

Outline

Greatest Common Divisor

Euclid's Algorithm

Extended Euclid's Algorithm

Greatest Common Divisor

Definition

The **greatest common divisor**, $\gcd(a, b)$, of integers a and b (not both equal to zero) is the largest integer that divides both a and b

Greatest Common Divisor

Definition

The **greatest common divisor**, $\gcd(a, b)$, of integers a and b (not both equal to zero) is the largest integer that divides both a and b .

Examples

$$\gcd(24, 16) = 8, \gcd(9, 17) = 1,$$
$$\gcd(239, 0) = 239$$

Greatest Common Divisor

Definition

The **greatest common divisor**, $\gcd(a, b)$, of integers a and b (not both equal to zero) is the largest integer that divides both a and b

Examples

$$\gcd(24, 16) = 8, \gcd(9, 17) = 1,$$
$$\gcd(239, 0) = 239$$

Convention

We assume that a and b are non-negative

First Application: Computing Inverses

Given integers a and n , how to find an integer k such that $ak \equiv 1 \pmod{n}$? Basic primitive in modern crypto protocols, used billions times per day



Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59}$$

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59}$$

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443}$$

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443} = \frac{2}{3}$$

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443} = \frac{2}{3}$$

```
from fractions import Fraction  
print(Fraction(31, 177) + Fraction(29, 59))
```

Second Application: Computing Fractions

$$\frac{31}{177} + \frac{29}{59} = \frac{31 \times 59 + 177 \times 29}{177 \times 59} = \frac{6962}{10443} = \frac{2}{3}$$

```
from fractions import Fraction  
print(Fraction(31, 177) + Fraction(29, 59))
```

2/3

Naive Algorithm

To find the greatest common divisor, simply try all numbers and select the largest one

Naive Algorithm: Code

```
def gcd(a, b):
    assert a >= 0 and b >= 0 and a + b > 0

    if a == 0 or b == 0:
        return max(a, b)

    for d in range(min(a, b), 0, -1):
        if a % d == 0 and b % d == 0:
            return d

    return 1
```

Naive Algorithm: Code

```
def gcd(a, b):
    assert a >= 0 and b >= 0 and a + b > 0

    if a == 0 or b == 0:
        return max(a, b)

    for d in range(min(a, b), 0, -1):
        if a % d == 0 and b % d == 0:
            return d

    return 1

print(gcd(24, 16))
```

Naive Algorithm: Analysis

- If $\gcd(a, b) = 1$, the algorithm will perform $\min\{a, b\}$ divisions

Naive Algorithm: Analysis

- If $\gcd(a, b) = 1$, the algorithm will perform $\min\{a, b\}$ divisions
- Modern laptops perform roughly one billion (10^9) operations per second

Naive Algorithm: Analysis

- If $\gcd(a, b) = 1$, the algorithm will perform $\min\{a, b\}$ divisions
- Modern laptops perform roughly one billion (10^9) operations per second
- On your laptop, the call

```
print(gcd(790933790547, 1849639579327))
```

will take more than one minute

Supercomputer



If a and b consist of hundreds of digits (typical case for crypto protocols), even on a supercomputer with quadrillion operations per second this algorithm will run for more than thousand years

Next Parts

Euclid's algorithm:
efficient algorithm for
computing the greatest
common divisor of two
integers



Outline

Greatest Common Divisor

Euclid's Algorithm

Extended Euclid's Algorithm

Euclid's Lemma

Euclid's Lemma

d divides a and b , if and only if d divides $a - b$ and b .

Euclid's Lemma

Euclid's Lemma

d divides a and b , if and only if d divides $a - b$ and b .

Proof

\Rightarrow if $a = dp$ and $b = dq$, then
 $a - b = d(p - q)$

Euclid's Lemma

Euclid's Lemma

d divides a and b , if and only if d divides $a - b$ and b .

Proof

\Rightarrow if $a = dp$ and $b = dq$, then

$$a - b = d(p - q)$$

\Leftarrow if $a - b = dp$ and $b = dq$, then

$$a = d(p + q)$$



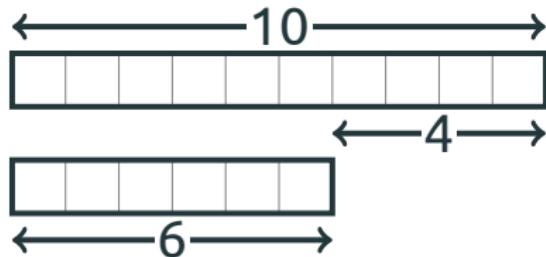
Euclid's Lemma: Pictorially

2 divides 10 and 6, hence 2 divides $4 = 10 - 6$



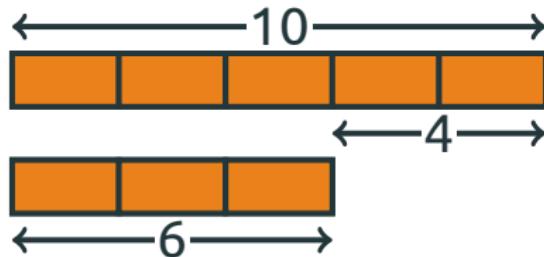
Euclid's Lemma: Pictorially

2 divides 10 and 6, hence 2 divides $4 = 10 - 6$



Euclid's Lemma: Pictorially

2 divides 10 and 6, hence 2 divides $4 = 10 - 6$



Algorithm

```
def gcd(a, b):
    assert a >= 0 and b >= 0 and a + b > 0

    while a > 0 and b > 0:
        if a >= b:
            a = a - b
        else:
            b = b - a

    return max(a, b)
```

Analysis

- On some inputs, works much faster than the previous algorithm

Analysis

- On some inputs, works much faster than the previous algorithm
- Still, there are inputs where this code is too slow:

```
gcd(790933790548, 7)
```

Analysis

- On some inputs, works much faster than the previous algorithm
- Still, there are inputs where this code is too slow:

```
gcd(790933790548, 7)
```

- Reason: the code will subtract 7 billions of times!

Analysis

- On some inputs, works much faster than the previous algorithm
- Still, there are inputs where this code is too slow:

```
gcd(790933790548, 7)
```

- Reason: the code will subtract 7 billions of times!
- Idea: what is left is the remainder modulo 7

Euclid's Algorithm

```
def gcd(a, b):
    assert a >= 0 and b >= 0 and a + b > 0

    while a > 0 and b > 0:
        if a >= b:
            a = a % b
        else:
            b = b % a

    return max(a, b)
```

Analysis

- Already quite fast: if a and b are 100 digits long, the number of iterations of the while loop is at most 660

Analysis

- Already quite fast: if a and b are 100 digits long, the number of iterations of the while loop is at most 660
- Each iteration is a division

Code with Logging

```
def gcd(a, b):
    assert a >= 0 and b >= 0 and a + b > 0

    while a > 0 and b > 0:
        print("gcd({} , {})=".format(a, b))
        if a >= b:
            a = a % b
        else:
            b = b % a

    print("gcd({} , {})=".format(a, b))
    return max(a, b)

print(gcd(790933790547, 1849639579327))
```

Code with Logging: Output

```
gcd(790933790547, 1849639579327)=  
gcd(790933790547, 267771998233)=  
gcd(255389794081, 267771998233)=  
gcd(255389794081, 12382204152)=  
gcd(7745711041, 12382204152)=  
gcd(7745711041, 4636493111)=  
gcd(3109217930, 4636493111)=  
gcd(3109217930, 1527275181)=  
gcd(54667568, 1527275181)=  
gcd(54667568, 51250845)=  
gcd(3416723, 51250845)=  
gcd(3416723, 0)=  
3416723
```

Analysis

- The numbers are getting shorter and shorter
- A more quantitative statement: at each iteration of the while loop the larger number drops by at least a factor of 2

Analysis

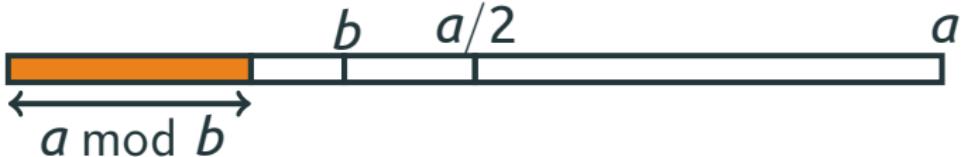
- The numbers are getting shorter and shorter
- A more quantitative statement: at each iteration of the while loop the larger number drops by at least a factor of 2

Lemma

Let $a \geq b > 0$. Then $(a \bmod b) < a/2$.

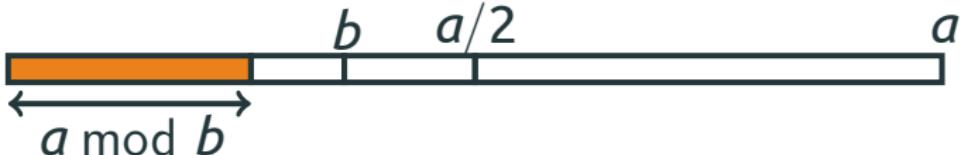
Proof

- If $b \leq a/2$, then $(a \text{ mod } b) < b \leq a/2$.

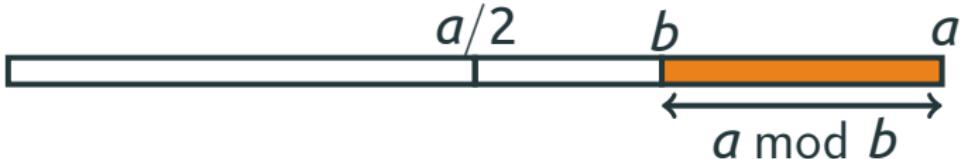


Proof

- If $b \leq a/2$, then $(a \text{ mod } b) < b \leq a/2$.



- If $b > a/2$, then $(a \text{ mod } b) = a - b < a/2$.



□

Final Analysis

- Hence, at each iteration, either a or b is dropped by at least a factor of 2

Final Analysis

- Hence, at each iteration, either a or b is dropped by at least a factor of 2
- Thus, the total number of iterations is at most $\log_2 a + \log_2 b$

Final Analysis

- Hence, at each iteration, either a or b is dropped by at least a factor of 2
- Thus, the total number of iterations is at most $\log_2 a + \log_2 b$
- If a consists of less than 5 000 decimal digits (i.e., $a < 10^{5\,000}$), then $\log_2 a < 16\,610$

Compact Code

```
def gcd(a, b):  
    assert a >= b and b >= 0 and a + b > 0  
    return gcd(b, a % b) if b > 0 else a
```

Outline

Greatest Common Divisor

Euclid's Algorithm

Extended Euclid's Algorithm

Certificate

- Somebody computed the greatest common divisor of a and b and wants to convince you that it is equal to d

Certificate

- Somebody computed the greatest common divisor of a and b and wants to convince you that it is equal to d
- You can check that d divides both a and b , but this only shows that d is a common divisor of a and b , but does not guarantees that is the greatest one

Certificate

- Somebody computed the greatest common divisor of a and b and wants to convince you that it is equal to d
- You can check that d divides both a and b , but this only shows that d is a common divisor of a and b , but does not guarantees that is the greatest one
- It turns out that it is enough to represent d as $ax + by$ (for integers x and y)!

Test

Lemma

If d divides a and b and $d = ax + by$ for integers x and y , then $d = \gcd(a, b)$.

Test

Lemma

If d divides a and b and $d = ax + by$ for integers x and y , then $d = \gcd(a, b)$.

Proof

- d is a common divisor of a and b , hence
 $d \leq \gcd(a, b)$

Test

Lemma

If d divides a and b and $d = ax + by$ for integers x and y , then $d = \gcd(a, b)$.

Proof

- d is a common divisor of a and b , hence $d \leq \gcd(a, b)$
- $\gcd(a, b)$ divides both a and b , hence it also divides $d = ax + by$, and hence $\gcd(a, b) \leq d$



Examples

- $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$

Examples

- $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
- $\gcd(7, 5) = 1 = 7 \cdot (-2) + 5 \cdot 3$

Examples

- $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
- $\gcd(7, 5) = 1 = 7 \cdot (-2) + 5 \cdot 3$
- $\gcd(391, 299) = 23 = 391 \cdot (-3) + 299 \cdot 4$

Examples

- $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
- $\gcd(7, 5) = 1 = 7 \cdot (-2) + 5 \cdot 3$
- $\gcd(391, 299) = 23 = 391 \cdot (-3) + 299 \cdot 4$
- $\gcd(239, 201) = 1 = 239 \cdot (-37) + 201 \cdot 44$

Extending Euclid's Algorithm

- Recall that Euclid's algorithm uses the fact that, for $a \geq b$, $\gcd(a, b) = \gcd(b, a \bmod b)$

Extending Euclid's Algorithm

- Recall that Euclid's algorithm uses the fact that, for $a \geq b$, $\gcd(a, b) = \gcd(b, a \bmod b)$
- Assume that $d = \gcd(b, a \bmod b)$ and that $d = bp + (a \bmod b)q$

Extending Euclid's Algorithm

- Recall that Euclid's algorithm uses the fact that, for $a \geq b$, $\gcd(a, b) = \gcd(b, a \bmod b)$
- Assume that $d = \gcd(b, a \bmod b)$ and that $d = bp + (a \bmod b)q$
- Then

$$\begin{aligned}d &= bp + (a \bmod b)q \\&= bp + \left(a - \left\lfloor \frac{a}{b} \right\rfloor b\right)q \\&= aq + b\left(p - \left\lfloor \frac{a}{b} \right\rfloor q\right)\end{aligned}$$

Outline

Least Common Multiple

Diophantine Equations: Examples

Diophantine Equations: Theorem

Modular Division

Least Common Multiple

Definition

The **least common multiple**, $\text{lcm}(a, b)$, of integers a and b (both different from zero) is the smallest positive integer that is divisible by both a and b

Least Common Multiple

Definition

The **least common multiple**, $\text{lcm}(a, b)$, of integers a and b (both different from zero) is the smallest positive integer that is divisible by both a and b

Examples

$\text{lcm}(24, 16) = 48$, $\text{lcm}(9, 17) = 153$,
 $\text{lcm}(239, 0)$ — undefined

Convention

We assume that a and b are positive

Naive Algorithm

Clearly, $a \cdot b$ is divisible by a and b . To find the *least* common multiple, simply try all numbers up to $a \cdot b$ and select the smallest one

Naive Algorithm: Code

```
def lcm(a, b):
    assert a > 0 and b > 0

    for d in range(1, a * b + 1):
        if d % a == 0 and d % b == 0:
            return d
```

Naive Algorithm: Code

```
def lcm(a, b):
    assert a > 0 and b > 0

    for d in range(1, a * b + 1):
        if d % a == 0 and d % b == 0:
            return d

print(lcm(24, 16))
```

Naive Algorithm: Analysis

- If $\text{lcm}(a, b) = a \cdot b$, the algorithm will perform $a \cdot b$ divisions

Naive Algorithm: Analysis

- If $\text{lcm}(a, b) = a \cdot b$, the algorithm will perform $a \cdot b$ divisions
- Again, very slow: the call

```
print(lcm(531441, 262144))
```

will take more than one minute

Euclid's algorithm

Can we use efficient Euclid's algorithm to compute the **least common multiple**, too?



Euclid's algorithm

Can we use efficient Euclid's algorithm to compute the **least common multiple**, too?

Yes!

$$\text{lcm}(a, b) \cdot \text{gcd}(a, b) = a \cdot b$$



lcm and gcd

Lemma

If $a, b > 0$, then $\text{lcm}(a, b) = ab/\text{gcd}(a, b)$

lcm and gcd

Lemma

If $a, b > 0$, then $\text{lcm}(a, b) = ab/\text{gcd}(a, b)$

Proof

- Let $d = \text{gcd}(a, b)$, $a = dp$, $b = dq$
- Then $m = ab/d = pb = qa$ is a multiple of a and b



lcm and gcd

Lemma

If $a, b > 0$, then $\text{lcm}(a, b) = ab/\text{gcd}(a, b)$

Proof

- Let $d = \text{gcd}(a, b)$, $a = dp$, $b = dq$
- Then $m = ab/d = pb = qa$ is a multiple of a and b
- If there was a smaller multiple $\bar{m} < m$, then $\bar{d} = ab/\bar{m} > d$ would be a common divisor: $a/\bar{d} = \bar{m}/b$, $b/\bar{d} = \bar{m}/a$



Outline

Least Common Multiple

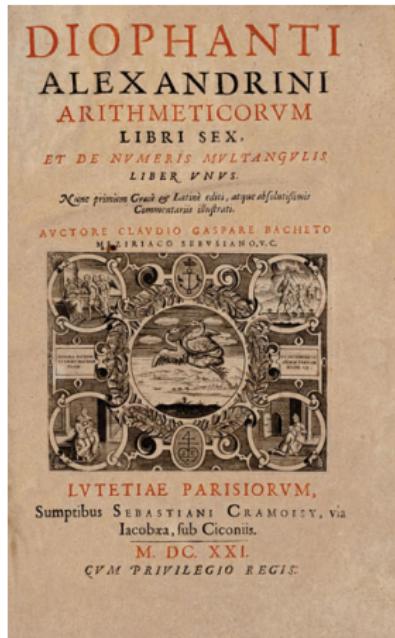
Diophantine Equations: Examples

Diophantine Equations: Theorem

Modular Division

Diophantine Equations

A Diophantine equation
is an equation where only
integer solutions are al-
lowed



Diophantine Equations: Examples

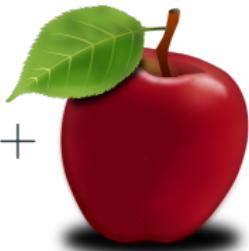
- 1 apple costs 22 pesos

Diophantine Equations: Examples

- 1 apple costs 22 pesos
- You only have 3-peso bills
- The cashier only has 5-peso bills
- $3x = 22 + 5y$, x, y are non-negative integers

Diophantine Equations: Examples

- 1 apple costs 22 pesos
- You only have 3-peso bills
- The cashier only has 5-peso bills
- $3x = 22 + 5y$, x, y are non-negative integers

 $\times 9 =$  $\times 1 +$ 

Diophantine Equations: Examples

- 1 apple costs 22 pesos
- You only have 3-peso bills
- The cashier only has 5-peso bills
- $3x = 22 + 5y$, x, y are non-negative integers

 $\times 9 =$  $\times 1 +$  $\times 14 =$  $\times 4 +$ 

Diophantine Equations: Examples

- $187x + 55y = 121$, x and y are integers

Diophantine Equations: Examples

- $187x + 55y = 121$, x and y are integers
 - $187 \cdot 3 + 55 \cdot (-8) = 121$

Diophantine Equations: Examples

- $187x + 55y = 121$, x and y are integers
 - $187 \cdot 3 + 55 \cdot (-8) = 121$
 - $187 \cdot (-2) + 55 \cdot 9 = 121$

Diophantine Equations: Examples

- $187x + 55y = 121$, x and y are integers
 - $187 \cdot 3 + 55 \cdot (-8) = 121$
 - $187 \cdot (-2) + 55 \cdot 9 = 121$
 - Infinitely many solutions!

Diophantine Equations: Examples

- $187x + 55y = 121$, x and y are integers
 - $187 \cdot 3 + 55 \cdot (-8) = 121$
 - $187 \cdot (-2) + 55 \cdot 9 = 121$
 - Infinitely many solutions!
- $187x + 55y = 45$, x and y are integers

Diophantine Equations: Examples

- $187x + 55y = 121$, x and y are integers
 - $187 \cdot 3 + 55 \cdot (-8) = 121$
 - $187 \cdot (-2) + 55 \cdot 9 = 121$
 - Infinitely many solutions!
- $187x + 55y = 45$, x and y are integers
 - No solutions!

Diophantine Equations: Examples

- $187x + 55y = 121$, x and y are integers
 - $187 \cdot 3 + 55 \cdot (-8) = 121$
 - $187 \cdot (-2) + 55 \cdot 9 = 121$
 - Infinitely many solutions!
- $187x + 55y = 45$, x and y are integers
 - No solutions!

When does a Diophantine equation have solutions?

Outline

Least Common Multiple

Diophantine Equations: Examples

Diophantine Equations: Theorem

Modular Division

Solutions of Diophantine Equations

Theorem

Given integers a, b, c (at least one of a and $b \neq 0$), the Diophantine equation

$$ax + by = c$$

has a solution (where x and y are integers) if and only if

$$\gcd(a, b) \mid c .$$

Proof of Theorem

Proof

Let $d = \gcd(a, b)$

$\Rightarrow a = dp$ and $b = dq$, thus,

$$c = ax + by = d(px + qy)$$

Proof of Theorem

Proof

Let $d = \gcd(a, b)$

$\Rightarrow a = dp$ and $b = dq$, thus,

$$c = ax + by = d(px + qy)$$

\Leftarrow Extended Euclid's algorithm:

$$a\bar{x} + b\bar{y} = d$$

If $c = td$, then $x = t \cdot \bar{x}, y = t \cdot \bar{y}$:

$$ax + by = t(a\bar{x} + b\bar{y}) = td = c$$

□

Finding a Solution

- $10x + 6y = 14$

Finding a Solution

- $10x + 6y = 14$
 - Extended Euclid's algorithm:
 $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
 - $14 = 10 \cdot (-1) \cdot 7 + 6 \cdot 2 \cdot 7$

Finding a Solution

- $10x + 6y = 14$
 - Extended Euclid's algorithm:
 $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
 - $14 = 10 \cdot (-1) \cdot 7 + 6 \cdot 2 \cdot 7$
 - $x = -7, y = 14$

Finding a Solution

- $10x + 6y = 14$
 - Extended Euclid's algorithm:
 $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
 - $14 = 10 \cdot (-1) \cdot 7 + 6 \cdot 2 \cdot 7$
 - $x = -7, y = 14$
- $391x + 299y = -69$

Finding a Solution

- $10x + 6y = 14$
 - Extended Euclid's algorithm:
 $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
 - $14 = 10 \cdot (-1) \cdot 7 + 6 \cdot 2 \cdot 7$
 - $x = -7, y = 14$
- $391x + 299y = -69$
 - Extended Euclid's Algorithm:
 $\gcd(391, 299) = 23 = 391 \cdot (-3) + 299 \cdot 4$

Finding a Solution

- $10x + 6y = 14$
 - Extended Euclid's algorithm:
 $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
 - $14 = 10 \cdot (-1) \cdot 7 + 6 \cdot 2 \cdot 7$
 - $x = -7, y = 14$
- $391x + 299y = -69$
 - Extended Euclid's Algorithm:
 $\gcd(391, 299) = 23 = 391 \cdot (-3) + 299 \cdot 4$
 - $-69 = 391 \cdot (-3) \cdot (-3) + 299 \cdot 4 \cdot (-3)$

Finding a Solution

- $10x + 6y = 14$
 - Extended Euclid's algorithm:
 $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
 - $14 = 10 \cdot (-1) \cdot 7 + 6 \cdot 2 \cdot 7$
 - $x = -7, y = 14$
- $391x + 299y = -69$
 - Extended Euclid's Algorithm:
 $\gcd(391, 299) = 23 = 391 \cdot (-3) + 299 \cdot 4$
 - $-69 = 391 \cdot (-3) \cdot (-3) + 299 \cdot 4 \cdot (-3)$
 - $x = 9, y = -12$

Finding a Solution

- $10x + 6y = 14$
 - Extended Euclid's algorithm:
 $\gcd(10, 6) = 2 = 10 \cdot (-1) + 6 \cdot 2$
 - $14 = 10 \cdot (-1) \cdot 7 + 6 \cdot 2 \cdot 7$
 - $x = -7, y = 14$
- $391x + 299y = -69$
 - Extended Euclid's Algorithm:
 $\gcd(391, 299) = 23 = 391 \cdot (-3) + 299 \cdot 4$
 - $-69 = 391 \cdot (-3) \cdot (-3) + 299 \cdot 4 \cdot (-3)$
 - $x = 9, y = -12$
 - But $x = -4, y = 5$ is also a solution. How do we find **all** solutions?

Euclid's Lemma

Euclid's Lemma

If $n \mid ab$ and $\gcd(a, n) = 1$, then $n \mid b$.

Euclid's Lemma

Euclid's Lemma

If $n \mid ab$ and $\gcd(a, n) = 1$, then $n \mid b$.

Proof

- From Extended Euclid's algorithm (a, n):

Euclid's Lemma

Euclid's Lemma

If $n \mid ab$ and $\gcd(a, n) = 1$, then $n \mid b$.

Proof

- From Extended Euclid's algorithm (a, n):
- $ax + ny = 1$

Euclid's Lemma

Euclid's Lemma

If $n \mid ab$ and $\gcd(a, n) = 1$, then $n \mid b$.

Proof

- From Extended Euclid's algorithm (a, n):
- $ax + ny = 1$
- $axb + nyb = b$

Euclid's Lemma

Euclid's Lemma

If $n \mid ab$ and $\gcd(a, n) = 1$, then $n \mid b$.

Proof

- From Extended Euclid's algorithm (a, n):
 - $ax + ny = 1$
 - $axb + nyb = b$
- From $ab = kn$,
 $b = axb + nyb = n(xk + yb)$



Finding All Solution

Theorem

Let $\gcd(a, b) = d$, $a = dp$, $b = dq$. If (x_0, y_0) is a solution of the Diophantine equation
 $ax + by = c$.

$$ax_0 + by_0 = c,$$

then all the solutions have the form

$$a(x_0 + tq) + b(y_0 - tp) = c,$$

where t is an arbitrary integer.

Proof of Theorem

Proof

- $a = dp, b = dq, ax_0 + by_0 = c$

Proof of Theorem

Proof

- $a = dp, b = dq, ax_0 + by_0 = c$
- For any integer t ,

$$\begin{aligned} & a(x_0 + tq) + b(y_0 - tp) \\ &= ax_0 + by_0 + t(aq - bp) \\ &= c + t(dpq - dpq) = c \end{aligned}$$

is a solution

Proof of Theorem

Proof (continued)

- Consider 2 solutions: (x_1, y_1) and (x_2, y_2)

Proof of Theorem

Proof (continued)

- Consider 2 solutions: (x_1, y_1) and (x_2, y_2)
- $a(x_1 - x_2) + b(y_1 - y_2) = c - c = 0$

Proof of Theorem

Proof (continued)

- Consider 2 solutions: (x_1, y_1) and (x_2, y_2)
- $a(x_1 - x_2) + b(y_1 - y_2) = c - c = 0$
- $p(x_1 - x_2) + q(y_1 - y_2) = 0$

Proof of Theorem

Proof (continued)

- Consider 2 solutions: (x_1, y_1) and (x_2, y_2)
- $a(x_1 - x_2) + b(y_1 - y_2) = c - c = 0$
- $p(x_1 - x_2) + q(y_1 - y_2) = 0$
- $\gcd(p, q) = 1$

Proof of Theorem

Proof (continued)

- Consider 2 solutions: (x_1, y_1) and (x_2, y_2)
- $a(x_1 - x_2) + b(y_1 - y_2) = c - c = 0$
- $p(x_1 - x_2) + q(y_1 - y_2) = 0$
- $\gcd(p, q) = 1$
- By Euclid's lemma: $x_1 - x_2 = tq$

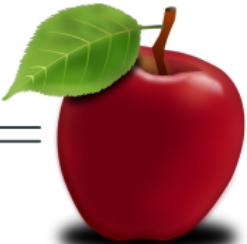
Proof of Theorem

Proof (continued)

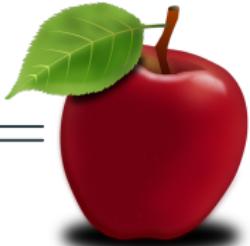
- Consider 2 solutions: (x_1, y_1) and (x_2, y_2)
- $a(x_1 - x_2) + b(y_1 - y_2) = c - c = 0$
- $p(x_1 - x_2) + q(y_1 - y_2) = 0$
- $\gcd(p, q) = 1$
- By Euclid's lemma: $x_1 - x_2 = tq$
- Then $y_1 - y_2 = -tp$

□

Example

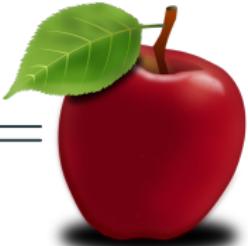
 $\times 9 +$  $\times 1 =$ 

Example

 $\times 9 +$  $\times 1 =$ 

- $3x + 5y = 22$

Example

 $\times 9 +$  $\times 1 =$ 

- $3x + 5y = 22$
- $x_0 = 9, y_0 = -1$

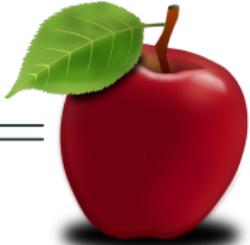
Example



$\times 9 +$

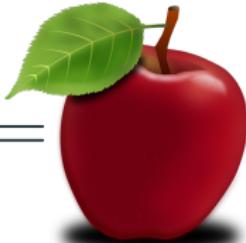


$\times 1 =$



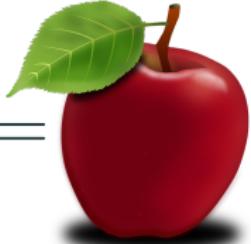
- $3x + 5y = 22$
- $x_0 = 9, y_0 = -1$
- $a = 3, b = 5, d = 1$

Example

 $\times 9 +$  $\times 1 =$ 

- $3x + 5y = 22$
- $x_0 = 9, y_0 = -1$
- $a = 3, b = 5, d = 1$
- $a = dp, b = dq, p = 3, q = 5$

Example

 $\times 9 +$  $\times 1 =$ 

- $3x + 5y = 22$
- $x_0 = 9, y_0 = -1$
- $a = 3, b = 5, d = 1$
- $a = dp, b = dq, p = 3, q = 5$
- All solutions:

$$x = x_0 + tq = 9 + 5t$$

$$y = y_0 - tp = -1 - 3t$$

Example

- All solutions:

$$x = x_0 + tq = 9 + 5t$$

$$y = y_0 - tp = -1 - 3t$$

Example

- All solutions:

$$x = x_0 + tq = 9 + 5t$$

$$y = y_0 - tp = -1 - 3t$$

- If we want $x \geq 0$ and $y \leq 0$, then take

$$9 + 5t \geq 0$$

$$-1 - 3t \leq 0$$

Example

- All solutions:

$$x = x_0 + tq = 9 + 5t$$

$$y = y_0 - tp = -1 - 3t$$

- If we want $x \geq 0$ and $y \leq 0$, then take

$$9 + 5t \geq 0$$

$$-1 - 3t \leq 0$$

- That is, $t \geq -1/3$, or $t \geq 0$

Outline

Least Common Multiple

Diophantine Equations: Examples

Diophantine Equations: Theorem

Modular Division

Division mod 7

\times	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Division mod 7

\times	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

- Given $a \neq 0$ and b , there exists x such that $a \times x \equiv b \pmod{7}$

Division mod 7

\times	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

- Given $a \neq 0$ and b , there exists x such that $a \times x \equiv b \pmod{7}$
- x plays the role of modular division $x = b/a \pmod{7}$

Division mod 6

\times	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Division mod 6

\times	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

- $2/5 \equiv 4 \pmod{6}$.
Indeed, $4 \times 5 \equiv 2 \pmod{6}$

Division mod 6

\times	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

- $2/5 \equiv 4 \pmod{6}$.
Indeed, $4 \times 5 \equiv 2 \pmod{6}$
- But there is no x s.t. $3 \times x \equiv 1 \pmod{6}$

Division mod 6

\times	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

- $2/5 \equiv 4 \pmod{6}$.
Indeed, $4 \times 5 \equiv 2 \pmod{6}$
- But there is no x s.t. $3 \times x \equiv 1 \pmod{6}$
- We can't divide 1 by 3 modulo 6!

Multiplicative Inverse

- A multiplicative inverse of $a \bmod n$ is \bar{a} s.t.

$$a \times \bar{a} \equiv 1 \pmod{n}$$

Multiplicative Inverse

- A multiplicative inverse of $a \bmod n$ is \bar{a} s.t.

$$a \times \bar{a} \equiv 1 \pmod{n}$$

- If a has a multiplicative inverse \bar{a} , then one can divide by a :

$$b/a \equiv b \times \bar{a} \pmod{n}$$

Multiplicative Inverse

- A multiplicative inverse of $a \bmod n$ is \bar{a} s.t.

$$a \times \bar{a} \equiv 1 \pmod{n}$$

- If a has a multiplicative inverse \bar{a} , then one can divide by a :

$$b/a \equiv b \times \bar{a} \pmod{n}$$

- Indeed, for every b ,

$$b/a \times a \equiv b \times \bar{a} \times a \equiv b \pmod{n}$$

Uniqueness of Inverses

Lemma

If a has a multiplicative inverse, then it is unique

Uniqueness of Inverses

Lemma

If a has a multiplicative inverse, then it is unique

Proof

If x and y are multiplicative inverses of a , then

$$x = x \times (a \times y) = (x \times a) \times y = y$$



Existence of Inverses

Theorem

a has a multiplicative inverse modulo n if and only if $\gcd(a, n) = 1$

Existence of Inverses

Theorem

a has a multiplicative inverse modulo n if and only if $\gcd(a, n) = 1$

Proof

- $ax \equiv 1 \pmod{n}$ iff $ax + kn = 1$

Existence of Inverses

Theorem

a has a multiplicative inverse modulo n if and only if $\gcd(a, n) = 1$

Proof

- $ax \equiv 1 \pmod{n}$ iff $ax + kn = 1$
- For fixed a and n , this Diophantine equation has a solution (x) iff $\gcd(a, n) \mid 1$



Modular Division

- If $\gcd(a, n) = 1$ then one can divide by a modulo n

Modular Division

- If $\gcd(a, n) = 1$ then one can divide by a modulo n
- Given a, b, n , we want to find $x \equiv b/a \pmod{n}$:

Modular Division

- If $\gcd(a, n) = 1$ then one can divide by a modulo n
- Given a, b, n , we want to find $x \equiv b/a \pmod{n}$:
 - First, use Extended Euclid's algorithm to find s and t : $nt + as = 1$

Modular Division

- If $\gcd(a, n) = 1$ then one can divide by a modulo n
- Given a, b, n , we want to find $x \equiv b/a \pmod{n}$:
 - First, use Extended Euclid's algorithm to find s and t : $nt + as = 1$
 - s is the multiplicative inverse of a modulo n

Modular Division

- If $\gcd(a, n) = 1$ then one can divide by a modulo n
- Given a, b, n , we want to find $x \equiv b/a \pmod{n}$:
 - First, use Extended Euclid's algorithm to find s and t : $nt + as = 1$
 - s is the multiplicative inverse of a modulo n
 - Now $x \equiv b/a \equiv b \times s \pmod{n}$

Example

- $\gcd(9, 2) = 1$, so we can compute

$$7/2 \pmod{9}$$

Example

- $\gcd(9, 2) = 1$, so we can compute

$$7/2 \pmod{9}$$

- Extended Euclid's algorithm gives us

$$9 \times 1 + 2 \times (-4) = 1$$

Example

- $\gcd(9, 2) = 1$, so we can compute

$$7/2 \pmod{9}$$

- Extended Euclid's algorithm gives us

$$9 \times 1 + 2 \times (-4) = 1$$

- $-4 \equiv 5 \pmod{9}$ is the inverse of 2 mod 9

Example

- $\gcd(9, 2) = 1$, so we can compute

$$7/2 \pmod{9}$$

- Extended Euclid's algorithm gives us

$$9 \times 1 + 2 \times (-4) = 1$$

- $-4 \equiv 5 \pmod{9}$ is the inverse of 2 mod 9
- $7/2 \equiv 7 \times 5 \equiv 8 \pmod{9}$

Example

- $\gcd(9, 2) = 1$, so we can compute

$$7/2 \pmod{9}$$

- Extended Euclid's algorithm gives us

$$9 \times 1 + 2 \times (-4) = 1$$

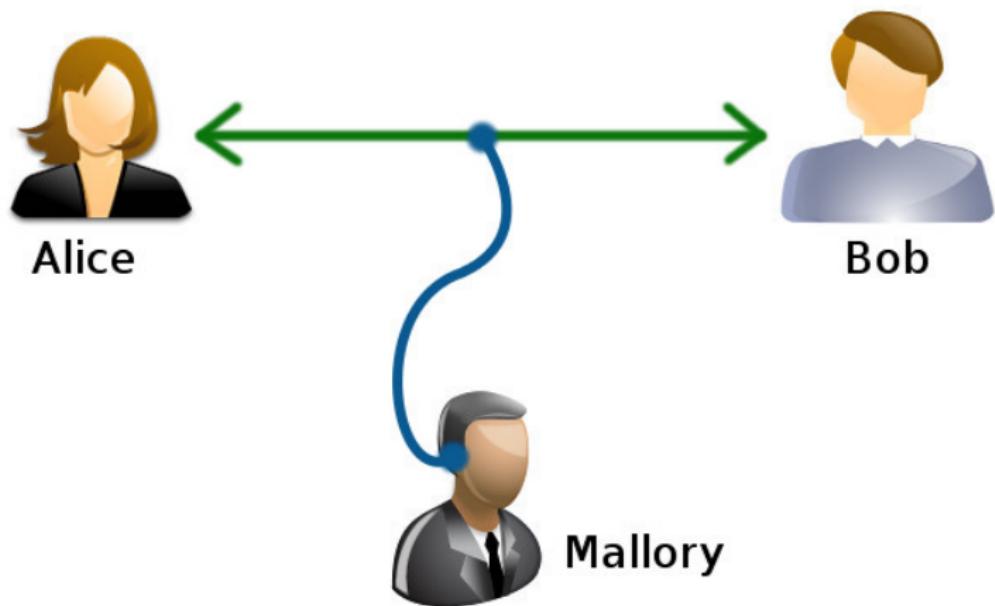
- $-4 \equiv 5 \pmod{9}$ is the inverse of 2 mod 9
- $7/2 \equiv 7 \times 5 \equiv 8 \pmod{9}$
- Indeed, $8 \times 2 \equiv 7 \pmod{9}$

Sharing Secrets

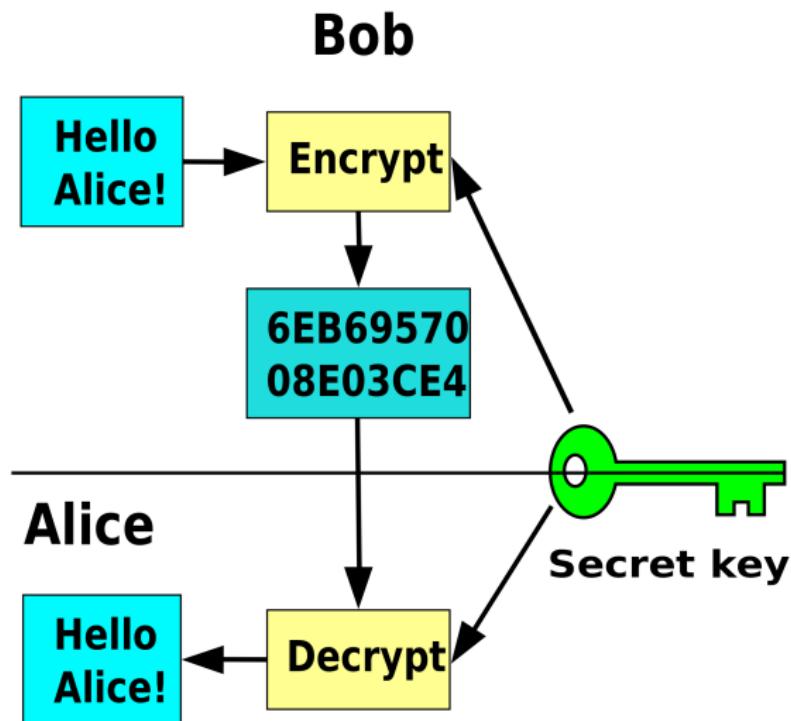


- Private communications via phone, e-mail, messengers
- Secure money transfer, online shopping
- Secure authorization for online services
- Secure software installation

Eavesdropping



Secret Code



Changing the Code

- If you use the same secret code many times, people around can guess what it is

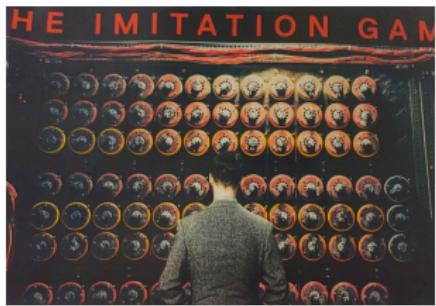
Changing the Code

- If you use the same secret code many times, people around can guess what it is
- Changing words to their opposites, replacing some words with other special words, rearranging letters — all these can be broken using statistics if there are many examples of encrypted messages

Changing the Code

- If you use the same secret code many times, people around can guess what it is
- Changing words to their opposites, replacing some words with other special words, rearranging letters — all these can be broken using statistics if there are many examples of encrypted messages
- Need to change the code often

- The Nazis changed their code once a day during the war, but the Allies led by Alan Turing still broke the cipher
- One should use different code for each communication

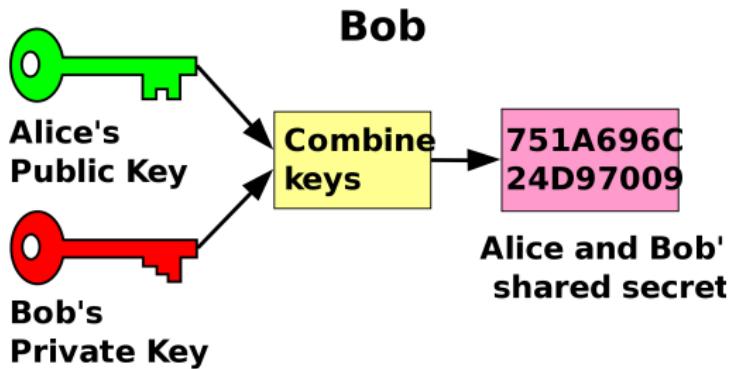
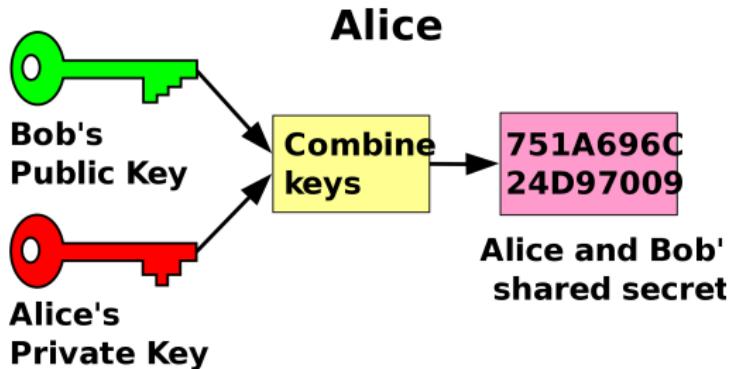


flickr.com

Sharing the Secret Code

- But how to share the secret code itself?
- Eavesdropper can get it
- What if you are communicating from different continents?
- And you need a new code for each communication

Public Key Cryptography



Cryptography

- Sharing secrets in such a way that noone can eavesdrop or change your messages
- Authorization and making sure a person cannot deny having sent the message
- Billions of money transactions use encryption everyday
- RSA encryption — arguably the most used program in the world

Outline

Remainders

Chinese Remainder Theorem

Remainders

- Properties of remainders
- When remainder modulo a defines remainder modulo b
- When remainders modulo a and b are independent

n	0	1	2	3	4	5	6	7	8	9	10	11
$n \bmod 4$	0	1	2	3	0	1	2	3	0	1	2	3
$n \bmod 2$	0	1	0	1	0	1	0	1	0	1	0	1

n	0	1	2	3	4	5	6	7	8	9	10	11
$n \bmod 4$	0	1	2	3	0	1	2	3	0	1	2	3
$n \bmod 2$	0	1	0	1	0	1	0	1	0	1	0	1

- $n \bmod 4$ defines $n \bmod 2$!
- Indeed, if $n_1 \equiv n_2 \pmod{4}$, then $4 \mid (n_1 - n_2)$, so $2 \mid (n_1 - n_2)$ and $n_1 \equiv n_2 \pmod{2}$

n	0	1	2	3	4	5	6	7	8	9	10	11
$n \bmod 3$	0	1	2	0	1	2	0	1	2	0	1	2
$n \bmod 2$	0	1	0	1	0	1	0	1	0	1	0	1

- $n \bmod 3$ doesn't define $n \bmod 2$
- All pairs of remainders are possible
- One remainder doesn't give information about another

Divisibility Criteria

- Divisibility by 2: the last digit is even

Divisibility Criteria

- Divisibility by 2: the last digit is even
- Divisibility by 5: the last digit is 0 or 5 (5 divides the last digit)

Divisibility Criteria

- Divisibility by 2: the last digit is even
- Divisibility by 5: the last digit is 0 or 5 (5 divides the last digit)
- The last digit is the remainder after division of n by 10

Divisibility Criteria

- Divisibility by 2: the last digit is even
- Divisibility by 5: the last digit is 0 or 5 (5 divides the last digit)
- The last digit is the remainder after division of n by 10
- $10 = 2 \cdot 5$

Divisibility Criteria

- Divisibility by 2: the last digit is even
- Divisibility by 5: the last digit is 0 or 5 (5 divides the last digit)
- The last digit is the remainder after division of n by 10
- $10 = 2 \cdot 5$
- Divisibility by 3: the sum of digits is divisible by 3 — is not determined just by the last digit

Remainders

Lemma

If n_1 and n_2 have the same remainders modulo b and $a \mid b$, then n_1 and n_2 have the same remainders modulo a .

Proof

- $n_1 \equiv n_2 \pmod{b} \Rightarrow b \mid (n_1 - n_2)$

Proof

- $n_1 \equiv n_2 \pmod{b} \Rightarrow b \mid (n_1 - n_2)$
- $b \mid (n_1 - n_2), a \mid b \Rightarrow a \mid (n_1 - n_2)$

Proof

- $n_1 \equiv n_2 \pmod{b} \Rightarrow b \mid (n_1 - n_2)$
- $b \mid (n_1 - n_2), a \mid b \Rightarrow a \mid (n_1 - n_2)$
- $a \mid (n_1 - n_2) \Rightarrow n_1 \equiv n_2 \pmod{a}$

□

Problem

If $n \equiv 1 \pmod{6}$, then what can be $n \pmod{4}$?

Solution

- If n is even, then the remainder modulo 6 can be only 0, 2 or 4

Solution

- If n is even, then the remainder modulo 6 can be only 0, 2 or 4
- So n is odd, and $n \bmod 4$ can be either 1 or 3

Solution

- If n is even, then the remainder modulo 6 can be only 0, 2 or 4
- So n is odd, and $n \bmod 4$ can be either 1 or 3
- If $n = 1$, then $n \equiv 1 \bmod 6$ and $n \equiv 1 \bmod 4$

Solution

- If n is even, then the remainder modulo 6 can be only 0, 2 or 4
- So n is odd, and $n \bmod 4$ can be either 1 or 3
- If $n = 1$, then $n \equiv 1 \bmod 6$ and $n \equiv 1 \bmod 4$
- If $n = 7$, then $n \equiv 1 \bmod 6$ and $n \equiv 3 \bmod 4$

Solution

- If n is even, then the remainder modulo 6 can be only 0, 2 or 4
- So n is odd, and $n \bmod 4$ can be either 1 or 3
- If $n = 1$, then $n \equiv 1 \bmod 6$ and $n \equiv 1 \bmod 4$
- If $n = 7$, then $n \equiv 1 \bmod 6$ and $n \equiv 3 \bmod 4$
- So $n \bmod 4$ can be either 1 or 3

- Remainders modulo 4 and 6 are dependent
- This is because 2 is their common divisor
- In general, if $d \mid a$ and $d \mid b$, then remainders modulo a and b are dependent
- It turns out that if $\text{GCD}(a, b) = 1$, then the remainders modulo a and b are independent

Outline

Remainders

Chinese Remainder Theorem

Chinese Remainder Theorem

Theorem

If $\text{GCD}(a, b) = 1$, then for any remainder r_a modulo a and any remainder r_b modulo b there exists integer n , such that $n \equiv r_a \pmod{a}$ and $n \equiv r_b \pmod{b}$. If n_1 and n_2 are two such integers, then $n_1 \equiv n_2 \pmod{ab}$.

In Other Words

Consider all ab remainders modulo ab :

$$0, 1, 2, \dots, ab - 1$$

Every such remainder r corresponds to a pair of remainders (r_a, r_b) :

$$r \equiv r_a \pmod{a}, r \equiv r_b \pmod{b}$$

Claim: if we consider pairs corresponding to each r , then each of the possible ab pairs (r_a, r_b) appears exactly once.

Proof

First, let us prove that if n_1 and n_2 correspond to the same pair (r_a, r_b) , then $n_1 \equiv n_2 \pmod{ab}$:

- $n_1 \equiv r_a \equiv n_2 \pmod{a}$

Proof

First, let us prove that if n_1 and n_2 correspond to the same pair (r_a, r_b) , then $n_1 \equiv n_2 \pmod{ab}$:

- $n_1 \equiv r_a \equiv n_2 \pmod{a}$
- $n_1 \equiv n_2 \pmod{a} \Rightarrow a \mid (n_1 - n_2)$

Proof

First, let us prove that if n_1 and n_2 correspond to the same pair (r_a, r_b) , then $n_1 \equiv n_2 \pmod{ab}$:

- $n_1 \equiv r_a \equiv n_2 \pmod{a}$
- $n_1 \equiv n_2 \pmod{a} \Rightarrow a \mid (n_1 - n_2)$
- Similarly, $b \mid (n_1 - n_2)$

Proof

First, let us prove that if n_1 and n_2 correspond to the same pair (r_a, r_b) , then $n_1 \equiv n_2 \pmod{ab}$:

- $n_1 \equiv r_a \equiv n_2 \pmod{a}$
- $n_1 \equiv n_2 \pmod{a} \Rightarrow a \mid (n_1 - n_2)$
- Similarly, $b \mid (n_1 - n_2)$
- a and b are coprime and both divide $(n_1 - n_2)$, so $ab \mid (n_1 - n_2)$

Proof

First, let us prove that if n_1 and n_2 correspond to the same pair (r_a, r_b) , then $n_1 \equiv n_2 \pmod{ab}$:

- $n_1 \equiv r_a \equiv n_2 \pmod{a}$
- $n_1 \equiv n_2 \pmod{a} \Rightarrow a \mid (n_1 - n_2)$
- Similarly, $b \mid (n_1 - n_2)$
- a and b are coprime and both divide $(n_1 - n_2)$, so $ab \mid (n_1 - n_2)$
- $ab \mid (n_1 - n_2) \Rightarrow n_1 \equiv n_2 \pmod{ab}$ □

Corollary

- Different r lead to different pairs (r_a, r_b)
- This already proves the first part of the theorem
- The number of remainders r modulo ab is ab , and the number of pairs of remainders (r_a, r_b) is also $a \cdot b = ab$
- Each r corresponds to unique (r_a, r_b) , so each pair (r_a, r_b) corresponds to unique r
- We will also show a constructive proof

Proof

- $\text{GCD}(a, b) = 1$, so $1 = ax + by$ for some integer x, y

Proof

- $\text{GCD}(a, b) = 1$, so $1 = ax + by$ for some integer x, y
- $ax \equiv 1 \pmod{b}$, $by \equiv 1 \pmod{a}$

Proof

- $\text{GCD}(a, b) = 1$, so $1 = ax + by$ for some integer x, y
- $ax \equiv 1 \pmod{b}$, $by \equiv 1 \pmod{a}$
- Thus ax corresponds to pair of remainders $(0, 1)$, and by corresponds to $(1, 0)$

Proof

- $\text{GCD}(a, b) = 1$, so $1 = ax + by$ for some integer x, y
- $ax \equiv 1 \pmod{b}$, $by \equiv 1 \pmod{a}$
- Thus ax corresponds to pair of remainders $(0, 1)$, and by corresponds to $(1, 0)$
- Combine: $(r_a, r_b) = r_a \cdot (1, 0) + r_b \cdot (0, 1)$

Proof

- $\text{GCD}(a, b) = 1$, so $1 = ax + by$ for some integer x, y
- $ax \equiv 1 \pmod{b}$, $by \equiv 1 \pmod{a}$
- Thus ax corresponds to pair of remainders $(0, 1)$, and by corresponds to $(1, 0)$
- Combine: $(r_a, r_b) = r_a \cdot (1, 0) + r_b \cdot (0, 1)$
- Consider $n = r_a by + r_b ax$

Proof

- $\text{GCD}(a, b) = 1$, so $1 = ax + by$ for some integer x, y
- $ax \equiv 1 \pmod{b}$, $by \equiv 1 \pmod{a}$
- Thus ax corresponds to pair of remainders $(0, 1)$, and by corresponds to $(1, 0)$
- Combine: $(r_a, r_b) = r_a \cdot (1, 0) + r_b \cdot (0, 1)$
- Consider $n = r_a by + r_b ax$
- $n = r_a by + r_b ax \equiv r_a by \equiv r_a \pmod{a}$

Proof

- $\text{GCD}(a, b) = 1$, so $1 = ax + by$ for some integer x, y
- $ax \equiv 1 \pmod{b}$, $by \equiv 1 \pmod{a}$
- Thus ax corresponds to pair of remainders $(0, 1)$, and by corresponds to $(1, 0)$
- Combine: $(r_a, r_b) = r_a \cdot (1, 0) + r_b \cdot (0, 1)$
- Consider $n = r_a by + r_b ax$
- $n = r_a by + r_b ax \equiv r_a by \equiv r_a \pmod{a}$
- $n = r_a by + r_b ax \equiv r_b ax \equiv r_b \pmod{b}$ □

Algorithm

The proof gives us this simple algorithm to find such n :

- Use extended Euclid's algorithm to find such x, y that $ax + by = 1$
- Take $n = r_a \cdot by + r_b \cdot ax$

More Modules

What about the case of 3 modules a , b and c ?

More Modules

What about the case of 3 modules a, b and c ?

Turns out, if all pairs (a, b) , (a, c) and (b, c) are coprime, then remainders modulo a, b and c uniquely determine remainder modulo abc .

More Modules

What about the case of 3 modules a, b and c ?

Turns out, if all pairs (a, b) , (a, c) and (b, c) are coprime, then remainders modulo a, b and c uniquely determine remainder modulo abc .

To prove, first go from remainders modulo a and b to remainder modulo ab , then go from remainders modulo ab and c to remainder modulo abc .

Computations with Large Integers

Instead of large integers, we can work with their remainders modulo several big prime numbers: if $0 \leq n_1, n_2 < p_1 p_2 \dots p_k$, and $n_1 \equiv n_2 \pmod{p_i}$ for all i , then $n_1 = n_2$.

In this form, it would be fast to sum and multiply large integers, but hard to compare. This is actually used to speed up computations.

Conclusion

- Remainder modulo n uniquely determines remainder modulo any divisor of n
- Remainders modulo a and b are independent if and only if $\text{GCD}(a, b) = 1$
- Remainders modulo coprime a and b uniquely determine remainder modulo ab
- Algorithm for constructing remainder modulo ab given remainders modulo coprime a and b
- Extends to more modules if every pair is coprime

Outline

Fast Modular Exponentiation

Fermat's Little Theorem

Euler's Totient Function

Euler's Theorem

Modular Exponentiation

- The central operation for public key cryptography
- Properties allow fast encryption and decryption for Alice and Bob
- Easy in one direction, hard in reverse — for Eve
- Computing modular exponent quickly
- Key properties for encryption and decryption

Modular Exponentiation

$$c \leftarrow b^e \pmod{m}$$

- How to compute $b^e \bmod m$?

- How to compute $b^e \bmod m$?
- No need to compute the giant number b^e and divide by m : we can start with 1, then multiply by b and immediately take the result modulo m , repeat e times

$b = 7, e = 4, m = 11$:

$$c \leftarrow 1$$

$$c \leftarrow (c \cdot b) \equiv (1 \cdot 7) \equiv 7 \bmod 11 = 7$$

$b = 7, e = 4, m = 11$:

$$c \leftarrow 7$$

$$c \leftarrow (c \cdot b) \equiv (7 \cdot 7) \equiv 49 \bmod 11 = 5$$

$b = 7, e = 4, m = 11$:

$$c \leftarrow 5$$

$$c \leftarrow (c \cdot b) \equiv (5 \cdot 7) \equiv 35 \bmod 11 = 2$$

$$b = 7, e = 4, m = 11:$$

$$c \leftarrow 2$$

$$c \leftarrow (c \cdot b) \equiv (2 \cdot 7) \equiv 14 \bmod{11} = 3$$

$b = 7, e = 4, m = 11$:

$$c \leftarrow 2$$

$$c \leftarrow (c \cdot b) \equiv (2 \cdot 7) \equiv 14 \bmod 11 = 3$$

$$b^e \bmod m = 7^4 \bmod 11 = 3$$

Straightforward Algorithm

- Start with $c \leftarrow 1$
- Repeat e times: $c \leftarrow c \cdot b \bmod m$

- Just e multiplications

- Just e multiplications
- Fast, right?

- Just e multiplications
- Fast, right?
- What if b, e, m are integers with 1000 digits?

- Just e multiplications
- Fast, right?
- What if b, e, m are integers with 1000 digits?
- Can we do faster?

What if $e = 2^k$?

$b = 7, e = 128, m = 11:$

$$7^1 \equiv 7 \pmod{11} = 7$$

$$7^2 \equiv 7 \cdot 7 \equiv 49 \pmod{11} = 5$$

$b = 7, e = 128, m = 11$:

$$7^2 \equiv 7 \cdot 7 \equiv 49 \pmod{11} = 5$$

$$7^4 \equiv 7^2 \cdot 7^2 \equiv 5 \cdot 5 \equiv 25 \pmod{11} = 3$$

$b = 7, e = 128, m = 11$:

$$7^4 \equiv 7^2 \cdot 7^2 \equiv 5 \cdot 5 \equiv 25 \pmod{11} = 3$$

$$7^8 \equiv 7^4 \cdot 7^4 \equiv 3 \cdot 3 \equiv 9 \pmod{11} = 9$$

$b = 7, e = 128, m = 11$:

$$7^8 \equiv 7^4 \cdot 7^4 \equiv 3 \cdot 3 \equiv 9 \pmod{11} = 9$$

$$7^{16} \equiv 7^8 \cdot 7^8 \equiv 9 \cdot 9 \equiv 81 \pmod{11} = 4$$

$b = 7, e = 128, m = 11:$

$$7^{16} \equiv 7^8 \cdot 7^8 \equiv 9 \cdot 9 \equiv 81 \bmod 11 = 4$$

$$7^{32} \equiv 7^{16} \cdot 7^{16} \equiv 4 \cdot 4 \equiv 16 \bmod 11 = 5$$

$b = 7, e = 128, m = 11:$

$$7^{32} \equiv 7^{16} \cdot 7^{16} \equiv 4 \cdot 4 \equiv 16 \pmod{11} = 5$$

$$7^{64} \equiv 7^{32} \cdot 7^{32} \equiv 5 \cdot 5 \equiv 25 \pmod{11} = 3$$

$b = 7, e = 128, m = 11$:

$$7^{64} \equiv 7^{32} \cdot 7^{32} \equiv 5 \cdot 5 \equiv 25 \pmod{11} = 3$$

$$7^{128} \equiv 7^{64} \cdot 7^{64} \equiv 3 \cdot 3 \equiv 9 \pmod{11} = 9$$

- Start with $c \leftarrow b \bmod m$
- Repeat k times: $c \leftarrow c^2 \bmod m$
- In the end, $c = b^{2^k} = b^e \bmod m$

- Just $k = \log_2 2^k = \log_2 e$ multiplications — much faster!

- Just $k = \log_2 2^k = \log_2 e$ multiplications — much faster!
- What if e is not a power of 2?

$$b,b^2,b^4,b^8,\ldots$$

$$b,b^2,b^4,b^8,\ldots$$

$$b^{13}?$$

$$b,b^2,b^4,b^8,\ldots$$

$$b^{13}?$$

$$13=8+4+1$$

$$b,b^2,b^4,b^8,\ldots$$

$$b^{13}?$$

$$13=8+4+1$$

$$b^{13}=b^8\cdot b^4\cdot b^1$$

$b = 7, e = 13, m = 11:$

$b = 7, e = 13, m = 11:$

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b = 7, e = 13, m = 11:$

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$
$b^{2^1} \bmod m$
$b^{2^2} \bmod m$
$b^{2^3} \bmod m$
$b^{13} \bmod m$

$b = 7, e = 13, m = 11:$

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$	7
$b^{2^1} \bmod m$	5
$b^{2^2} \bmod m$	3
$b^{2^3} \bmod m$	9
$b^{13} \bmod m$	

$b = 7, e = 13, m = 11:$

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$	7	\times
$b^{2^1} \bmod m$	5	
$b^{2^2} \bmod m$	3	\times
$b^{2^3} \bmod m$	9	\times
$b^{13} \bmod m$		

$b = 7, e = 13, m = 11:$

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$	7	\times
$b^{2^1} \bmod m$	5	
$b^{2^2} \bmod m$	3	\times
$b^{2^3} \bmod m$	9	\times
$b^{13} \bmod m$		

$$7 \cdot 3 \cdot 9 \equiv 21 \cdot 9 \equiv 10 \cdot 9 \equiv 90 \equiv 2 \bmod 11$$

$b = 7, e = 13, m = 11:$

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$	7	\times
$b^{2^1} \bmod m$	5	
$b^{2^2} \bmod m$	3	\times
$b^{2^3} \bmod m$	9	\times
$b^{13} \bmod m$	2	

$$7 \cdot 3 \cdot 9 \equiv 21 \cdot 9 \equiv 10 \cdot 9 \equiv 90 \equiv 2 \bmod 11$$

- Rewrite e in binary form: $e = 1101 \dots 01_2$
- Compute $b^{2^k} \bmod m$ for all $2^k \leq e$
- Multiply together the results for all 2^k in the binary representation of e
- $\log_2 e$ multiplications to compute all $b^{2^k} \bmod m$
- At most $\log_2 e$ multiplications to compute $b^e \bmod m$ by multiplying all the needed $b^{2^k} \bmod m$
- At most $2 \log_2 e$ multiplications in total!

Conclusion

- Modular exponentiation can be computed using at most $2 \log_2 e$ multiplications
- Represent e in binary form
- Compute $b^{2^k} \bmod m$ for all $2^k \leq e$ using squaring
- Compute $b^e \bmod m$ as a product of all needed b^{2^k} using binary representation of e

Outline

Fast Modular Exponentiation

Fermat's Little Theorem

Euler's Totient Function

Euler's Theorem

Fermat's Little Theorem

- Key result for cryptography using modular exponentiation
- Key result for fast algorithms testing whether a large integer is prime
- Can be used to make modular exponentiation even faster

Fermat's Little Theorem

Theorem

If prime p doesn't divide integer a , then

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

Proof

- Consider all $p - 1$ non-zero remainders modulo p : $1, 2, \dots, p - 1$

Proof

- Consider all $p - 1$ non-zero remainders modulo p : $1, 2, \dots, p - 1$
- Multiplying any such remainder by a is invertible since $p \nmid a$, so the new remainder is also non-zero

Proof

- Consider all $p - 1$ non-zero remainders modulo p : $1, 2, \dots, p - 1$
- Multiplying any such remainder by a is invertible since $p \nmid a$, so the new remainder is also non-zero
- Graph on remainders: edge from r to ar

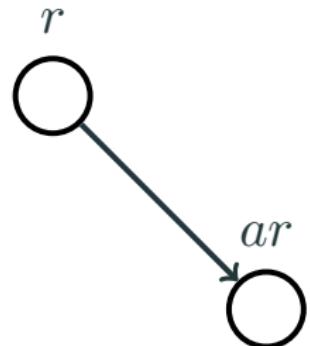
Proof

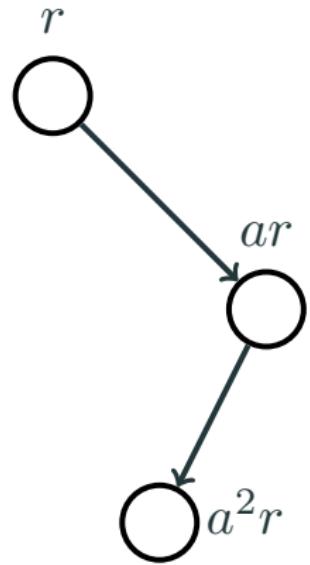
- Consider all $p - 1$ non-zero remainders modulo p : $1, 2, \dots, p - 1$
- Multiplying any such remainder by a is invertible since $p \nmid a$, so the new remainder is also non-zero
- Graph on remainders: edge from r to ar
- All incoming and outgoing degrees are 1

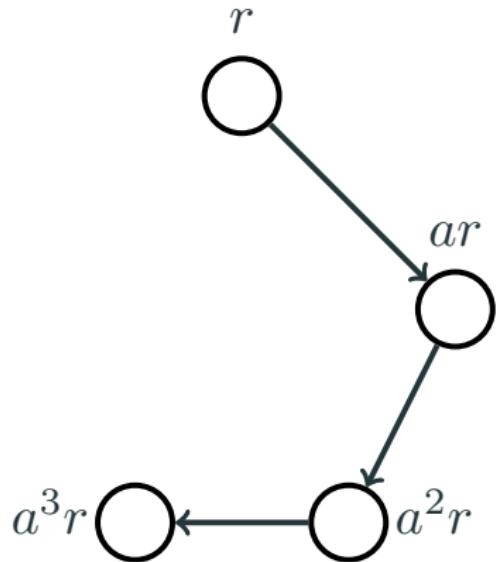
Proof

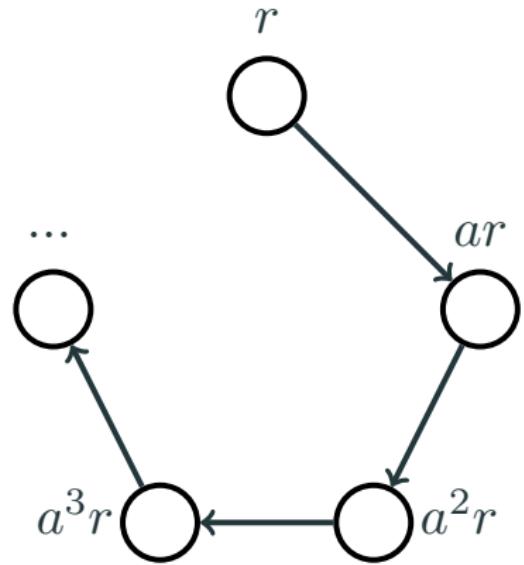
- Consider all $p - 1$ non-zero remainders modulo p : $1, 2, \dots, p - 1$
- Multiplying any such remainder by a is invertible since $p \nmid a$, so the new remainder is also non-zero
- Graph on remainders: edge from r to ar
- All incoming and outgoing degrees are 1
- What happens if we start with r , multiply it by a , take remainder modulo p , repeat several times?

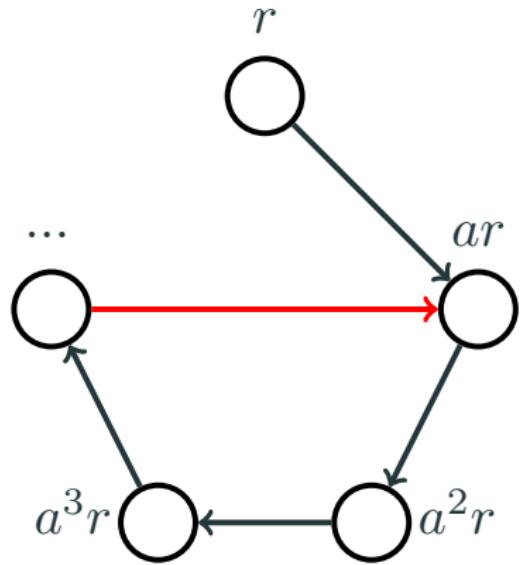
r 

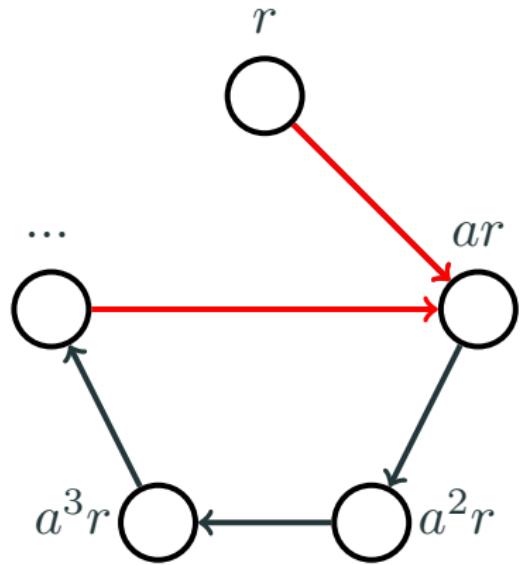


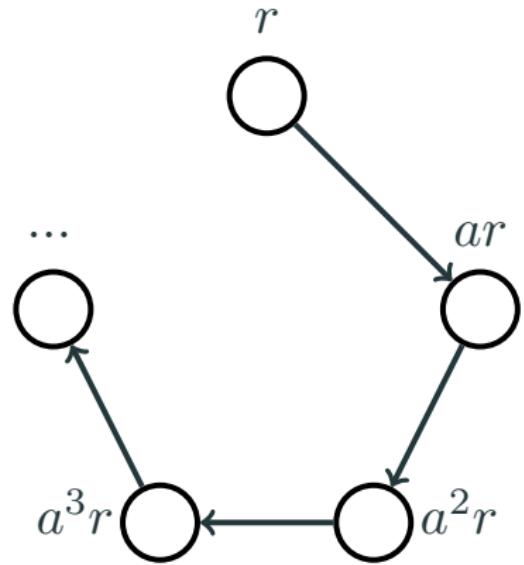


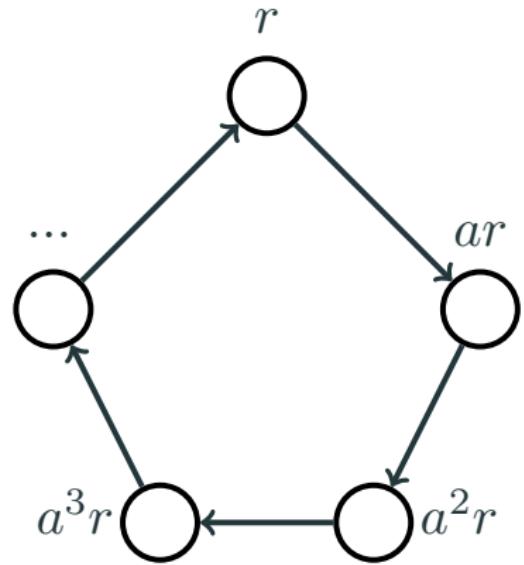


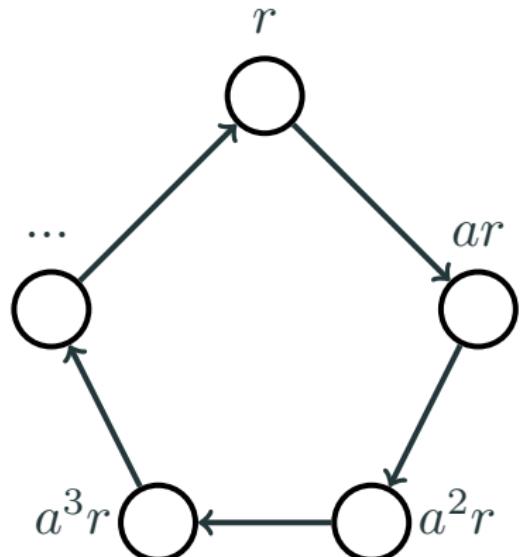




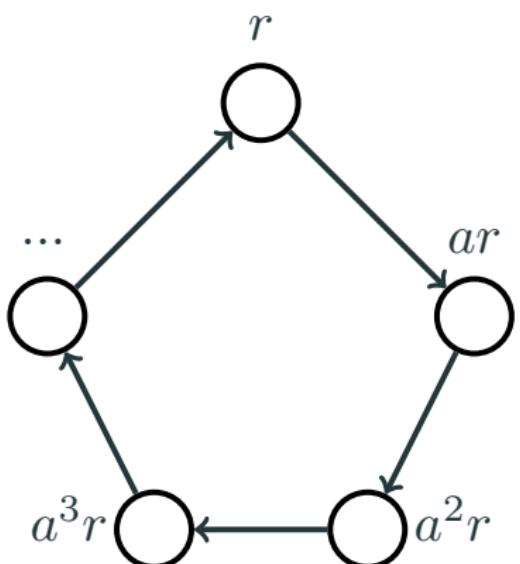




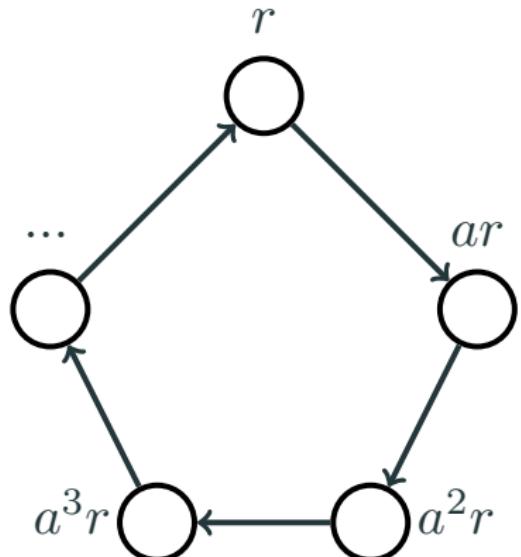




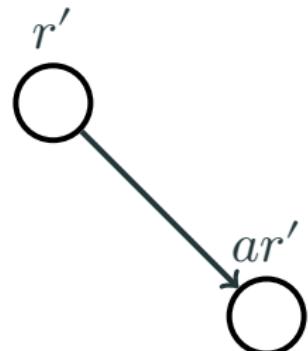
$$a^l \equiv 1 \pmod{p}$$

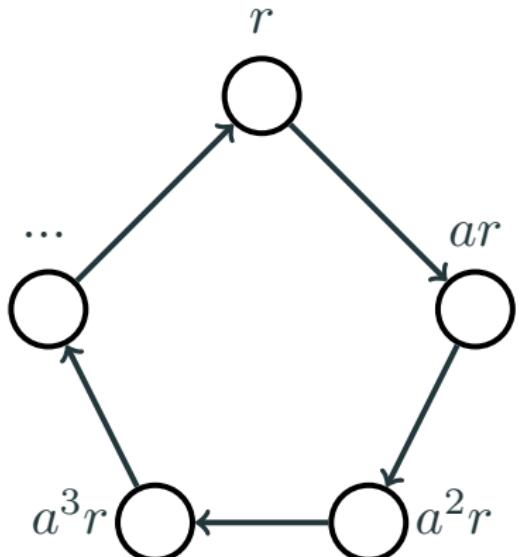


$$a^l \equiv 1 \pmod{p}$$

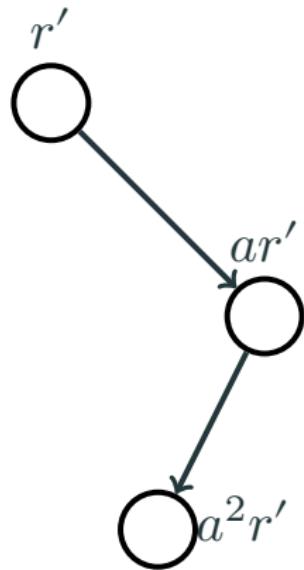


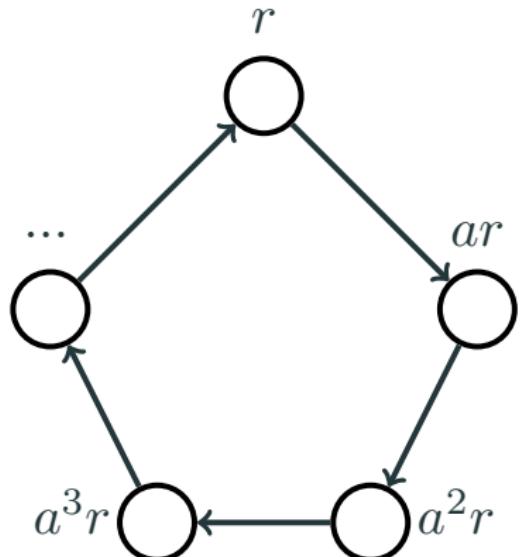
$$a^l \equiv 1 \pmod{p}$$



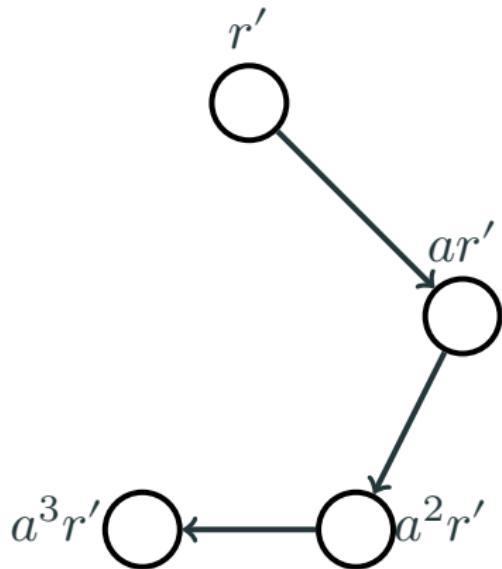


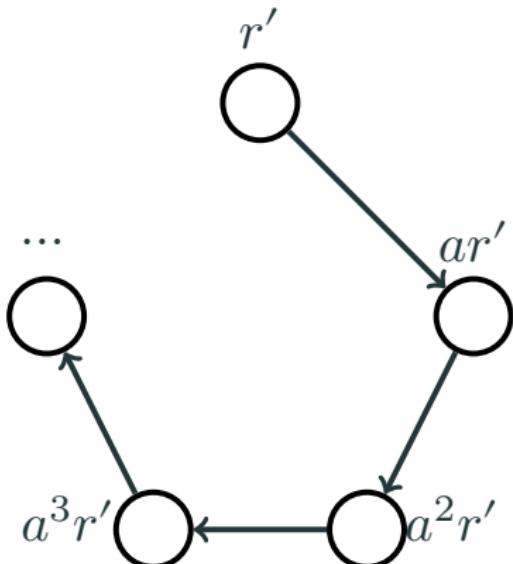
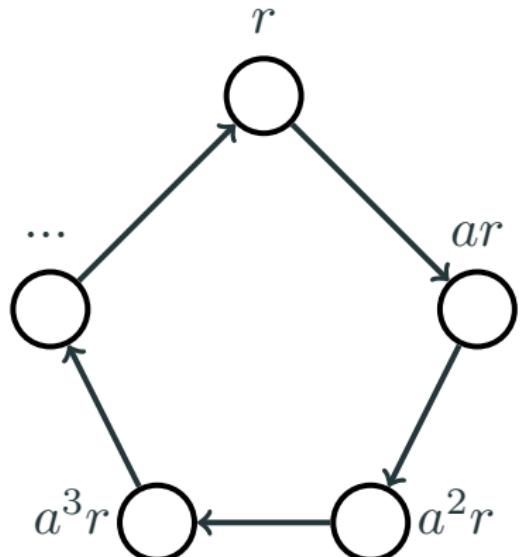
$$a^l \equiv 1 \pmod{p}$$



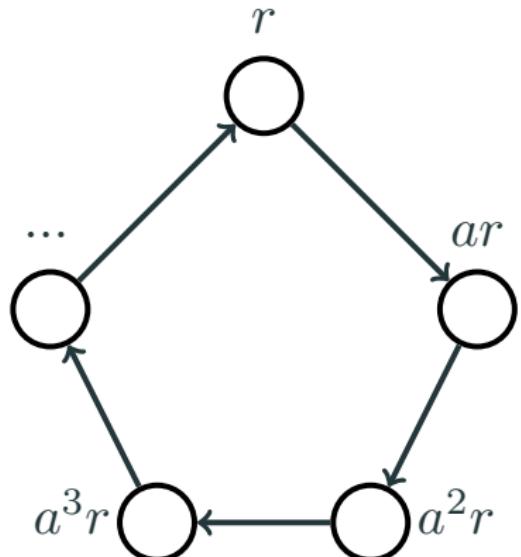


$$a^l \equiv 1 \pmod{p}$$

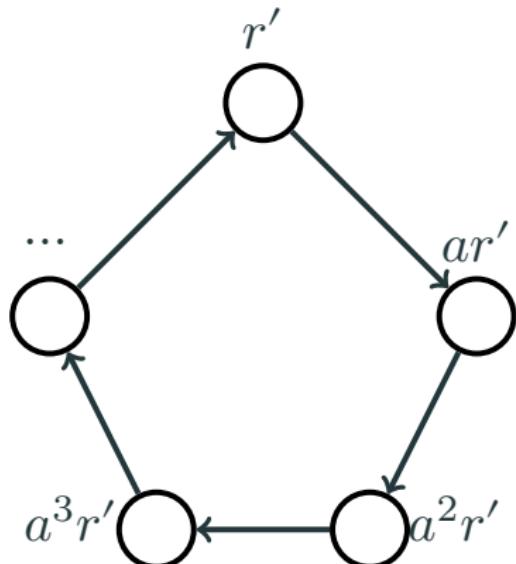




$$a^l \equiv 1 \pmod{p}$$



$$a^l \equiv 1 \pmod{p}$$



$$a^l \equiv 1 \pmod{p}$$

Proof

- Starting with any r , we get a cycle of some length l

Proof

- Starting with any r , we get a cycle of some length l
- This means $a^l \equiv 1 \pmod{p}$

Proof

- Starting with any r , we get a cycle of some length l
- This means $a^l \equiv 1 \pmod{p}$
- Starting with any r' we get a cycle of length l

Proof

- Starting with any r , we get a cycle of some length l
- This means $a^l \equiv 1 \pmod{p}$
- Starting with any r' we get a cycle of length l
- These cycles don't intersect and contain all $p - 1$ non-zero remainders

Proof

- Starting with any r , we get a cycle of some length l
- This means $a^l \equiv 1 \pmod{p}$
- Starting with any r' we get a cycle of length l
- These cycles don't intersect and contain all $p - 1$ non-zero remainders
- If there are c cycles, then $cl = p - 1$

Proof

- Starting with any r , we get a cycle of some length l
- This means $a^l \equiv 1 \pmod{p}$
- Starting with any r' we get a cycle of length l
- These cycles don't intersect and contain all $p - 1$ non-zero remainders
- If there are c cycles, then $cl = p - 1$
- $a^{p-1} = a^{cl} = (a^l)^c \equiv 1^c \equiv 1 \pmod{p}$

□

Optimizing Modular Exponentiation

- If $p \nmid a$, $a^{p-1} \equiv 1 \pmod{p}$
- $a^n \equiv a^{n \bmod (p-1)} \pmod{p}$
- If $p \mid a$, $a^n \equiv 0 \equiv a^{n \bmod (p-1)} \pmod{p}$
- For any a and n , $a^n \equiv a^{n \bmod (p-1)} \pmod{p}$
- Compute only powers up to $p - 1$

Outline

Fast Modular Exponentiation

Fermat's Little Theorem

Euler's Totient Function

Euler's Theorem

Euler's Totient Function

- Key for the RSA encryption
- Easy to compute if factorization of n is known
- No fast algorithms are known to compute if factorization of n is unknown
- No fast algorithms are known for factorization of integers
- Easy to compute with some private information, but no known way to compute without it — this is the key property for cryptography

Euler's Totient Function

Definition

Euler's totient function $\phi(n)$ counts integers between 0 and $n - 1$ which are coprime with n

$$n = 1$$

0 is coprime with $n = 1$, so $\phi(1) = 1$

$$n = 2$$

0 is not coprime with 2, and 1 is coprime with 2, so

$$\phi(2) = 1$$

$$n = 3$$

1 and 2 are coprime with 3, so $\phi(3) = 2$

$$n = 10$$

1, 3, 7 and 9 are coprime with 10, so $\phi(10) = 4$

Lemma

If p is prime, $\phi(p) = p - 1$.

Proof

0 is not coprime with p , and $1, 2, 3, \dots, p - 1$ are coprime with p , so $\phi(p) = p - 1$.



Lemma

If p and q are prime, then $\phi(pq) = (p - 1)(q - 1)$.

Proof

- Consider all pq remainders modulo pq
- By Chinese Remainder Theorem, each r corresponds to pair (r_p, r_q)
- $\text{GCD}(pq, r) = 1 \Leftrightarrow r_p, r_q \neq 0$

	0	1	2	...	$q - 1$	
0						
1						
2						
...						
$p - 1$						

Proof

- Consider all pq remainders modulo pq
- By Chinese Remainder Theorem, each r corresponds to pair (r_p, r_q)
- $\text{GCD}(pq, r) = 1 \Leftrightarrow r_p, r_q \neq 0$

	0	1	2	...	$q - 1$	
0
1
2
...
$p - 1$

Proof

- Consider all pq remainders modulo pq
- By Chinese Remainder Theorem, each r corresponds to pair (r_p, r_q)
- $\text{GCD}(pq, r) = 1 \Leftrightarrow r_p, r_q \neq 0$

	0	1	2	...	$q - 1$	
0						
1
2
...
$p - 1$

$(p - 1)(q - 1)$
such (r_p, r_q)

Outline

Fast Modular Exponentiation

Fermat's Little Theorem

Euler's Totient Function

Euler's Theorem

Euler's Theorem

- Generalization of Fermat's Little Theorem
- Together with modular exponentiation, is used to encrypt and decrypt in RSA

Euler's Theorem

Theorem

If a is coprime with n , $a^{\phi(n)} \equiv 1 \pmod{n}$.

Proof

- Very similar to Fermat's Little Theorem
- Consider all $\phi(n)$ remainders modulo n which are coprime with n
- Multiplying by a is invertible, so new remainder is also coprime with n
- Multiplying some r by a many times leads to cycle of length l
- $a^l \equiv 1 \pmod{n}$

Proof

- $a^l \equiv 1 \pmod{n}$
- All cycle lengths are the same
- Cycles don't intersect and cover all remainders coprime with n
- If there are c cycles, then $cl = \phi(n)$
- $a^{\phi(n)} = a^{cl} = (a^l)^c \equiv 1^c \equiv 1 \pmod{n}$ □

Conclusion

- Defined modular exponentiation

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation
- Proved Fermat's Little Theorem

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation
- Proved Fermat's Little Theorem
- Defined Euler's totient function

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation
- Proved Fermat's Little Theorem
- Defined Euler's totient function
- Proved Euler's theorem

Conclusion

- Defined **modular exponentiation**
- Designed fast algorithm to compute **modular exponentiation**
- Proved Fermat's Little Theorem
- Defined **Euler's totient function**
- Proved Euler's theorem
- Next module — learn public key cryptography and break some secret codes yourself!