) = 1$ and $x|yz$ then $x|z$.

Idea: x doesn't share common factors with y
so it must divide z.

No shared common factors, and products.

Claim: For $x, y, z \in \mathbb{Z}^+$ with $gcd(x, y) = 1$ and $x|yz$ then $x|z$.

Idea: x doesn't share common factors with y
so it must divide z.

Euclid: $1 = ax + by$.

No shared common factors, and products.

Claim: For $x, y, z \in \mathbb{Z}^+$ with $gcd(x, y) = 1$ and $x|yz$ then $x|z$.

Idea: x doesn't share common factors with y
so it must divide z.

Euclid: $1 = ax + by$.

Observe: $x|axz$ and $x|byz$ (since $x|yz$), and x divides the sum.
$\implies x|axz + byz$
And $axz + byz = z$, thus $x|z$.

$\square$

Wrap-up

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

Wrap-up

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

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

Wrap-up

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

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

$2^{n/2}$

Wrap-up

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

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

$2^{n/2}$

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

Wrap-up

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

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

$2^{n/2}$

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

Wrap-up

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

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

$2^{n/2}$

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

Wrap-up

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

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

$2^{n/2}$

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

Wrap-up

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

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

$2^{n/2}$

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

Internet Security.

Wrap-up

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

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

$2^{n/2}$

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

Internet Security.
Public Key Cryptography: 512 digits.

Wrap-up

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

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

$2^{n/2}$

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

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

Wrap-up

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

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

$2^{n/2}$

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

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.
$(1000000000000000000000000000000000000000000000)^5$ divisions.

Wrap-up

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

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

$2^{n/2}$

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

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.
$(100000000000000000000000000000000000000000000000)^5$ divisions.

Internet Security:

Wrap-up

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

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

$2^{n/2}$

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

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.
$(100000000000000000000000000000000000000000000)^5$ divisions.

Internet Security: Soon.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \ldots q_l$, then $p = q_j$ for some j.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \dots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1, \implies p_1 | q_1 \cdots q_{l-1}$ by Claim.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \ldots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1, \implies p_1 | q_1 \cdots q_{l-1}$ by Claim.

If $gcd(p, q_l) = d$, then d is a common factor.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \dots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1$, $\implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \dots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1$, $\implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.
Thus, $p = q_l = d$.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \ldots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1$, $\implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.
Thus, $p = q_l = d$.

End proof of fact.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.
$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \ldots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1$, $\implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.
Thus, $p = q_l = d$.
End proof of fact.

Proof by induction.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \dots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1, \implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.
Thus, $p = q_l = d$.

End proof of fact.

Proof by induction.
Base case: If $l = 1$, $p_1 \cdots p_k = q_1$.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.
$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \ldots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1$, $\implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.
Thus, $p = q_l = d$.
End proof of fact.

Proof by induction.
Base case: If $l = 1$, $p_1 \cdots p_k = q_1$.
But if q_1 is prime, only prime factor is q_1 and $p_1 = q_1$ and $l = k = 1$.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \dots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1, \implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.
Thus, $p = q_l = d$.
End proof of fact.

Proof by induction.
Base case: If $l = 1$, $p_1 \cdots p_k = q_1$.
But if q_1 is prime, only prime factor is q_1 and $p_1 = q_1$ and $l = k = 1$.

Induction step:

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \ldots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1, \implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.
Thus, $p = q_l = d$.

End proof of fact.

Proof by induction.
Base case: If $l = 1$, $p_1 \cdots p_k = q_1$.
But if q_1 is prime, only prime factor is q_1 and $p_1 = q_1$ and $l = k = 1$.

Induction step: From Fact: $p_1 = q_j$ for some j.

Fundamental Theorem of Arithmetic:uniqueness

Thm: The prime factorization of n is unique up to reordering.

Assume not.

$n = p_1 \cdot p_2 \cdots p_k$ and $n = q_1 \cdot q_2 \cdots q_l$.

Fact: If $p | q_1 \dots q_l$, then $p = q_j$ for some j.

If $gcd(p, q_l) = 1$, $\implies p_1 | q_1 \cdots q_{l-1}$ by Claim.
If $gcd(p, q_l) = d$, then d is a common factor.
If both prime, both only have 1 and themselves as factors.
Thus, $p = q_l = d$.

End proof of fact.

Proof by induction.
Base case: If $l = 1$, $p_1 \cdots p_k = q_1$.
But if q_1 is prime, only prime factor is q_1 and $p_1 = q_1$ and $l = k = 1$.

Induction step: From Fact: $p_1 = q_j$ for some j.
$n/p_1 = p_2 \dots p_k$ and $n/q_j = \prod_{i \neq j} q_i$.
These two expressions are the same up to reordering by induction.
And p_1 is matched to q_j.

□

Lots of Mods

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

Lots of Mods

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

What is $x \pmod{35}$?

Lots of Mods

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

What is $x \pmod{35}$?

Let's try 5.

Lots of Mods

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

What is $x \pmod{35}$?

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

Lots of Mods

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

What is $x \pmod{35}$?

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

Lots of Mods

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

What is $x \pmod{35}$?

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

Lots of Mods

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

What is $x \pmod{35}$?

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

Lots of Mods

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

What is $x \pmod{35}$?

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

If $x = 5 \pmod 7$

Lots of Mods

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

What is $x \pmod{35}$?

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

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

Lots of Mods

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

What is $x \pmod{35}$?

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

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

Lots of Mods

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

What is $x \pmod{35}$?

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

If $x = 5 \pmod 7$
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

$x = 5 \pmod 7$ 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.

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... 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.

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 (solution exists):

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 (solution exists):
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 (solution exists):
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 (solution exists):
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\qquad$ $u = 1 \pmod{m}$

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

Simple Chinese Remainder Theorem.

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

Simple Chinese Remainder Theorem.

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 (solution exists):
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.

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 (solution exists):
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$
Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$
Let $x = au + bv$.
$x = a \pmod{m}$

Simple Chinese Remainder Theorem.

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 (solution exists):
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 $au = a \pmod{m}$

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 (solution exists):
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 $au = a \pmod{m}$

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 (solution exists):
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad$ $u = 1 \pmod{m}$
Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad$ $v = 0 \pmod{m}$
Let $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$

Simple Chinese Remainder Theorem.

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 (solution exists):
Consider $u = n(n^{-1} \pmod{m})$.
$u = 0 \pmod{n}$ $\quad u = 1 \pmod{m}$
Consider $v = m(m^{-1} \pmod{n})$.
$v = 1 \pmod{n}$ $\quad v = 0 \pmod{m}$
Let $x = au + bv$.
$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
$x = b \pmod{n}$ since $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

Simple Chinese Remainder Theorem.

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 (solution exists):
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 $bv = b \pmod{n}$
This shows there is a solution. □

Simple Chinese Remainder Theorem.

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

Simple Chinese Remainder Theorem.

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

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
If not, two solutions, x and y.

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
If not, two solutions, x and y.

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
If not, two solutions, x and y.

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

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
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

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
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
$\gcd(m,n)=1 \implies$ no common primes in factorization m and n

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
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
$\gcd(m,n) = 1 \implies$ no common primes in factorization m and n
$\implies mn|(x-y)$

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
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
$\gcd(m,n) = 1 \implies$ no common primes in factorization m and n
$\implies mn|(x-y)$
$\implies x-y \geq mn$

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
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
$\gcd(m,n) = 1 \implies$ no common primes in factorization m and n
$\implies mn|(x-y)$
$\implies x-y \geq mn \implies x,y \notin \{0,\ldots,mn-1\}$.

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
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
$\gcd(m,n) = 1 \implies$ no common primes in factorization m and n
$\implies mn|(x-y)$
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn-1\}$.
Thus, only one solution modulo mn.

Simple Chinese Remainder Theorem.

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

Proof (uniqueness):
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
$\gcd(m,n) = 1 \implies$ no common primes in factorization m and n
$\implies mn | (x-y)$
$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn-1\}$.
Thus, only one solution modulo mn. □

Poll.

My love is won,
Zero and one.
Nothing and nothing done.

Poll.

My love is won,
Zero and one.
Nothing and nothing done.

What is the rhyme saying?

Poll.

My love is won,
Zero and one.
Nothing and nothing done.

What is the rhyme saying?

(A) Multiplying by 1, gives back number. (Does nothing.)
(B) Adding 0 gives back number. (Does nothing.)
(C) Rao has gone mad.
(D) Multiplying by 0, gives 0.
(E) Adding one does, not too much.

Poll.

My love is won,
Zero and one.
Nothing and nothing done.

What is the rhyme saying?

(A) Multiplying by 1, gives back number. (Does nothing.)
(B) Adding 0 gives back number. (Does nothing.)
(C) Rao has gone mad.
(D) Multiplying by 0, gives 0.
(E) Adding one does, not too much.

All are (maybe) correct.

Poll.

My love is won,
Zero and one.
Nothing and nothing done.

What is the rhyme saying?

(A) Multiplying by 1, gives back number. (Does nothing.)
(B) Adding 0 gives back number. (Does nothing.)
(C) Rao has gone mad.
(D) Multiplying by 0, gives 0.
(E) Adding one does, not too much.

All are (maybe) correct.
(E) doesn't have to do with the rhyme.

Poll.

My love is won,
Zero and one.
Nothing and nothing done.

What is the rhyme saying?

(A) Multiplying by 1, gives back number. (Does nothing.)
(B) Adding 0 gives back number. (Does nothing.)
(C) Rao has gone mad.
(D) Multiplying by 0, gives 0.
(E) Adding one does, not too much.

All are (maybe) correct.
(E) doesn't have to do with the rhyme.
(C) Recall Polonius:

Poll.

My love is won,
Zero and one.
Nothing and nothing done.

What is the rhyme saying?

(A) Multiplying by 1, gives back number. (Does nothing.)
(B) Adding 0 gives back number. (Does nothing.)
(C) Rao has gone mad.
(D) Multiplying by 0, gives 0.
(E) Adding one does, not too much.

All are (maybe) correct.
(E) doesn't have to do with the rhyme.
(C) Recall Polonius:
"Though this be madness, yet there is method in 't."

CRT:isomorphism.

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

CRT:isomorphism.

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

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

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 $y = d \mod n$

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 $b + d \mod n$.

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:

Fermat's Theorem: Reducing Exponents.

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

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

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

Fermat's Theorem: Reducing Exponents.

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

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

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

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

Fermat's Theorem: Reducing Exponents.

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

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

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

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

Fermat's Theorem: Reducing Exponents.

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

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

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

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

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

Fermat's Theorem: Reducing Exponents.

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

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

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

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

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

Since multiplication is commutative.

Fermat's Theorem: Reducing Exponents.

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

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

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

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

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

Since multiplication is commutative.

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

Fermat's Theorem: Reducing Exponents.

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

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

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

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

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

Since multiplication is commutative.

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

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

Fermat's Theorem: Reducing Exponents.

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

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

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

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

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

Since multiplication is commutative.

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

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

Fermat's Theorem: Reducing Exponents.

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

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

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

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

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

Since multiplication is commutative.

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

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

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

Fermat's Theorem: Reducing Exponents.

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

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

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

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

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

Since multiplication is commutative.

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

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

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

□

Poll

Which was used in Fermat's theorem proof?

Poll

Which was used in Fermat's theorem proof?

(A) The mapping $f(x) = ax \mod p$ is a bijection.
(B) Multiplying a number by 1, gives the number.
(C) All nonzero numbers mod p, have an inverse.
(D) Multiplying a number by 0 gives 0.
(E) Mutliplying elements of sets A and B together is the same if $A = B$.

Poll

Which was used in Fermat's theorem proof?

(A) The mapping $f(x) = ax \mod p$ is a bijection.
(B) Multiplying a number by 1, gives the number.
(C) All nonzero numbers mod p, have an inverse.
(D) Multiplying a number by 0 gives 0.
(E) Mutliplying elements of sets A and B together is the same if $A = B$.

(A), (C), and (E)

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: $2^{101} = 2^{7*14+3} = 2^3 \pmod{7}$

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: $2^{101} = 2^{7*14+3} = 2^3 \pmod{7}$

Fermat: 2 is relatively prime to 7. $\implies 2^6 = 1 \pmod{7}$.

Correct: $2^{101} = 2^{6*16+5} = 2^5 = 32 = 4 \pmod{7}$.

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: $2^{101} = 2^{7*14+3} = 2^3 \pmod{7}$

Fermat: 2 is relatively prime to 7. $\Longrightarrow 2^6 = 1 \pmod{7}$.

Correct: $2^{101} = 2^{6*16+5} = 2^5 = 32 = 4 \pmod{7}$.

For a prime modulus, we can reduce exponents modulo $p-1$!

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

Chinese Remainder Theorem:

If $gcd(n, m) = 1$, $x = a \pmod{n}, x = b \pmod{m}$ unique sol.

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

Chinese Remainder Theorem:

If $gcd(n, m) = 1$, $x = a \pmod{n}, x = b \pmod{m}$ unique sol.

Proof: Find $u = 1 \pmod{n}$, $u = 0 \pmod{m}$,

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

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 $v = 0 \pmod{n}$, $v = 1 \pmod{m}$.

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

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.

Extended Euclid: Find a,b where $ax+by=gcd(x,y)$.

Idea: compute a,b recursively (euclid), or iteratively.

Inverse: $ax+by = ax = gcd(x,y) \pmod{y}$.

If $gcd(x,y)=1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

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!

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

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

Lecture in a minute.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

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.

Extended Euclid: Find a, b where $ax + by = gcd(x, y)$.

Idea: compute a, b recursively (euclid), or iteratively.

Inverse: $ax + by = ax = gcd(x, y) \pmod{y}$.

If $gcd(x, y) = 1$, we have $ax = 1 \pmod{y}$

$\rightarrow a = x^{-1} \pmod{y}$.

Fundamental Theorem of Algebra: Unique prime factorization of any natural number.

Claim: any prime that divides a number n, divides a number in any factorization of n.

From Extended Euclid.

Induction.

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 $S = \{1, \ldots, p-1\}$.