MATHEMATICS IN OI (I)

VINCENT CHIU

WELCOME

to the World of

Number Theory

“

Mathematics
is the queen of
the sciences — and

number theory
is the queen of
mathematics

”

GREEK ALPHABET

Αα	Ββ	Γγ	Δδ	Εε	Ζζ
ALPHA [a] *ἄλφα*	**BETA** [b] *βῆτα*	**GAMMA** [g] *γάμμα*	**DELTA** [d] *δέλτα*	**EPSILON** [e] *ἒ ψιλόν*	**ZETA** [dz] *ζῆτα*
Ηη	Θθ	Ιι	Κκ	Λλ	Μμ
ETA [ɛː] *ἦτα*	**THETA** [tʰ] *θῆτα*	**IOTA** [i] *ἰῶτα*	**KAPPA** [k] *κάππα*	**LAMBDA** [l] *λάμβδα*	**MU** [m] *μῦ*
Νν	Ξξ	Οο	Ππ	Ρρ	Σσς
NU [n] *νῦ*	**XI** [ks] *ξεῖ*	**OMICRON** [o] *ὂ μικρόν*	**PI** [p] *πεῖ*	**RHO** [r] *ῥῶ*	**SIGMA** [s] *σῖγμα*
Ττ	Υυ	Φφ	Χχ	Ψψ	Ωω
TAU [t] *ταῦ*	**UPSILON** [u] *ὐ ψιλόν*	**PHI** [pʰ] *φεῖ*	**CHI** [kʰ] *χεῖ*	**PSI** [ps] *ψεῖ*	**OMEGA** [ɔː] *ὠ μέγα*

CONTENT

- More Mathematical Notations
- Modular Arithmetic
- Fast Exponential
- Greatest Common Divisor (GCD)
- Modular Division
- Euclidean Algorithm
- Extended Euclidean Algorithm
- Modular Inverse
- Prime Numbers

MORE MATHEMATICAL NOTATIONS – CAPITAL SIGMA Σ

- Sigma: σ (lower case), Σ (upper case)
- For any integers $m \leq n$,

$$\sum_{i=m}^{n} a_i = a_m + a_{m+1} + \cdots + a_{n-1} + a_n$$

- e.g. $\sum_{i=4}^{7} i^2 = 4^2 + 5^2 + 6^2 + 7^2 = 126$
- Called the Summation Sign

MORE MATHEMATICAL NOTATIONS – CAPITAL SIGMA Σ

- Task 1.1: Compute the following:

$$\sum_{i=20}^{30} (i^2 + 4i - 7)$$

MORE MATHEMATICAL NOTATIONS – CAPITAL SIGMA Σ

- Task 1.1: Compute the following:

$$\sum_{i=20}^{30} (i^2 + 4i - 7)$$

- Code:

```
int sum = 0;
for(int i = 20; i <= 30; i++) sum += i * i + 4 * i - 7;
return sum;
```

MORE MATHEMATICAL NOTATIONS – CAPITAL SIGMA Σ

- Task 1.1: Compute the following:

$$\sum_{i=20}^{30} (i^2 + 4i - 7)$$

- Code:

```
int sum = 0;
for(int i = 20; i <= 30; i++) sum += i * i + 4 * i - 7;
return sum;
```

Initialization

MORE MATHEMATICAL NOTATIONS – CAPITAL SIGMA Σ

- Task 1.1: Compute the following:

$$\sum_{i=20}^{30} (i^2 + 4i - 7)$$

- Code:

Lower bound

```
int sum = 0;
for(int i = 20; i <= 30; i++) sum += i * i + 4 * i - 7;
return sum;
```

MORE MATHEMATICAL NOTATIONS – CAPITAL SIGMA Σ

- Task 1.1: Compute the following:

$$\sum_{i=20}^{30} (i^2 + 4i - 7)$$

Upper bound

- Code:

```
int sum = 0;
for(int i = 20; i <= 30; i++) sum += i * i + 4 * i - 7;
return sum;
```

MORE MATHEMATICAL NOTATIONS – CAPITAL SIGMA Σ

- Task 1.1: Compute the following:

$$\sum_{i=20}^{30} (i^2 + 4i - 7)$$

Expression inside summation

- Code:

```
int sum = 0;
for(int i = 20; i <= 30; i++) sum += i * i + 4 * i - 7;
return sum;
```

MORE MATHEMATICAL NOTATIONS – CAPITAL PI $\prod$

- Similar to Capital Sigma,
- For any integers $m \leq n$

$$\prod_{i=m}^{n} a_i = a_m \times a_{m+1} \times \cdots \times a_{n-1} \times a_n$$

- Example: $\prod_{i=4}^{7} i^2 = 4^2 \times 5^2 \times 6^2 \times 7^2 = 705600$
- Called the Product Sign

MORE MATHEMATICAL NOTATIONS – CAPITAL PI Π

- The most common example: Factorial

$$n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$$

- Note: $0! = 1$ by definition
- For any positive integer n,

$$n! = \prod_{i=1}^{n} i$$

- Task 1.2: Input n $(0 \leq \mathrm{n} \leq 12)$, output $n!$

MORE MATHEMATICAL NOTATIONS – VERTICAL BAR | ∤

- The Divisibility Sign
- $a \mid n$ means "a divides n", "n is divisible by a", and "a is a factor of n"
- e.g. $2|4, 3|15, 5|15$
- $a \nmid n$ means "a does not divide n"

MORE MATHEMATICAL NOTATIONS – OTHER NOTATIONS AND ABBREVIATIONS

- $\forall$: for all
- $\exists$: there exists
- $s.\ t.$: such that
- $\in$: is an element of
- $\mathbb{N}$: set of natural numbers (i.e. positive integers)
- $\mathbb{Z}$: set of integers
- $\wedge$: logical AND
- $\vee$: logical OR

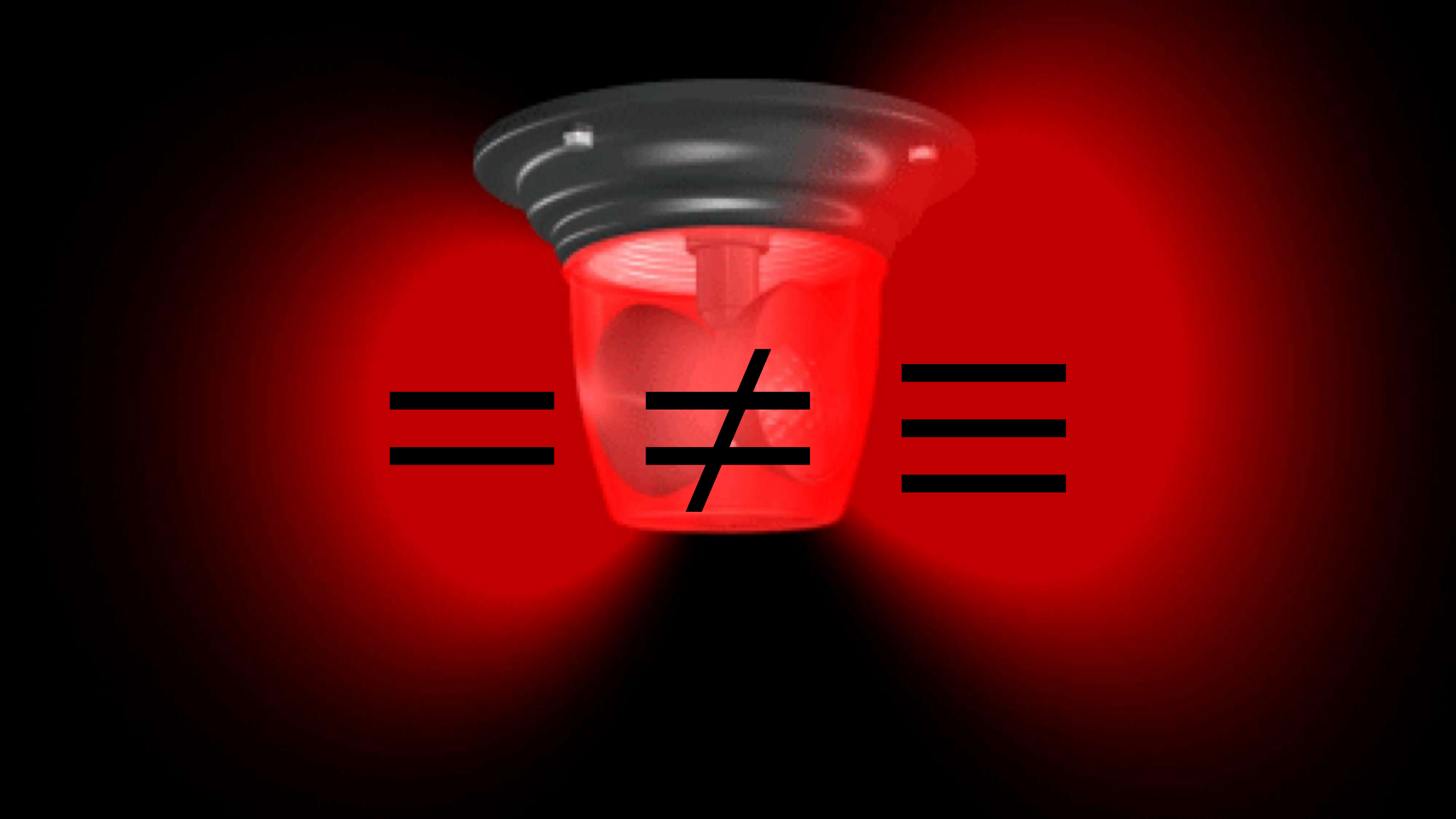
CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic
- Fast Exponential
- Greatest Common Divisor (GCD)
- Modular Division
- Euclidean Algorithm
- Extended Euclidean Algorithm
- Modular Inverse
- Prime Numbers

MODULAR ARITHMETIC – INTRODUCTION

- Recall what you have learnt in primary about division
- $13 \div 5 = 2 \ldots 3$ (Dividend ÷ Divisor = Quotient … Remainder)
- So, we say the remainder of $13 \div 5$ is 3

- Now, we may say $13 \equiv 3 \ (mod\ 5)$
- ”13 and 3 are congruent modulo 5”



MODULAR ARITHMETIC – NOTATION

- Generally, if a and b are congruent modulo m where a, b are integers, m is a positive integer, we we write $a \equiv b \ (mod \ m)$
- Translate into programming language:
  - Pascal: $a \ mod \ m = b \ mod \ m$
  - C/C++: $a \ \% \ m == b \ \% \ m$
- The remainder of the division of both a and b by m are the same

MODULAR ARITHMETIC – CAUTION

- Take care of the sign of the dividend when writing programs.
- If the dividend is negative, the remainder becomes non-positive!
- $-5 \equiv 2\ (mod\ 7)$, but $-5\ \%\ 7$ gives result -5
- Hence, you may need to write this:

$$\big((a\ \%\ b) + b\big)\ \%\ b$$

MODULAR ARITHMETIC – MEANING

- Instead of saying a and b have the same remainder when divided by m
- We are more often to say that
- $a - b$ is divisible by m.
  - Mathematical notation: $m|(a - b)$
- $\exists k \in \mathbb{Z}\, s.t.\, a = km + b$
  - There exists some integer k such that $a = km + b$

MODULAR ARITHMETIC – ADDITION / SUBTRACTION

$$a \equiv b \ (mod\ m), x \in \mathbb{Z}$$
$$\Rightarrow a + x \equiv b + x \ (mod\ m)$$

- Proof:

$$a \equiv b \ (mod\ m)$$
$$\Rightarrow \exists k \in \mathbb{Z}\ s.t.\ a = km + b$$
$$a + x = km + (b + x) \blacksquare$$

MODULAR ARITHMETIC – ADDITION / SUBTRACTION

$$a \equiv b\ (mod\ m), c \equiv d\ (mod\ m)$$
$$\Rightarrow a \pm c \equiv b \pm d\ (mod\ m)$$

- Proof:

$$a \equiv b\ (mod\ m), c \equiv d\ (mod\ m)$$

$$\Rightarrow \exists k, l \in \mathbb{Z}\ s.t.\ a = km + b \bigwedge c = lm + d$$

$$a \pm c = (k \pm l)m + (b \pm d), (k \pm l) \in \mathbb{Z} \blacksquare$$

MODULAR ARITHMETIC – MULTIPLICATION

$$a \equiv b \ (mod\ m), x \in \mathbb{Z}$$
$$\Rightarrow ax \equiv bx \ (mod\ m)$$

- Proof:

$$a \equiv b \ (mod\ m)$$
$$\Rightarrow \exists k \in \mathbb{Z}\ s.t.\ a = km + b$$
$$ax = (km + b)x = (kx)m + bx, kx \in \mathbb{Z} \blacksquare$$

- In fact, we have

$$ax = k(mx) + bx \Rightarrow ax \equiv bx \ (mod\ mx) \ (if\ x > 0)$$

MODULAR ARITHMETIC – MULTIPLICATION

$$a \equiv b\ (mod\ m), c \equiv d\ (mod\ m)$$
$$\Rightarrow ac \equiv bd\ (mod\ m)$$

- Proof:

$$a \equiv b\ (mod\ m), c \equiv d\ (mod\ m)$$

$$\Rightarrow \exists k, l \in \mathbb{Z}\ s.t.\ a = km + b \bigwedge c = lm + d$$

$$ac = (km + b)(lm + d) = (klm + bl + kd)m + bd$$
$$(klm + bl + kd) \in \mathbb{Z}\blacksquare$$

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential
- Greatest Common Divisor (GCD)
- Modular Division
- Euclidean Algorithm
- Extended Euclidean Algorithm
- Modular Inverse
- Prime Numbers

FAST EXPONENTIAL – PROBLEM

- Task 2: Given a, b, m, find $a^b \ (mod\ m)$
- Code #1:

```
int product = 1;
for(int i = 1; i <= b; i++) product = (product*a) % m;
return product;
```

- Time complexity: $O(b)$
- Not fast enough for big b
- Also, use `long long`!

FAST EXPONENTIAL – SUPP. INFO: HIGH PRECISION ARITHMETIC (H.P.A.)

- 32-bit integers: $-2^{31} \leq n < 2^{31}$
  - [-2,147,483,648, 2,147,483,647]
- 64-bit integers: $-2^{63} \leq n < 2^{63}$
  - [-9,223,372,036,854,775,808, 9,223,372,036,854,775,807]
- What if you are asked to process any integers which consists of ≥ 20 digits…

FAST EXPONENTIAL – SUPP. INFO: HIGH PRECISION ARITHMETIC (H.P.A.)

- Treat the number as a string. Use an array to store each digit.
- Operations:
  - Comparison
  - Addition
  - Subtraction
  - Multiplication (e.g. Factorial)
  - Division / Remainder
- Beware of Carrying & Borrowing! ☹

FAST EXPONENTIAL – SOLUTION

- Remember: law of indices: $a^m a^n = a^{m+n}$
- Solution: express b in binary form

$$b = \overline{b_k b_{k-1} \dots b_1 b_0} \text{ (in binary form, } b_i \in \{0,1\})$$

$$a^b = a^{\sum_{i=0}^{k} b_i 2^i} = \prod_{i=0}^{k} \left(a^{2^i}\right)^{b_i}$$

- Also known as "Big Mod"

FAST EXPONENTIAL – SOLUTION

- Time complexity: $O(\lg b)$
- Example: $7^{77}\ mod\ 17 = 6$
- Code: left as exercise ☺

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential ✓
- Greatest Common Divisor (GCD)
- Modular Division
- Euclidean Algorithm
- Extended Euclidean Algorithm
- Modular Inverse
- Prime Numbers

GREATEST COMMON DIVISOR – INTRODUCTION

- Greatest Common Divisor (GCD)
- Alternative name: Highest Common Factor (HCF)
- Definition of common divisor:
- If d divides both positive integers a and b $(d|a \wedge d|b)$, then d is a common divisor of a and b

GREATEST COMMON DIVISOR – DEFINITION

- Definition of greatest common divisor (GCD):
- Among all common divisors of positive integers a and b, the largest one is called the greatest common divisor
- Notation: We use $\gcd(a, b)$ to represent the greatest common divisor
  - Some people just write (a, b)
  - Not to be mixed with coordinates

GREATEST COMMON DIVISOR – PROPERTY

$$k|a \bigwedge k|b \Rightarrow k|\gcd(a,b)$$

- If k is a common divisor of a and b, then k divides $\gcd(a,b)$
- Proof: Fundamental theorem of arithmetic: Every integer greater than 1 has a unique representation of product of prime numbers

GREATEST COMMON DIVISOR – PROPERTY

- Canonical representation of positive integer: $n = p_1^{q_1} p_2^{q_2} p_3^{q_3} \ldots = \prod_{i=1}^{\infty} p_i^{q_i}, p_i\ prime, q_i \in \mathbb{N} \cup \{0\}$
- $a = \prod_{i=1}^{\infty} p_i^{a_i}, b = \prod_{i=1}^{\infty} p_i^{b_i}, \gcd(a,b) = \prod_{i=1}^{\infty} p_i^{\min(a_i,b_i)}, k = \prod_{i=1}^{\infty} p_i^{k_i}$
- $k_i \leq a_i \wedge k_i \leq b_i \Rightarrow k_i \leq \min(a_i, b_i) \Rightarrow k | \gcd(a,b)$ ■

GREATEST COMMON DIVISOR – >2 NUMBERS

- What is $\gcd(a_1, a_2, \dots, a_n)$?
- Just calculate the gcd one by one (order does not matter):

$$\gcd(a_1, a_2, \dots, a_n) = \gcd(a_1, \gcd(a_2, \dots, \gcd(a_{n-2}, \gcd(a_{n-1}, a_n))))$$

GREATEST COMMON DIVISOR – APPLICATION

- Simplifying a fraction $\frac{a}{b}$

$$\frac{a}{b} = \frac{a \div \gcd(a,b)}{b \div \gcd(a,b)}$$

- Calculating the least common multiple (LCM)
  - Definition of common multiple:
    - If m is divisible by both positive integers a and b, i.e. $a|m \wedge b|m$, then m is a common multiple of a and b

GREATEST COMMON DIVISOR – APPLICATION

- Calculating the least common multiple (LCM)
  - Definition of least common multiple:
    - Among all common multiples of positive integers a and b, the smallest one is called the least common multiple
  - Notation: we use $lcm(a, b)$ to represent the least common multiple. Call this m
  - Property 1: If k is a common multiple of a and b, then $m|k$
    - Proof: Let $k = qm + r$ for $0 \leq r < m$
    - $a|k \wedge b|k \wedge a|m \wedge b|m \Rightarrow a|(k - qm) = r \wedge b|r$
    - $0 \leq r < m = lcm(a, b) \Rightarrow r = 0 \Rightarrow m|k \blacksquare$

GREATEST COMMON DIVISOR – APPLICATION

- Calculating the least common multiple (LCM)
  - Property 2: the relationship between lcm and gcd:

$$ab = \gcd(a,b) \times lcm(a,b)$$

    - Proof: Fundamental theorem of arithmetic: Every integer greater than 1 has a unique representation of product of prime numbers
    - Canonical representation of positive integer: $n = p_1^{q_1} p_2^{q_2} p_3^{q_3} \ldots = \prod_{i=1}^{\infty} p_i^{q_i}, p_i\ prime, q_i \in \mathbb{N} \cup \{0\}$
    - $a = \prod_{i=1}^{\infty} p_i^{a_i}, b = \prod_{i=1}^{\infty} p_i^{b_i}, \gcd(a,b) = \prod_{i=1}^{\infty} p_i^{\min(a_i,b_i)}, lcm(a,b) = \prod_{i=1}^{\infty} p_i^{max(a_i,b_i)}$
    - $a_i + b_i = \min(a_i,b_i) + \max(a_i,b_i)$ ■

GREATEST COMMON DIVISOR – APPLICATION

- Calculating the least common multiple (LCM)
  - What is $lcm(a_1, a_2, \dots, a_n)$?
  - Just calculate the lcm one by one (order does not matter):

$$lcm(a_1, a_2, \dots, a_n) = lcm\left(a_1, lcm\left(a_2, \dots, lcm\left(a_{n-2}, lcm(a_{n-1}, a_n)\right)\right)\right)$$

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential ✓
- Greatest Common Divisor (GCD) ✓
- Modular Division
- Euclidean Algorithm
- Extended Euclidean Algorithm
- Modular Inverse
- Prime Numbers

MODULAR DIVISION – INTRODUCTION

- We have already talked about modular addition, subtraction, and multiplication.
- Except the modulo and congruent symbol, these operations are almost the same as what we used to do in normal arithmetic operations
- However, this is not the case for modular division
- For any integers c, if $ac \equiv bc \ (mod\ m)$, it is not necessary that $a \equiv b \ (mod\ m)$
- WHY? Any counter-example?

MODULAR DIVISION – EXAMPLE

- $6 \equiv 2 \ (mod \ 4)$ but $3 \not\equiv 1 \ (mod \ 4)$
- $ac \equiv bc \ (mod \ m) \Rightarrow a \equiv b \ (mod \ m)$: when does this hold?

MODULAR DIVISION – PROPERTIES

- Suppose

$$ac \equiv bc \ (mod\ mc)$$

$$\exists k \in \mathbb{Z}\ s.t.\ ac = kmc + bc$$

$$a = k\left(\frac{mc}{c}\right) + b = km + b$$

$$\Rightarrow a \equiv b \ (mod\ m)$$

MODULAR DIVISION – PROPERTIES

- Now suppose

$$ac \equiv bc \ (mod\ m)$$

$$\exists k \in \mathbb{Z}\ s.t.\ ac = km + bc$$

$$a = \frac{k}{c}m + b$$

- So if $\frac{k}{c}$ is an integer, then we will have $a \equiv b \ (mod\ m)$
- This is true when c and m are co-prime, i.e. $\gcd(c, m) = 1$

MODULAR DIVISION – PROPERTIES

- Proof:
- Euclid's lemma
  - $p \nmid a \wedge p \nmid b \Rightarrow p \nmid ab$
  - Contrapositive: $p|ab \Rightarrow p|a$ and/or $p|b$
  - + Fundamental Theorem of Arithmetic ⇒ Generalized theorem:
  - $n|ab \wedge \gcd(n, a) = 1 \Rightarrow n|b$

MODULAR DIVISION – PROPERTIES

- Proof (cont.):

$$ac \equiv bc \ (mod\ m)$$
$$\exists k \in \mathbb{Z}\ s.t.\ ac = km + bc$$
$$c|(ac - bc) = km$$
$$\gcd(c, m) = 1 \Rightarrow c|k \blacksquare$$

MODULAR DIVISION – PROPERTIES

- Generalize:

$$ac \equiv bc \ (mod\ m)$$

$$\exists k \in \mathbb{Z}\ s.t.\ ac = km + bc$$

$$a\left(\frac{c}{\gcd(c,m)}\right) = k\left(\frac{m}{\gcd(c,m)}\right) + b\left(\frac{c}{\gcd(c,m)}\right)$$

$$a\left(\frac{c}{\gcd(c,m)}\right) \equiv b\left(\frac{c}{\gcd(c,m)}\right)\left(mod\ \frac{m}{\gcd(c,m)}\right)$$

MODULAR DIVISION – PROPERTIES

- Since $\frac{c}{\gcd(c,m)}$ and $\frac{m}{\gcd(c,m)}$ must be co-prime
- We have $a \equiv b \left(mod \frac{m}{\gcd(c,m)}\right)$

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential ✓
- Greatest Common Divisor (GCD) ✓
- Modular Division ✓
- Euclidean Algorithm
- Extended Euclidean Algorithm
- Modular Inverse
- Prime Numbers

A SHORT BREAK

+Take Attendance

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential ✓
- Greatest Common Divisor (GCD) ✓
- Modular Division ✓
- Euclidean Algorithm
- Extended Euclidean Algorithm
- Modular Inverse
- Prime Numbers

EUCLIDEAN ALGORITHM – PROBLEM

- Task 3: Input two positive integers a and b, output $\gcd(a, b)$
- Code #1:

```
int gcd = 1;
for(int i = 1; i <= a && i <= b; i++)
        if(a % i == 0 && b % i == 0) gcd = i;
return gcd;
```

- Time complexity: $O(\min(a, b))$
- Not fast enough for big a, b ☹

EUCLIDEAN ALGORITHM – PROBLEM

- It is not efficient to find the gcd using brute force, we need some fast algorithm
- We can use Euclidean Algorithm to speed up ☺

EUCLIDEAN ALGORITHM – CONCEPT

- For any positive integers a, b,
- If $a = qb + r$ where $0 \leq r < b$ (i.e. $r = a \,\%\, b$)
- Then $\gcd(a, b) = \gcd(b, r)$

EUCLIDEAN ALGORITHM – CONCEPT

- Proof: Let $d = \gcd(a, b)$ and $e = \gcd(b, r)$ and $a = qb + r$ for $0 \leq r < b$

$$d = \gcd(a, b) \Rightarrow d|a \bigwedge d|b \Rightarrow d|(a - qb) = r$$

$$\Rightarrow d|\gcd(b, r) \Rightarrow d|e \ (\Rightarrow d \leq e)$$

$$e = \gcd(b, r) \Rightarrow e|b \bigwedge e|r \Rightarrow e|(qb + r) = a$$

$$\Rightarrow e|\gcd(a, b) \Rightarrow e|d \ (\Rightarrow e \leq d)$$

- Since $d|e$ and $e|d$, we have $d = e$, i.e. $\gcd(a, b) = \gcd(b, r)$ ■

EUCLIDEAN ALGORITHM – PROCEDURE

- For any positive integers a, b, let $r = a \,\%\, b$ (or $r = a \; mod \; b$), then
$$\gcd(a, b) = \gcd(b, r)$$
- Iterate this process until $r = 0$, then the gcd is found.
- We can do this using recursion!

EUCLIDEAN ALGORITHM – PROCEDURE

- Proof of validity of Euclidean Algorithm in finding gcd
- Consider what we have done in the recursion Euclidean Algorithm;

$$a = q_0 b + r_0, 0 \leq r_0 < b, \gcd(a, b) = \gcd(b, r_0)$$
$$b = q_1 r_0 + r_1, 0 \leq r_1 < r_0, \gcd(b, r_0) = \gcd(r_0, r_1)$$
$$\dots$$
$$r_{i-2} = q_i r_{i-1} + r_i, 0 \leq r_i < r_{i-1}, \gcd(r_{i-2}, r_{i-1}) = \gcd(r_{i-1}, r_i)$$

- Eventually we get this relationship:

$$0 \leq r_i < r_{i-1} < \cdots < r_1 < r_0 < b$$

EUCLIDEAN ALGORITHM – PROCEDURE

$$0 \leq r_i < r_{i-1} < \cdots < r_1 < r_0 < b$$

- This is a strictly decreasing sequence.
- Then there MUST exist $N \in \mathbb{N}\ s.t. r_N = 0$. Then

$r_{N-2} = q_N r_{N-1} + r_N = q_N r_{N-1} \Rightarrow r_{N-1} | r_{N-2} \Rightarrow \gcd(a, b) = \gcd(r_{N-2}, r_{N-1}) = r_{N-1}$ ■

EUCLIDEAN ALGORITHM – PROCEDURE

- Code #2:

```
long long gcd(long long a, long long b) {
        if(b == 0) return a;
        else return gcd(b, a % b);
}
```

- Example: $\gcd(1239,4557) = 21$
- Time complexity: $O(\lg n)$
- But what are the worse cases?

EUCLIDEAN ALGORITHM – PROCEDURE

- Worse cases:
  - Since $r = a - qb$, the q should be as small as possible
  - q will not be 0, so the smallest possible q is 1
  - If all the q are 1, then that will be a worse case
- Worst case: Contiguous 2 Fibonacci Numbers
  - $a = F_{87} = 679{,}891{,}637{,}638{,}612{,}258$
  - $b = F_{86} = 420{,}196{,}140{,}727{,}489{,}673$

EUCLIDEAN ALGORITHM – BINARY VERSION

- If both a and b are even, $\gcd(a, b) = 2 \times \gcd\left(\frac{a}{2}, \frac{b}{2}\right)$
- If a is odd, while b is even, $\gcd(a, b) = \gcd\left(a, \frac{b}{2}\right)$
- Similarly, if b is odd, while a is even, $\gcd(a, b) = \gcd\left(\frac{a}{2}, b\right)$
- If both a and b are odd, then $\gcd(a, b) = \gcd(a - b, b) = \gcd\left(\frac{a-b}{2}, b\right)$

EUCLIDEAN ALGORITHM – BINARY VERSION

- Terminating case: when one of a or b equals 0, return another variable with non-zero value
- You may explore more about this algorithm, but this is not our main concern today

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential ✓
- Greatest Common Divisor (GCD) ✓
- Modular Division ✓
- Euclidean Algorithm ✓
- Extended Euclidean Algorithm
- Modular Inverse
- Prime Numbers

EXTENDED EUCLIDEAN ALGORITHM – INTRODUCTION

- Task 4: Find the integral solution of x and y such that $ax + by = \gcd(a, b)$
- It is guaranteed that a solution exists

EXTENDED EUCLIDEAN ALGORITHM – PROCEDURE

- Proof: as what the title tells you, this problem is an extension to Euclidean Algorithm:
- Consider what we have done in Euclidean Algorithm in finding $\gcd(a, b)$;

$$a = q_0 b + r_0, 0 \leq r_0 < b, \gcd(a, b) = \gcd(b, r_0)$$

$$b = q_1 r_0 + r_1, 0 \leq r_1 < r_0, \gcd(b, r_0) = \gcd(r_0, r_1)$$

$$\dots$$

$$r_{i-2} = q_i r_{i-1} + r_i, 0 \leq r_i < r_{i-1}, \gcd(r_{i-2}, r_{i-1}) = \gcd(r_{i-1}, r_i)$$

$$\dots$$

$$r_{N-2} = q_N r_{N-1} + r_N, r_N = 0$$

EXTENDED EUCLIDEAN ALGORITHM – PROCEDURE

$$\gcd(a, b) = r_{N-1}$$

- Translate the previous equations to get the following:

$$r_0 = a - q_0 b$$

$$r_1 = b - q_1 r_0$$

$$\dots$$

$$r_i = r_{i-2} - q_i r_{i-1}$$

$$\gcd(a, b) = r_{N-1} = r_{N-3} - q_{N-2} r_{N-2}$$

EXTENDED EUCLIDEAN ALGORITHM – PROCEDURE

- After all the remainders r_i have been substituted, the final expression will express $\gcd(a, b)$ as a linear sum of a and b: $ax + by = \gcd(a, b)$
- Which means we need to backtrack each step of Euclidean Algorithm to get q_i and r_i
- Example: $\gcd(1239,4557) = 21$
- Integral solution of $1239x + 4557y = 21$: $(103 - 217\mathrm{k}, -28 + 59\mathrm{k})\ \forall k \in \mathbb{Z}$
- Integral solution of $1239x + 4557y = 21m$: $(103m - 217k, -28m + 59k)\ \forall k, m \in \mathbb{Z}$
- The equations of the form $ax + by = d$ are called linear Diophantine equations, and has infinitely many integer solutions iff $\gcd(a, b) \mid d$

REVISIT: GREATEST COMMON DIVISOR – PROPERTY

$$k|a \bigwedge k|b \Rightarrow k|\gcd(a,b)$$

- If k is a common divisor of a and b, then k divides $\gcd(a,b)$
- Proof: we have $ax + by = d$

$$k|a \bigwedge k|b \Rightarrow k|(ax + by) = d\blacksquare$$

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential ✓
- Greatest Common Divisor (GCD) ✓
- Modular Division ✓
- Euclidean Algorithm ✓
- Extended Euclidean Algorithm ✓
- Modular Inverse
- Prime Numbers

MODULAR INVERSE – INTRODUCTION

- What is the inverse of a generally:
- Answer: a^{-1} (Property: $a \times a^{-1} = 1$)
- For any integer a, you want to find an integer b such that $ab \equiv 1 \ (mod\ m)$
- We denote this as $b \equiv a^{-1} \ (mod\ m)$
- i.e. b is the **modular** inverse of a modulo m
- We can do division by using modular inverse

$$ax \equiv c \ (mod\ m) \Rightarrow a^{-1}ax \equiv x \equiv a^{-1}c \ (mod\ m)$$

MODULAR INVERSE – EXISTENCE

- For all integers a and positive integer m, is modular inverse of a modulo m always exist?
- NO.
  - If $a = 6, m = 4$, for any integer b, ab is always even
  - i.e. $ab \equiv 0 \; or \; 2 \; (mod \; 4) \Rightarrow ab \not\equiv 1 \; (mod \; 4)$
- Modular inverse exists if and only if a and m are co-prime, i.e. $\gcd(a, m) = 1$
- If m is a prime while a is not a multiple of m, then modular inverse of a exists

MODULAR INVERSE – EXTENDED EUCLIDEAN ALGORITHM

- How can we find modular inverse of a modulo m?
- Recall the Extended Euclidean Algorithm we have talked before:
- We can always at least one integral solution for x, y in linear Diophantine equations such that $ax + my = \gcd(a, m)$
- If $\gcd(a, m) = 1$, then

$$ax + my = 1 \Rightarrow ax = (-y)m + 1 \Rightarrow ax \equiv 1 \ (mod\ m)$$

$$a^{-1} \equiv x \ (mod\ m)$$

MODULAR INVERSE – FERMAT'S LITTLE THEOREM (FOR PRIME NUMBERS ONLY)

- If a is an integer and p is a prime, then

$$a^p \equiv a \ (mod\ p)$$

  - If a is a multiple of p, i.e. $p|a$, then obviously $a^p \equiv 0 \equiv a \ (mod\ p)$

- Main focus: if a Is not divisible by p, i.e. $p \nmid a$, then

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

- We want to use the result for modular inverse: if $p \nmid a$, then

$$a^{-1} \equiv a^{p-2} \ (mod\ p)$$

- One can use the fast exponential algorithm taught earlier to calculate this.

MODULAR INVERSE – FERMAT'S LITTLE THEOREM (FOR PRIME NUMBERS ONLY)

- Proof: Let $a \in \mathbb{N}$ and $p \nmid a$. Consider the sequence

$$a \bmod p, 2a \bmod p, \dots, (p-1)a \bmod p$$

- Claim: The elements in the sequence above with length $(p-1)$ are all DISTINCT.
  - Proof by contradiction: assume $\exists 1 \leq j < i < p \; s.t. \; ai \equiv aj \; (mod \; p)$

$$a(i-j) \equiv 0 \; (mod \; p) \Rightarrow p|a \; (rej.) \; or \; p|(i-j) \; (rej. since \; 0 < i-j < p) \blacksquare$$

- In other words, the sequence mentioned above is a rearrangement of sequence

$$1,2, \dots, p-1$$

MODULAR INVERSE – FERMAT'S LITTLE THEOREM (FOR PRIME NUMBERS ONLY)

- If we multiply together the numbers in each sequence, they must be congruent modulo p

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

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

$$a^{p-1} \equiv 1 \ (mod\ p) \quad or \quad (p-1)! \equiv 0 \ (mod\ p) \ (rej.) \blacksquare$$

- Example: $7^{77}\ mod\ 17 = 6$
- What about composite integers?

MODULAR INVERSE – EULER'S TOTIENT THEOREM

- Euler's totient function $\phi(n)$ is the number of integers k satisfying $1 \leq k \leq n$ and $\gcd(k, n) = 1$
- Euler's totient theorem: if $\gcd(a, n) = 1$, then

$$a^{\phi(n)} \equiv 1 \ (mod\ n)$$

- Fermat's Little Theorem is a special case, where $\phi(p) = p - 1$
- Proof: similar as that of Fermat's Little Theorem's

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential ✓
- Greatest Common Divisor (GCD) ✓
- Modular Division ✓
- Euclidean Algorithm ✓
- Extended Euclidean Algorithm ✓
- Modular Inverse ✓
- Prime Numbers

PRIME NUMBERS – INTRODUCTION

- Prime number p: A positive number LARGER THAN 1 which contains exactly 2 positive divisors.
  - Note that 1 is not a prime.
  - One reason: Fundamental Theorem of Arithmetic: Unique Prime Factorization, $1 = 1 \times 1 \times \cdots$
- Hence, if $p = ab$, where a and b are positive integers and $a \leq b$, then $a = 1$ and $b = p$
- Euclid's Lemma: If $p|ab$, then $p|a$ and/or $p|b$

PRIME NUMBERS – CHECK PRIME

- Task 5.1: How to check whether p is a prime? $1 \leq p \leq 10^9$
- Notice that: p only contains exactly two positive divisors 1 and p

PRIME NUMBERS – CHECK PRIME

- Code #1:

```
for(int i = 2; i < p; i++) if(p % i == 0) return false;
return true;
```

- Time complexity: $O(p)$
- Not fast enough for $1 \leq p \leq 10^9$
- Any suggestions?

PRIME NUMBERS – CHECK PRIME

- Code #2:

```
for(int i = 2; i * i <= p; i++)
        if(p % i == 0) return false;
return true;
```

- Time complexity: $O(\sqrt{p})$ ☺
- Why can we do this?
- Because if p is divisible by some k where $\sqrt{p} < k < p$
- Then p is also divisible by $\frac{p}{k}$ where $1 < \frac{p}{k} < \sqrt{p}$

PRIME NUMBERS – PRIME LIST

- Task 5.2: Generate a prime number list in the range of $[1, 10^6]$
- Code #2*:

```
for(int j = 1; j <= 1000000; j++)
        if(is_prime(j)) output j; //Using Code #2
```

- Time complexity: $O(n\sqrt{n})$ ☹

PRIME NUMBERS – SIEVE OF ERATOSTHENES

- For any composite number (i.e. non-prime number greater than 1), it is divisible by some prime number.
- For any prime number, the only prime divisor is itself.
- We may store a list of prime numbers we have found.
- For each integer, check if it is divided by any of prime numbers found.
- If not, then it means it is a prime, and add it to the prime list.
- We may store an array of Boolean values which indicates whether i is a known composite number

PRIME NUMBERS – SIEVE OF ERATOSTHENES

- Code #3:

```
// initialize integer array composite[] = 0;
for(int i = 2; i <= n; i++) {
      if(comp[i] == false){
            //output i OR store it into prime list;
            for(int j = i * i; j <= n; j += i)
                  comp[j]=true;
      }
}
```

PRIME NUMBERS – SIEVE OF ERATOSTHENES

- Using Sieve of Eratosthenes:
- When we mark an integer as composite, store the current prime divisor i (it is the smallest prime divisor of j)
- For each integer, we can get the smallest prime divisor instantly
- Get the factorization recursively

REVISIT: MODULAR INVERSE – EULER'S TOTIENT THEOREM

- How to calculate Euler's totient function $\phi(n)$?
- Notice the pattern in Sieve of Eratosthenes
- For each prime number p, every multiple of p is crossed out
- Prime numbers does not affect the "frequency" of other primes cancelling out terms

REVISIT: MODULAR INVERSE – EULER'S TOTIENT THEOREM

- Do this for the prime divisors of n

$$n = \prod_{i=1}^{k} p_i^{q_i} \ (q_i > 0) \Rightarrow \phi(n) = n \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right)$$

$$\left(or\ equivalently, \phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)\right)$$

CONTENT

- More Mathematical Notations ✓
- Modular Arithmetic ✓
- Fast Exponential ✓
- Greatest Common Divisor (GCD) ✓
- Modular Division ✓
- Euclidean Algorithm ✓
- Extended Euclidean Algorithm ✓
- Modular Inverse ✓
- Prime Numbers ✓

REFERENCE

- Wikipedia
- Past Mathematics in OI (I) slides – 2015-18