# Mathematics in OI (I)

Charlie Li

# Table of content

# Mathematical notations (Capital Sigma)

- Sigma : Greek Alphabet σ (lower case), Σ (capital)
- Also, there are many other Greek alphabets are used in mathematics and science

- For example:
- Lower case Pi π (the ratio of circumference to diameter)
- Euler Phi Function ϕ(n) (no. of positive integers less than n which are co-prime to n) (This was taught in Math in OI (II) two weeks ago)
- Omega Ω(n) (Number of prime divisors of n)

# Mathematical notations (Capital Sigma)

- For any integers $m \leq n$,

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

- Example:

$$\sum_{i=4}^{7} i^2 = 4^2 + 5^2 + 6^2 + 7^2 = 16 + 25 + 36 + 49 = 126$$

- Capital Sigma is also called ”Summation Symbol”.

# Mathematical notations (Capital Sigma)

- Task 1.1: Compute the following:

$$\sum_{i=10}^{20} (i^2 - 5i + 2)$$

# Mathematical notations (Capital Sigma)

- Task 1.1: Compute the following:

$$\sum_{i=10}^{20} (i^2 - 5i + 2)$$

```
int sum = 0;
for (int i = 10; i <= 20; i++)
    sum += i * i – 5 * i + 2;
return sum;
```

# Mathematical notations (Capital Sigma)

- Task 1.1: Compute the following:

$$\sum_{i=10}^{20} (i^2 - 5i + 2)$$

```
int sum = 0;
for (int i = 10; i <= 20; i++)
    sum += i * i - 5 * i + 2;
return sum;
```

Initialization

# Mathematical notations (Capital Sigma)

- Task 1.1: Compute the following:

$$\sum_{i=10}^{20} (i^2 - 5i + 2)$$

Lower bound

```
int sum = 0;
for (int i = 10; i <= 20; i++)
    sum += i * i - 5 * i + 2;
return sum;
```

# Mathematical notations (Capital Sigma)

- Task 1.1: Compute the following:

$$\sum_{i=10}^{20} (i^2 - 5i + 2)$$

Upper bound

```
int sum = 0;
for (int i = 10; i <= 20; i++)
    sum += i * i - 5 * i + 2;
return sum;
```

# Mathematical notations (Capital Sigma)

- Task 1.1: Compute the following:

$$\sum_{i=10}^{20} (i^2 - 5i + 2)$$

Value corresponded to i

```
int sum = 0;
for (int i = 10; i <= 20; i++)
    sum += i * i - 5 * i + 2;
return sum;
```

# Mathematical notations (Capital Pi)

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

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

- Example:

$$\prod_{i=4}^{7} i^2 = 4^2 \cdot 5^2 \cdot 6^2 \cdot 7^2 = 16 \cdot 25 \cdot 36 \cdot 49 = 705600$$

- Capital Pi is also called ”Product Symbol”.

# Mathematical notations (Capital Pi)

- The most common example: Factorial
- $n! = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot (n-1) \cdot n$
- Note: $0! = 1$ by definition

- For any positive integer $n$,

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

- (Task 1.2) Input $n$ ($0 \leq n \leq 12$), Output $n!$

# Mathematical notations (Capital Pi)

- (Task 1.2) Input $n$ ($0 \leq n \leq 12$), Output $n!$
- Similar to the code of summation on previous slides.

```
int product = 1;
for (int i = 1; i <= n; i++)
    product *= i;
return product;
```

Initialized to 1 **(not 0)**

# Mathematical notations (Other notations)

- There another example involving Sigma
- The sum of positive divisor function $\sigma_x(n)$

$$\sigma_x(n) = \sum_{d \mid n} d^x$$

- Notation $d \mid n$ means $d$ divides $n$ (alternatively, $n$ is divisible by $d$).
- When $x = 0$, we have $\sigma_0(n)$ which is the number of divisor function
- When $x = 1$, we have $\sigma_1(n)$ which is the sum of divisor function

- (Task 1.3) Input $n$ $(1 \leq n \leq 10^6)$, output $\sigma_1(n)$. (Challenge: $1 \leq n \leq 10^{12}$)

# Greatest Common Divisor (G.C.D.) (Introduction)

- Alternative name: Highest Common Factor (H.C.F.)

- Definition of common divisor:
- If $d$ divides both positive integers $a$ and $b$ (in symbol, $d|a \wedge d|b$), then $d$ is a common divisor of $a$ and $b$.

# Greatest Common Divisor (G.C.D.) (Definition)

- Definition of greatest common divisor:
- 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.

# Greatest Common Divisor (G.C.D.) (Property)

- Property:
- Let $a, b$ be positive integers and $d = gcd(a, b)$
- For any integer $k$,
  - If $k$ is a common divisor of $a$ and $b$,
  - Then $k$ divides $d$ (or $k \mid d$).

- We will prove this later at slide 37.

# Greatest Common Divisor (G.C.D.) (Problem)

Task 2.1

- Input two positive integers $a$ and $b$, output $gcd(a, b)$.

- Constraints:
  - Subtask 1: $a, b \leq 10^6$
  - Subtask 2: $a, b \leq 10^{18}$

# Greatest Common Divisor (G.C.D.) (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))
- Can only pass subtask 1 ☹

# Euclidean Algorithm (Introduction)

- 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 $0 \leq r < b$ (ie. r = a % b)
  - Then $gcd(a, b) = gcd(b, r)$

- Proof:
- Let $d = gcd(a, b)$ and $e = gcd(b, r)$ and $a = qb + r$ for $0 \leq r < b$
  - $d = gcd(a, b) \Rightarrow d|a$ and $d|b \Rightarrow d|r \Rightarrow d|gcd(b, r) \Rightarrow d|e$
  - $e = gcd(b, r) \Rightarrow e|b$ and $e|r \Rightarrow e|a \Rightarrow e|gcd(a, b) \Rightarrow e|d$
- Since $d|e$ and $e|d$, we have $d = e$, ie. $gcd(a, b) = gcd(b, r)$

# 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 (Example)

- Take $gcd(1239, 4557)$ as an example.

| a | = | q | × | b | + | r |
|---|---|---|---|---|---|---|
| 1239 | = | 0 | × | 4557 | + | 1239 |
| 4557 | = | 3 | × | 1239 | + | 840 |
| 1239 | = | 1 | × | 840 | + | 399 |
| 840 | = | 2 | × | 399 | + | 42 |
| 399 | = | 9 | × | 42 | + | 21 |
| 42 | = | 2 | × | 21 | + | 0 |

- So $gcd(1239, 4557) = 21$

# Euclidean Algorithm (Code #2)

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

Time complexity: ???

# Euclidean Algorithm (Code #2)

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

- For $a > b$, $a \% b < \frac{a}{2}$
  - So we reduce the number by half ever time.
- Time complexity: O($\log n$)
- But what is the worse case?

# Euclidean Algorithm (Worst case)

- For the worse case,
  - Since r = a – qb, the q should be as small as possible
  - q will not be zero, 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

# Greatest Common Divisor (G.C.D.) (> 2 numbers)

- What is $gcd(a_1, a_2, a_3, \dots, a_n)$?

- Just calculate the gcd one by one (order does not matter) :

$$gcd(a_1, a_2, a_3, \dots, a_n)$$
$$= gcd\left(a_1, gcd\left(a_2, \dots gcd\left(a_{n-2}, gcd(a_{n-1}, a_n)\right)\right)\right)$$

# Greatest Common Divisor (Application)

- Simply a fraction $\frac{a}{b}$
  - Let $d = gcd(a, b)$
  - Then $\frac{a}{b} = \frac{a \div d}{b \div d}$

- Solving the equation $ax + by = k$ for integers x and y, where $k$ is divisible by $gcd(a, b)$
  - This is a linear Diophantine equation.

# Least Common Multiple (L.C.M.) (Introduction)

- Definition of Common Multiple:
- If $m$ is divisible by both positive integers $a$ and $b$ (in symbol, $a|m \wedge b|m$), then $m$ is a common multiple of $a$ and $b$.

- 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.

# Least Common Multiple (L.C.M.) (Property 1)

- Property 1:
- Let $a, b$ be positive integers and $m = lcm(a, b)$.
- For any integer $k$,
  - If $k$ is a common multiple of $a$ and $b$,
  - Then $m$ divides $k$ (or $m \mid k$).

- Proof: Let $k = qm + r$ for $0 \leq r < m$
- Since $a|k$ and $b|k$, also $a|m$ and $b|m$, so $a|r$ and $b|r$
- $0 \leq r < m = lcm(a, b) \Rightarrow r = 0 \Rightarrow m|k$

# Least Common Multiple (L.C.M.) (Property 2)

- Property 2:

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

- It is not easy to prove, but the idea is simple:
  - How do you find the LCM of two numbers in primary school without listing out the multiples
  - Keyword: Fundamental theorem of Arithmetics

# Least Common Multiple (L.C.M.) (> 2 numbers)

- What is $lcm(a_1, a_2, a_3, \dots, a_n)$?

- Just calculate the lcm one by one (order does not matter) :

$$lcm(a_1, a_2, a_3, \dots, a_n)$$
$$= lcm\left(a_1, lcm\left(a_2, \dots lcm\left(a_{n-2}, lcm(a_{n-1}, a_n)\right)\right)\right)$$

# Extended Euclidean Algorithm (Introduction)

- Extension to Euclidean Algorithm:
- Find the integral solution of $x$ and $y$ such that $ax + by = gcd(a, b)$

- It is guaranteed that a solution exists from Bézout's Lemma

# Extended Euclidean Algorithm (Example)

- In Slide 23, we have computed the value of $gcd(1239,4557) = 21$
- Find integral solution of $1239x + 4557y = 21$.

# Extended Euclidean Algorithm (Example)

- Find integral solution of $1239x + 4557y = 21$.

| $a_i = a_{i-2} \% a_{i-1}$ | $x_i$ | $y_i$ |
|---|---|---|
| 1239 | 1 | 0 |
| 4557 | 0 | 1 |
| 1239 = 1239 – 4557 × 0 | 1 – 0 × 0 = 1 | 0 – 1 × 0 = 0 |
| 840 = 4557 – 1239 × 3 | 0 – 1 × 3 = -3 | 1 – 0 × 3 = 1 |
| 399 = 1239 – 840 × 1 | 1 – (-3) × 1 = 4 | 0 – 1 × 1 = -1 |
| 42 = 840 – 399 × 2 | -3 – 4 × 2 = -11 | 1 – (-1) × 2 = 3 |
| 21 = 399 – 42 × 9 | 4 – (-11) × 9 = 103 | -1 – 3 × 9 = -28 |
| 0 = 42 – 21 × 2 | -11 – 103 × 2 = -217 | 3 – (-28) × 2 = 59 |

- General solution: $x = 103 - 217k$ and $y = -28 + 59k$ (for any integers $k$)

# Extended Euclidean Algorithm (Example)

- Find integral solution of $1239x + 4557y = 21$.
- General solution: $x = 103 - 217k$ and $y = -28 + 59k$ (for any integers $k$)

- If we are going to find the integral solution of $1239x + 4557y = 42$ (doubled), then 103 → 206; -28 → -56.
- General Solution: $x = 206 + 59k$ and $y = -56 - 22k$ (for any integers $k$)

# Greatest Common Divisor (G.C.D.) (proof of slide 17)

Property:

- Let $a, b$ be positive integers and $d = gcd(a, b)$
- For any integer $k$,
  - If $k$ is a common divisor of $a$ and $b$,
  - Then $k$ divides $d$ (or $k \mid d$).

- Proof:
- We have $ax + by = d$. Since $k \mid a$ and $k \mid b$, we have $k \mid ax + by$. Hence $k \mid d$.

# 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÷5 is 3

- Now, we may say 13 ≡ 3 (mod 5)
- This reads, 13 and 3 are congruent modulo 5.

# Modular Arithmetic (Notation)

- Generally, if $a$ and $b$ are congruent modulo $m$ where a where $a$, $b$ are integers, $m$ is a positive integer
  - we will write $a \equiv b \ (mod\ m)$

- But what is $a$ and $b$ are congruent modulo $m$ really means???
- 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 (Other meaning)

- Instead of saying $a$ and $b$ have the same reminder when divided by $m$
- We are more often to say that
  - $a - b$ is divisible by $m$. (or $m \mid (a - b)$)
  - $a = km + b$ for some integer $k$

# Modular Arithmetic (Addition / Subtraction)

- If $a \equiv b \pmod{m}$, for any integers $x$, we have $a + x \equiv b + x \pmod{m}$

Proof:

- Since $a \equiv b \pmod{m}$
- $a = km + b$ for some integer $k$.
- Then $a + x = km + b + x$ .
- Done.

# Modular Arithmetic (Addition / Subtraction)

- If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ , we have $a + c \equiv b + d \pmod{m}$

Proof:

- Since $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$
- $a = km + b$ for some integer $k$ and $c = lm + d$ for some integer $l$
- Then $a + c = km + b + lm + d$ .
- So $a + c = (k + l)m + b + d$
- Since $k + l$ is an integer, so done.

# Modular Arithmetic (Addition / Subtraction)

- Similarly for subtraction
- If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$ , we have $a - c \equiv b - d \pmod{m}$

Proof:

- Since $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$
- $a = km + b$ for some integer $k$ and $c = lm + d$ for some integer $l$
- Then $a + c = km + b - (lm + d)$ .
- So $a + c = (k - l)m + b - d$
- Since $k - l$ is an integer, so done.

# Modular Arithmetic (Multiplication)

- If $a \equiv b \pmod{m}$, for any integers $x$, we have $ax \equiv bx \pmod{m}$

Proof:

- Since $a \equiv b \pmod{m}$
- $a = km + b$ for some integer $k$
- Then $ax = (km + b)x = (kx)m + bx$.
- Since $kx$ is an integer, so done.

- In fact, we have $ax \equiv bx \pmod{mx}$ as well. (Here we need $x > 0$)

# Modular Arithmetic (Division)

- For any integers $c$, If $ac \equiv bc \pmod{m}$, it is not necessary that $a \equiv b \pmod{m}$. WHY? Any counter-example?

- $6 \equiv 2 \pmod{4}$ but $3 \not\equiv 1 \pmod{4}$

- When does it hold?

# Modular Arithmetic (Division)

- Suppose $ac \equiv bc \pmod{m}$
- We have $ac = km + bc$ for some integer $k$
- $a = \frac{k}{c}m + b$

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

# Modular Arithmetic (Division)

- Suppose $ac \equiv bc \pmod{mc}$
- We have $ac = kmc + bc$ for some integer $k$
- $a = k\left(\frac{mc}{c}\right) + b = km + b$

- So if $ac \equiv bc \pmod{mc}$, then $a \equiv b \pmod{m}$.

# Modular Arithmetic (Division)

- Suppose $ac \equiv bc \pmod{m}$
- We have $ac = km + bc$ for some integer $k$
- $\frac{ac}{gcd(c,m)} = k\left(\frac{m}{gcd(c,m)}\right) + \frac{bc}{gcd(c,m)}$
- So we have $\frac{ac}{gcd(c,m)} \equiv \frac{bc}{gcd(c,m)} \pmod{\frac{m}{gcd(c,m)}}$
- Since $\frac{c}{gcd(c,m)}$ and $\frac{m}{gcd(c,m)}$ must be co-prime
- So we have $a \equiv b \pmod{\frac{m}{gcd(c,m)}}$

# Modular Inverse (Introduction)

- What is an 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 $ab \equiv 1 \pmod{m}$.
- We may call $b \equiv a^{-1} \pmod{m}$.
  ($b$ is **the** modular inverse of $a$ modulo $m$)
- We can do division by using modular inverse.
- $ax \equiv c \pmod{m} \rightarrow a^{-1}ax \equiv x \equiv a^{-1}\,c \pmod{m}$

# Modular Inverse (Existence)

- For all integers $a$ and positive integer $m$, is modular inverse of $a$ modulo $m$ always exists?
- NO.
  - If $a$ = 6, $m$ = 4, for any integers $b$, $ab$ is always even
  - i.e. $ab \equiv 0$ or 2 (mod 4) → $ab \not\equiv 1$ (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

- Extended Euclidean Algorithm: We can always find at least one integral solution for $x$, $y$ such that $ax + my = gcd(a, m)$.

- If $gcd(a, m) = 1$, then $ax + my = 1 \Rightarrow ax = (-y)m + 1$
$\Rightarrow ax \equiv 1 \ (mod\ m)$
- Therefore, $a^{-1} \equiv x \ (mod\ m)$

# Modular Arithmetic (Caution!)

- Take care of the sign of the dividend when writing programs.

- If the dividend is negative, the remainder becomes non-positive!
- $-5 \equiv 2 \pmod 7$, but $-5 \bmod 7$ (or $-5 \,\%\, 7$) gives results $-5$.

- Hence, when you may need to write this:

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

# Finding Primes (Introduction)

- Prime number $p$: A positive Number which contains exactly 2 positive divisors. (Note that 1 is not a prime)

- Hence, if $p = ab$, where $a$ and $b$ are positive integers and $a \leq b$, we have: $a = 1$ and $b = p$.
- Property: If $p \mid ab$, then we have $p \mid a$ or $p \mid b$. (Euclid's Lemma)

- Prime numbers: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, …

# Finding Primes (Wilson's Theorem)

- $(p-1)! \equiv -1 \pmod{p}$ if and only if $p$ is a prime.
- i.e. $(p-1)! + 1$ is divisible by $p$ if and only if $p$ is a prime.
- This is only an interesting fact for prime, but this cannot help us to find prime very quickly.

- If you are interested in this and want to see the proof, you can always google or wiki it.

# Finding Primes (Checking)

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 $p$.

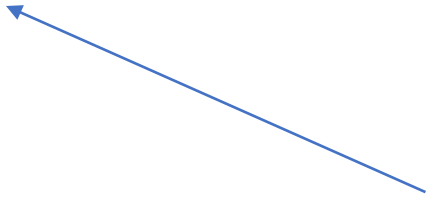
# Checking Primes (Code #1)

```
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 suggestion?

# Checking Primes (Code #2)

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



Never forget this =

- 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 $1 < \frac{p}{k} < \sqrt{p}$

# Find Primes (Prime List)

Task 5.2

- Generate a prime number list in the range of [1, $10^6$]

- You may want to apply your method in the last slide. (Code #2)

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

- Time complexity: O($n\sqrt{n}$) not fast enough ☹

# Find Primes (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 `Comp[i]` which indicates whether i is a known composite number

# Find Primes (Sieve of Eratosthenes)

| Prime List (List all number first!) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

# Find Primes (Sieve of Eratosthenes)

| Prime List (Eliminating 1) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

# Find Primes (Sieve of Eratosthenes)

| Prime List (Eliminating multiples of 2 except 2) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

# Find Primes (Sieve of Eratosthenes)

| Prime List (Eliminating multiples of 3 except 3) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | | 5 | | 7 | | 9 | |
| 11 | | 13 | | 15 | | 17 | | 19 | |
| 21 | | 23 | | 25 | | 27 | | 29 | |
| 31 | | 33 | | 35 | | 37 | | 39 | |
| 41 | | 43 | | 45 | | 47 | | 49 | |
| 51 | | 53 | | 55 | | 57 | | 59 | |
| 61 | | 63 | | 65 | | 67 | | 69 | |
| 71 | | 73 | | 75 | | 77 | | 79 | |
| 81 | | 83 | | 85 | | 87 | | 89 | |
| 91 | | 93 | | 95 | | 97 | | 99 | |

# Find Primes (Sieve of Eratosthenes)

| Prime List (Eliminating multiples of 5 except 5) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | | 5 | | 7 | | | |
| 11 | | 13 | | | | 17 | | 19 | |
| | | 23 | | 25 | | | | 29 | |
| 31 | | | | 35 | | 37 | | | |
| 41 | | 43 | | | | 47 | | 49 | |
| | | 53 | | 55 | | | | 59 | |
| 61 | | | | 65 | | 67 | | | |
| 71 | | 73 | | | | 77 | | 79 | |
| | | 83 | | 85 | | | | 89 | |
| 91 | | | | 95 | | 97 | | | |

# Find Primes (Sieve of Eratosthenes)

| Prime List (Eliminating multiples of 7 except 7) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | | 5 | | 7 | | | |
| 11 | | 13 | | | | 17 | | 19 | |
| | | 23 | | | | | | 29 | |
| 31 | | | | | | 37 | | | |
| 41 | | 43 | | | | 47 | | 49 | |
| | | 53 | | | | | | 59 | |
| 61 | | | | | | 67 | | | |
| 71 | | 73 | | | | 77 | | 79 | |
| | | 83 | | | | | | 89 | |
| 91 | | | | | | 97 | | | |

# Find Primes (Sieve of Eratosthenes)

| Prime List (Done since 11×11 > 100) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | | 5 | | 7 | | | |
| 11 | | 13 | | | | 17 | | 19 | |
| | | 23 | | | | | | 29 | |
| 31 | | | | | | 37 | | | |
| 41 | | 43 | | | | 47 | | | |
| | | 53 | | | | | | 59 | |
| 61 | | | | | | 67 | | | |
| 71 | | 73 | | | | | | 79 | |
| | | 83 | | | | | | 89 | |
| | | | | | | 97 | | | |

# Generating Prime List (Code #3)

```
// Initialize integer array comp[] = {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;
    }
}
```

# Find Primes (Prime Factorization)

- 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

# Factorial Factors (Introduction)

Task 6.1

- In this section, we are interested to find the highest power of $p$ such that it divides $n!$, where $p$ is a prime.

- Input a prime $p$ and a positive integer $n$
- Output the highest power $k$ such that $p^k \mid n!$

- Do you still remember how to compute $n!$ ?

# Factorial Factors (Code #1)

```
int num = factorial(n); // factorial(n) = n!
int k = 0;
while (num % p == 0){
     k++;
     num = num / p;
}
return k;
```

Subtask 1: $1 \le n \le 10$ Accepted ☺

Subtask 2: $1 \le n \le 10^6$ Wrong Answer ☹

Subtask 3: $1 \le n \le 10^9$ Wrong Answer ☹

# Factorial Factors (Observation)

- By definition, $n! = 1 \times 2 \times 3 \times 4 \times \cdots \times (n-2) \times (n-1) \times n$.

MATHEMATICS PART:

- If $C = A \times B$, where the highest power of $p$ to $A$ and $B$ are $x$ and $y$ respectively, i.e. $p^x \mid A$ and $p^y \mid B$. Then the highest power of $p$ to $C$ is $x + y$.

# Factorial Factors (Code #2)

```
int k = 0;
for (int i = 1; i <= n; i++){
    int num = i;
    while (num % p == 0){
        k++;
        num = num / p;
    }
}
return k;
```

Time complexity: O($n \log n$)
Subtask 1: $1 \le n \le 10$ Accepted ☺
Subtask 2: $1 \le n \le 10^6$ Accepted ☺
Subtask 3: $1 \le n \le 10^9$ TLE ☹

# Factorial Factors (Example)

| **Example: $n$ = 20 and $k$ = 2** | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Divisibility | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 2 | | ✓ | | ✓ | | ✓ | | ✓ | | ✓ | | ✓ | | ✓ | | ✓ | | ✓ | | ✓ |
| 4 | | | | ✓ | | | | ✓ | | | | ✓ | | | | ✓ | | | | ✓ |
| 8 | | | | | | | | ✓ | | | | | | | | ✓ | | | | |
| 16 | | | | | | | | | | | | | | | | ✓ | | | | |
| Highest power | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 3 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 4 | 0 | 1 | 0 | 2 |
| Sum of Highest Powers = 18. | | | | | | | | | | | | | | | | | | | | |

# Factorial Factors (Observation)

- Is there another way to come up the result 18?
- (Or equivalently, any other way to count the number of ticks?)

- Instead of counting the number of ticks per column
- why don’t we count it per row, which is meaningful

- Row $r$ represents: the number of integers between 1 and 20 in which they are divisible by $2^r$

# Factorial Factors (Conclusion)

- Row $r$ represents: the number of integers between 1 and $n$ in which they are divisible by $p^r$

- How many? It's easy! ☺
- Answer: $\left\lfloor \frac{n}{p^r} \right\rfloor$

# Factorial Factors (Code #3)

```
int k = 0;
while (n){
    n /= p;
    k += n;
}
return k;
```

Time complexity: O($\log n$)
Subtask 1: $1 \le n \le 10$ Accepted ☺
Subtask 2: $1 \le n \le 10^6$ Accepted ☺
Subtask 3: $1 \le n \le 10^9$ Accepted ☺

$$Answer = \left\lfloor \frac{n}{p} \right\rfloor + \left\lfloor \frac{n}{p^2} \right\rfloor + \left\lfloor \frac{n}{p^3} \right\rfloor + \cdots = \sum_{i=1}^{\infty} \left\lfloor \frac{n}{p^i} \right\rfloor$$

# Factorial Factors (Upgraded Exercise)

Task 6.2

- Find the highest power of $m$ such that it divides $n!$, where $m$ is an integer > 1
- Input an integer $m$ (> 1) and a positive integer $n$.
- Output the highest power $m$ such that $m^k \mid n!$
- Hint: Prime Factorization of $m$.
- Application:
- Finding the trailing zeros of $n!$, modular arithmetic with modulo power of $m$.

# High Precision Arithmetic (H.P.A.) (Introduction)

- 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 asking to read any integers which consists of ≥ 20 digits…

# High Precision Arithmetic (H.P.A.) (Introduction)

- Use 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! ☹

# High Precision Arithmetic (H.P.A.) (Selection of Base)

- Numerical System in Base $n$. ($n$ 進制)
- How should we select the value of $n$?

- Power of 2: Save memory
- Power of 10: Easy Input / Output ☺
  - 1 000 or 10 000: for 16-bit integer array
  - $10^9$: for 32-bit integer array

# High Precision Arithmetic (H.P.A.) (More!)

- How to store the following:
  - Negative numbers?
  - Fraction?
  - Floating-point numbers?

# The end