Math For CP - Beginner

- Viraj Chandra

Goal

To understand:

- Role Of Modulo
- Basic Modular Arithmetic
- GCD & LCM
- Common properties of GCD & LCM
- OEIS - Online Encyclopedia of Integer Sequences

Role Of Modulo

Example: For M = 3,

[0 to M-1]

- 10 % 3 = 1
- 11 % 3 = 2
- 12 % 3 = 0 (It resets after 3, creating a repeating cycle).

Real-Life Analogy:

- Think of clock arithmetic:
  - On a 12-hour clock, 13 % 12 = 1 = 1 % 12 = 1, meaning **13 o'clock is the same as 1 o'clock.**

Role Of Modulo

Handling Overflow:

Modulo helps reset values when they exceed a limit, avoiding overflow. **Example:** In a timer with 60 seconds, **75 % 60 = 15**

Checking Even or Odd:

Modulo 2 identifies even and odd numbers:

- N % 2 = 0 : The number is **even**.
- N % 2 = 1 : The number is **odd**.

Modular Arithmetic

Modular Arithmetic is arithmetic operations **involving taking the modulo with numbers. Symbol - '%'**

Modular Arithmetic involves the **following operations:**

- ✓ Modular Addition
- ✓ Modular Subtraction
- ✓ Modular Multiplication
- Modular Division

NOTE: Modular Division is not covered in this level.

Modular Addition

Modular Addition has the following formula:

$$\underline{(A + B) \% M} = \underline{(A \% M + B \% M)} \% M$$

Let us understand this formula with an example of candies. (Yum)

Modular Addition

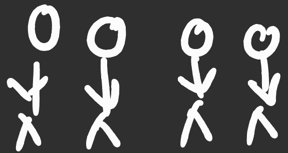
Imagine there are 12 candies split between two rooms -
7 in one, 5 in the other.

You want to share them equally among **4 friends.**
Each gets 3 candies, but you have no leftovers.

This is like modulo addition: add the candies (7 + 5 = 12), and find the remainder left with modulo 4.
=> 12 % 4 = 0 (oops, no candies left) ✓

$\infty - 7$

$\infty - 5$



$(7+5) = 12 \% 4 = 0$

$7 \% 4 = 3 \rightarrow [0 \text{ to } 3]$

$5 \% 4 = 1$

$$(7 \% 4 + 5 \% 4) \% 4$$

$$(3+1) \% 4 = 4 \% 4 = 0$$

$$\underline{\boxed{10^9-2}\ \%4 \quad \boxed{10^9-1}\ \%4}$$

$$\boxed{4}$$

$$\begin{tikzcd}
 & 10^9 \\
(2\times10^9-3) \arrow[ur, bend right] &
\end{tikzcd}$$

Modular Addition

Now, imagine that the candies count in each room were larger, larger than the integer limit in C++, **can you add them and divide them now? - (Overflow)**

Let’s do a smarter move. **What if we apply the distribution of candies for each room before we add up candies?**

Modular Addition

Let us distribute **7 candies** first, then the **5 candies** next.

=> 7 % 4 = 3 (candies left from room 1)
=> 5 % 4 = 1 (candies left from room 2)

Observe, can we add them up and distribute again? After all, candies are all same.

=> (3 + 1) % 4 = 0 (oops, no candies left again)

Modular Addition

This means, to prevent overflow, modular addition was useful.

=> (7 + 5) % 4 = (7 % 4 + 5 % 4) % 4 = 0

OR

=> (A + B) % M = (A % M + B % M) % M

This proves our initial formula, and also explains why we did a modulo outside the bracket again. **Fun, right?**

Modular Subtraction

Modular Subtraction has the following formula:

$$(A - B) \% M = ((A \% M - B \% M) + M) \% M$$

NOTE: Careful with the **extra addition of M outside.**

Let us understand this formula with an example of debt and money.

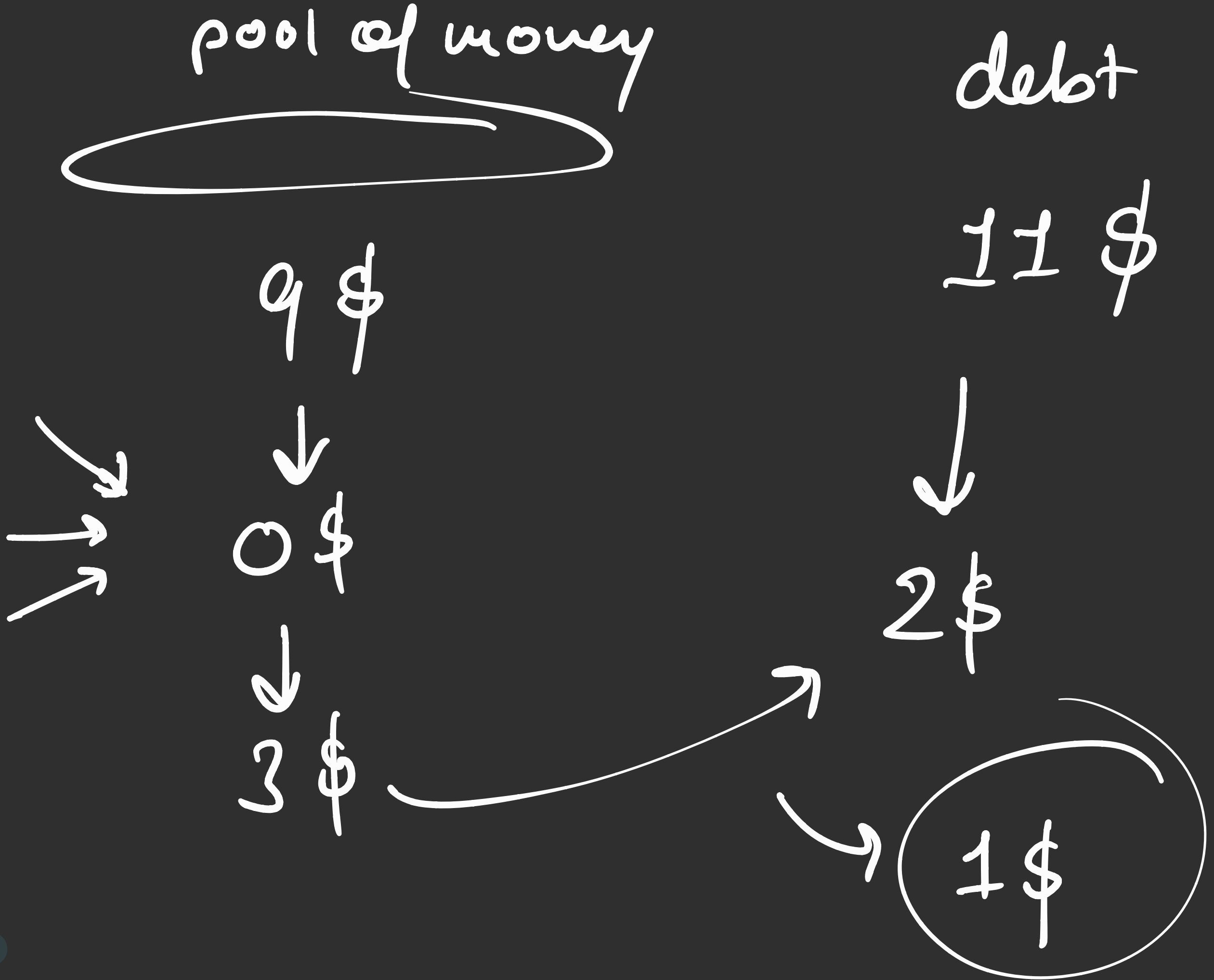
Modular Subtraction

Imagine **3 friends** are trying to pay a **debt $11,** but you only have **$9 in a pool of money.**

After the friends pay all they can, **they would owe $2 more.** So, in order to get more money, each of the friends contribute **$1 more to the pool.**

This can be written as the following,

=> ((9 - 11) % 3 + 3) % 3 = $1 left , where **+ 3** symbolizes the additional pool of money in order to keep the **pool positive in nature.**


pool of money
9$
0$
3$
debt
11$
2$
1$

$$(9 \% 3 - 11 \% 3) \% 3$$

$$(0 - 2) \% 3$$

$$(-2 \; \underline{+ 3}) \% 3$$

$$1 \% 1$$

$$= \underline{\underline{1}}$$

Modular Subtraction

Hence, it can be simply stated, that in regular subtraction,

(A - B) = (9 - 11) = -2

But, in modular arithmetic, **we don't take negative results.** Hence we **add M back** to wrap the result into the range **0 to M - 1**

((A - B) % M + M) % M = ((9 - 11) % 3 + 3) % 3 = 1

Modular Multiplication

Modular Multiplication has the following formula:

$$(A * B) \% M = ((A \% M) * (B \% M)) \% M$$

NOTE: Since the operator precedence level of “%” , “*” , “/” are all same, they are executed from left to right in a single line. Hence, the **colored brackets shown** are a must for correct execution of Modular Multiplication formula.

$$\begin{tikzcd}
{} \arrow[r] & \left(\overbrace{A \% M}\right) * \left(\overbrace{B \underline{\% M}}\right) \\
(A \% M) & \\
R \arrow[uur, bend right] &
\end{tikzcd}$$

Modular Multiplication

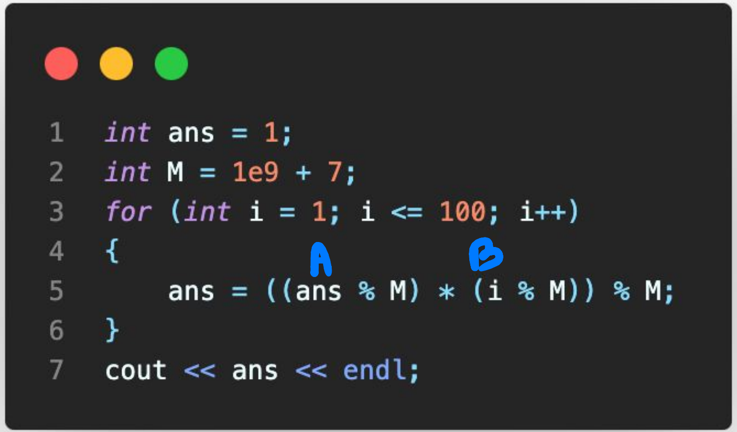
$100! > \underline{10^{18}}$

How can we tackle a question like this now?

M =

“Find factorial of 100, and output your result modulo $\underline{1e9 + 7}$”

X



A B

```
int ans = 1;
int M = 1e9 + 7;
for (int i = 1; i <= 100; i++)
{
    ans = ((ans % M) * (i % M)) % M;
}
cout << ans << endl;
```

Taking modulo again over **each iteration** keeps us within the required overflow limit.

Q $\rightarrow$

Output ans modulo M

Arithmetic $\longrightarrow$ modular

Greatest Common Divisor (GCD)

Greatest Common Divisor of two or more integers, which are not all zero, is the **largest positive integer** that **divides each of the integers.**

Example: The GCD of 8 and 12 is 4, that is, GCD(8, 12) = 4.

To calculate GCD efficiently, we use the Euclidean Algorithm in CP.

Greatest Common Divisor (GCD)

The algorithm is based on the below facts.

- If we subtract a smaller number from a larger one (we reduce a larger number), GCD doesn't change. **So if we keep subtracting repeatedly the larger of two, we end up with GCD.**
  GCD (A , B) = GCD (A - B , B) , assuming A >= B
- When difference reaches 0 , **the solution is B.**

$(12, 8)$

$$(12, 8) \rightarrow \overset{B}{(4}, \overset{A}{8)} \rightarrow (4, 4)$$

$$\rightarrow (0, \overset{2}{4})^{\text{GCD}} \times$$

$$\rightarrow \boxed{4}$$

Greatest Common Divisor (GCD)

Let us prove the Euclidean Algorithm.

If GCD (A , B) = G, this means A % G = 0 and B % G = 0

Let, A = a*G and B = b*G, where ‘a’ and ‘b’ are the factors not common.

(A - B) = (a - b)*G which is also divisible by G.

Hence **GCD (A - B , B) = G is true.**

Lowest Common Multiple (LCM)

Lowest Common Multiple of two or more integers is the **smallest positive integer** that **is divisible by each of the integers.**

LCM of two numbers A and B can be expressed with the following equation. **(This works only for two numbers)**

LCM (A , B) = A x B / GCD (A , B)

Why is this true? Let us see with a small proof.

Proof

Any number N can be represented as the product of its prime factors.
To generalise this, we can say the following,

$$N = p_1^{a1} \times p_2^{a2} \times p_3^{a3} \times \ldots \; p_n^{an}$$

where, $p_1, p_2, p_3 \ldots\; p_n$ are **prime numbers**
and, $a_1, a_2, a_3 \ldots\; a_n$ are **powers of those prime numbers**

Proof

If we take two numbers A and B,

$A = 2^3 \times 5^2 \times 7^1 \times 11^2$

$B = 2^5 \times 5^3 \times 7^2 \times 11^1$

then, GCD of A and B can be written as

GCD (A , B) = 2 ^ (min(3,5)) x 5 ^ (min(2,3)) ....

$= 2^3 \times 5^2 \times 7^1 \times 11^1$

$$\begin{tikzcd}
8\ ,\ 12 \arrow[rr] & & \boxed{4} \arrow[dd] \\
\underline{2}^3 \arrow[r, leftrightarrow, bend right] & \underline{2}^2 \times 3^1 & \\
2^2 \arrow[rr] & & \boxed{2^2}
\end{tikzcd}$$

Proof

and, LCM of A and B can be written as

LCM (A , B) = 2 ^ (max(3,5)) x 5 ^ (max(2,3)) ....

$= 2^5 \times 5^3 \times 7^2 \times 11^2$

This comes from a simple fact that GCD is trying to take the common divisor hence **minimum of the two powers,** and LCM is trying to take the common multiple hence **maximum of two powers.**

Proof

Now, if we multiply GCD and LCM together, we get back the product of A and B, which is what we wanted.

GCD (A , B) x LCM (A , B) = $2^{(3+5)}$ x $5^{(2+3)}$ …. = A x B

$\underline{\text{- - gcd}(a,b)} \rightarrow$

$a*b$

Hence proved.

Common Properties of GCD & LCM

- GCD (A , B) can be represented as product of **min(px^{ax}, px^{bx})** for each prime factor.
- LCM (A , B) can be represented as product of **max(px^{ax}, px^{bx})** for each prime factor.
- **GCD (A , B) x LCM (A , B) = A x B**
- **GCD (A , A + 1) = 1**

$$\left(\overset{\downarrow}{a}, \overset{\downarrow}{a+1}\right) \curvearrowleft 1$$

One will be even

Other will be odd

Common Properties of GCD & LCM

- GCD is associative in nature, as in,

  GCD (A , B , C , ...) = GCD (A , GCD (B , GCD (C , ...)))
- Similarly, LCM is associative in nature, as in,

  LCM (A , B , C , ...) = LCM (A , $\underset{\text{LCM}}{\cancel{\text{GCD}}}$ **(B ,** $\underset{\text{LCM}}{\cancel{\text{GCD}}}$ **(C , ...)))**
- **GCD (A , B) <= min (A , B)**

$$R = \frac{a[i] * R}{\gcd(a[i], R)}$$

Online Encyclopedia of Integer Sequences

Link: https://oeis.org/

OEIS can be used to find the **formula of an integer sequence** with just the first few values (which could be computed using brute-force or manually by hand).

Helpful in contests.

Important Links

- Modular Arithmetic Basics: https://bit.ly/3cl0Vdj