%, GCD, LCM

Math For CP - Intermediate

- Viraj Chandra

Goal

To understand:

- Importance of Primes
  - Basic Hashing
  - Irregular Nature
- Primality Testing
  - Brute Force Method
  - SQRT Method
- Prime Factorisation
  - Trial Division Method

Importance Of Primes

Have you seen questions which ask to print **answer modulo 1e9+7** ?

- What is so special about 1e9+7 ?
- What is this helping us with ?
- Can we use some other number than this ?

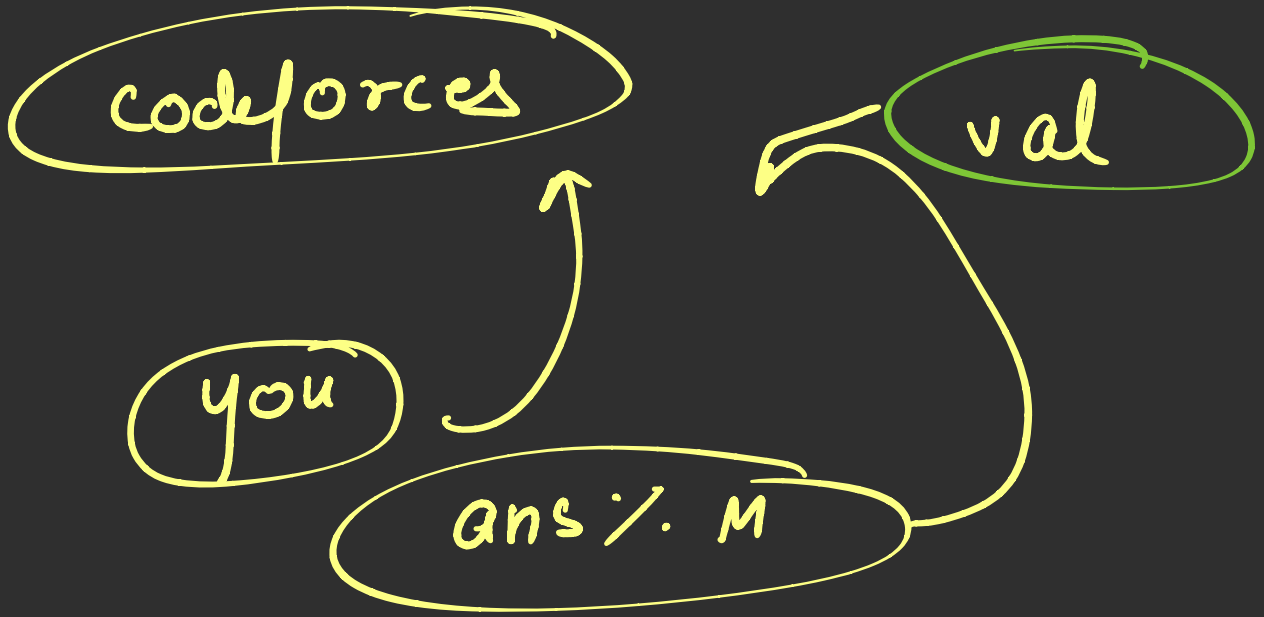
Importance Of Primes

1e9+7 is visibly -

- A very large number
- Prime in nature

Let us understand importance of primes with this.

Hashing - It is a fundamental technique in CP that is used to **efficiently manipulate and process large amounts of data.** Basically, this helps us to **control overflow,** and maintain answers within a **specific range,** defined by modulo M, i.e. **[0 , M - 1].**



Importance Of Primes

Example:

Let's say you calculate your **answer to a problem is 123456,** but the platform asks you to print **answer modulo 103**

Hashing fn

=> 123456 % 103 = 62

This means that **123456 has been mapped to a value of 62** and now, 62 will be treated as the right answer. This phenomenon is called Hashing.
(Link attached at the end of slides)

a

$123456 \% 103 = 62$ h_1

b

$123559 \% 103 = 62$ h_2

codeforces

$a \% M$

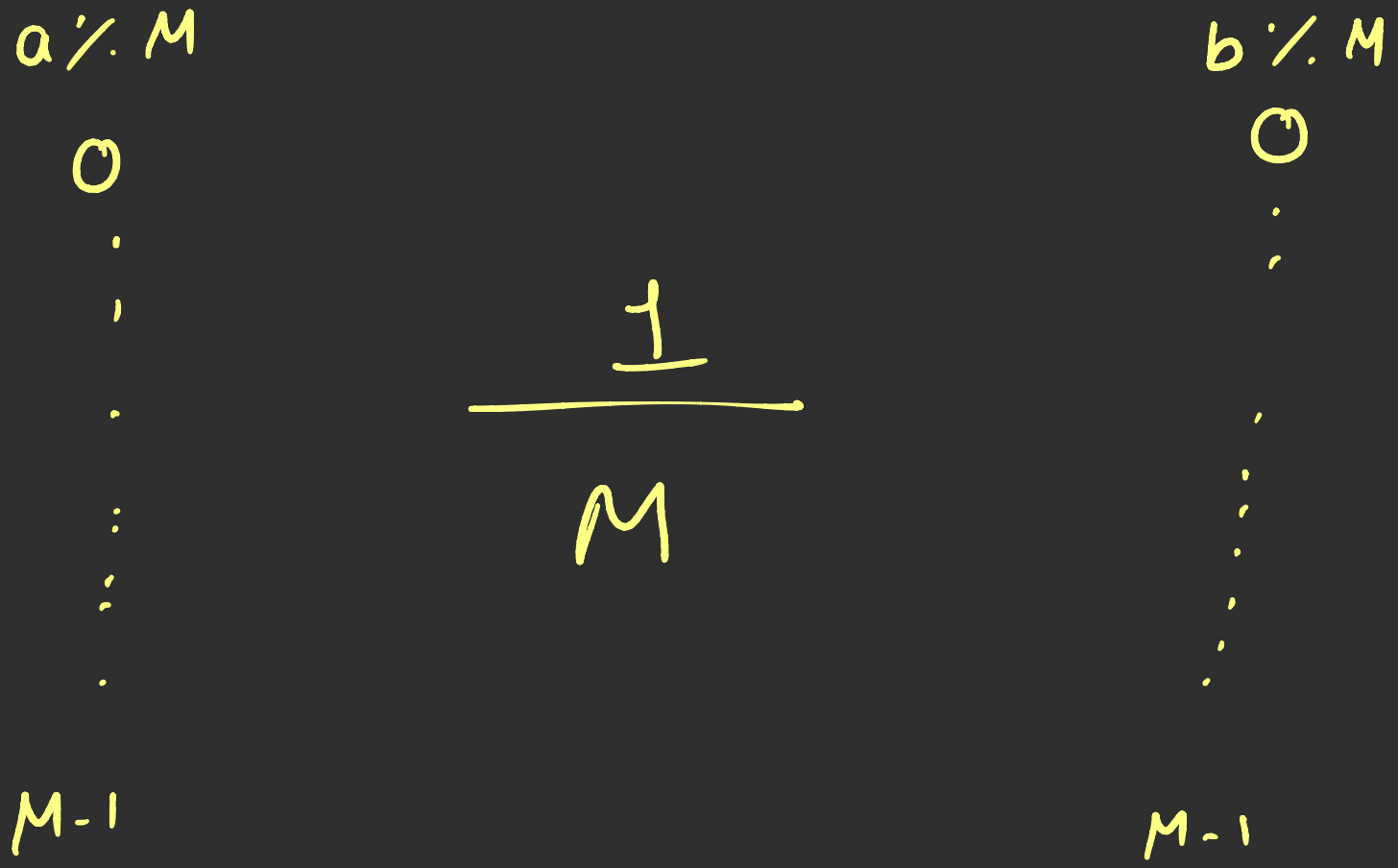
$b \% M$

} what is the probability that their hash values collide?

$a \% M = [0 \text{ to } M-1]$

↓

M numbers



↓ $p = 1/M$ ↑ ⟶ large value → then collision is avoided

↓

prob.. of collision should be as low

$1/1e^{9+7} \approx 10^{-9}$

$a \% M = h_1$

$b \% M = h_2$

$a != b$

$h_1 != h_2$

$a != b$

but, $h_1 == h_2$

collision clash

Importance Of Primes

Example:

Answer maps to a range of [0 , 102], due to modulo 103. **Can we get the same result again?**

=> 123559 % 103 = 62

Interesting. Seems wrong, or to be specific, this seems like a clash of answers. **How do we avoid this?**

Importance Of Primes

If we increase the value of M, our said range of mapping also increases.

Therefore, **M** should be a **large number.**

1e9+7 fits well now**,** since it gives us a huge range of numbers our answer can be mapped to from [0 , 1e9+6].

Importance Of Primes

But, why a prime?

Observe the first 10 primes.

2, 3, 5, 7, 11, 13, 17, 19, 23, 27 … ?

Can you just guess the **11th prime?** Is there some visible pattern. **NO.**

Importance Of Primes

Conclusion?

- Primes are highly irregular in nature. They do not follow any series of rules.
- Lower chances of clash, since answers being **multiples of large prime numbers is a very low possibility. Example -** 1e9+7

M = 1e9+7

a $1e9+7$ $\because M = 0$ h_1

b $2 \times (1e9+7)$ $\because M = 0$ h_2

Importance Of Primes

Now that we understand the true nature of M here, can more numbers fit our case? **YES.**

- 1e9+9 and 1e9+11
- 998244353

Overall, **large prime numbers are the most common choice for M.**

1e9+7

Primality Testing

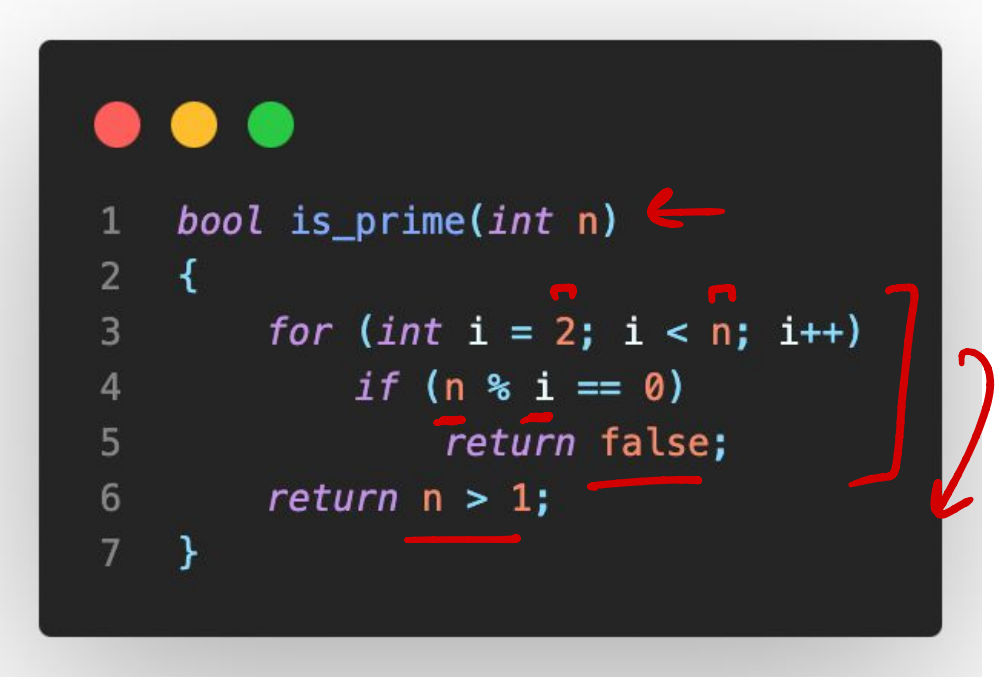
A primality test is an algorithm for determining whether **a number is prime or composite.**

There are many algorithms for primality testing such as:

- Brute Force Method
- SQRT Method
- Using Sieve (Precomputation)

NOTE: Sieve is not covered in this level. (Link attached at the end of slides)

Brute Force Method



```
1  bool is_prime(int n)
2  {
3      for (int i = 2; i < n; i++)
4          if (n % i == 0)
5              return false;
6      return n > 1;
7  }
```

TC = O(N)

SC = O(1)

N = composite

N = A × B → pairs

$A > \sqrt{N} \times B \geq \sqrt{N}$

$A > \sqrt{N}$

$B \geq \sqrt{N}$

} NO

$A \times B > N$

$$N = A \times B$$

$$A = a < \sqrt{N}$$

$$B = \frac{N}{a}$$

$$6 = 2 \times B$$

$$B = \frac{6}{2} = 3$$

→ for(i = 2 $\leq \sqrt{N}$)

{

}

SQRT Method

If a **number N is not a prime,** then it must be possible to factor into two values A and B such that **A*B = N**

If **A > √N and B > √N,** then

A*B > N (impossible)

Therefore, if the number is not a prime, it must have **at least one factor less than or equal to the square root of the number.**

SQRT Method

sqrt(n)

```
1  bool is_prime(int n)
2  {
3      for (int i = 2; i * i <= n; i++)
4          if (n % i == 0)
5              return false;
6      return n > 1;
7  }
```

TC = O(sqrt(n))

SC = O(1)

Quiz

Find all divisors of a number.

Example:

N = 20, Divisors = { 1, 2, 4, 5, 10, 20 }

Hint: Extend **SQRT Method** here.

Quiz

$\bar{3} \times \bar{3}$

perfect sq = 9

```
1  for (int i = 1; i * i <= n; i++)
2  {
3      if (n % i == 0)
4      {
5          if (i != n / i)
6              cout << i << " " << n / i << endl;
7          else
8              cout << i << endl;
9      }
10 }
```

A i = factor

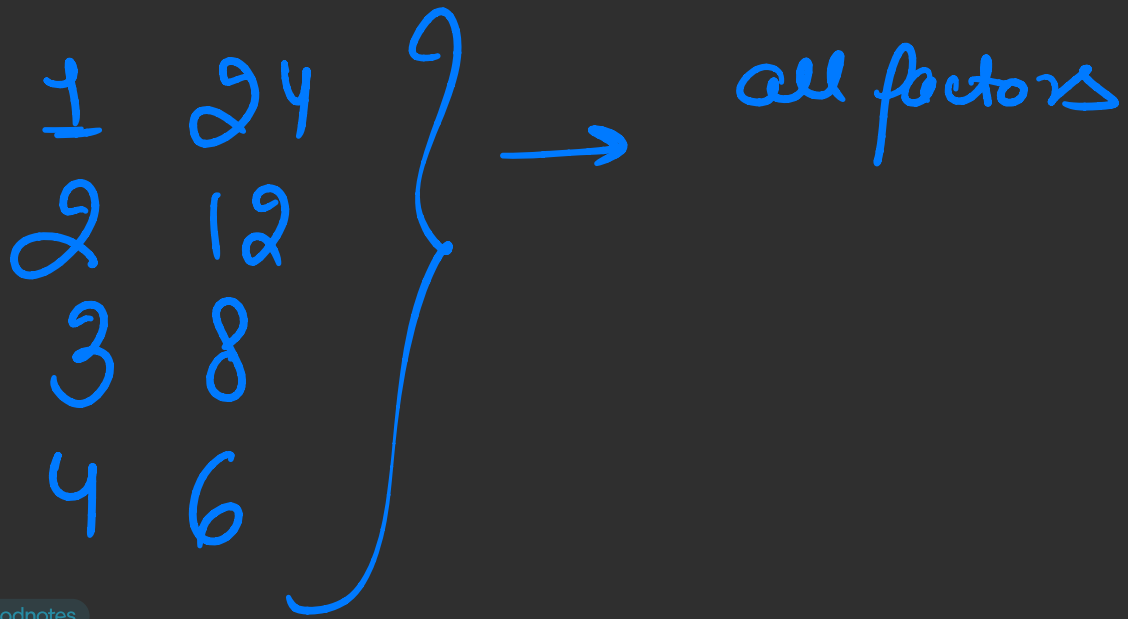
B n/i = factor

TC = O(√N)

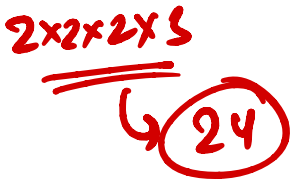
SC = O(1)

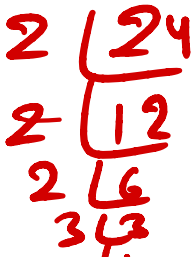
N = 24

1 to 4



Prime Factorisation





Prime Factorization is finding all the prime factors of a given number.

There are multiple ways to factor primes, such as:

- Trial Division Method
- Using Sieve (Precomputation)

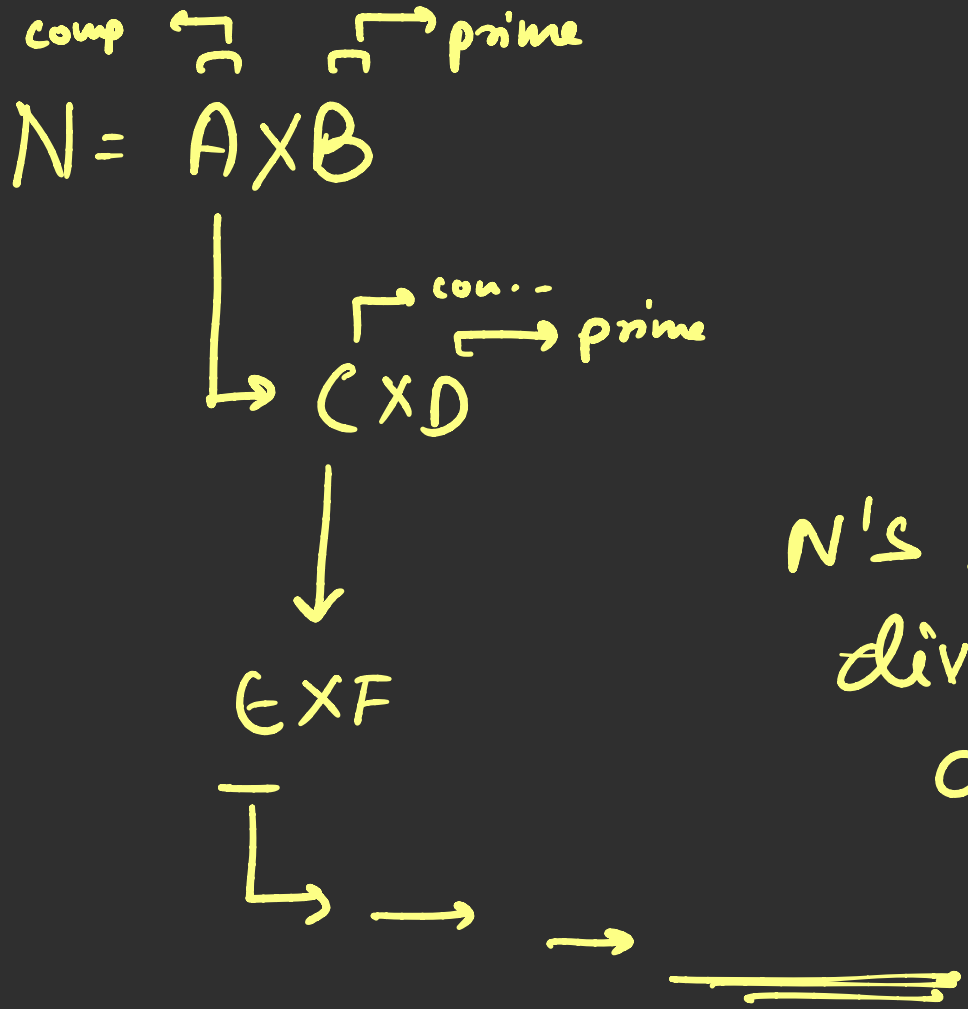
NOTE: Sieve is not covered in this level. (Link attached at the end of slides)

Trial Division Method

Trial division is the most basic method of prime factorization.

We will use the fact that the smallest divisor of any number N is prime, and **it will be less than $\sqrt{N}$.**

This comes from our previous discussion on **SQRT Method.**



Trial Division Method

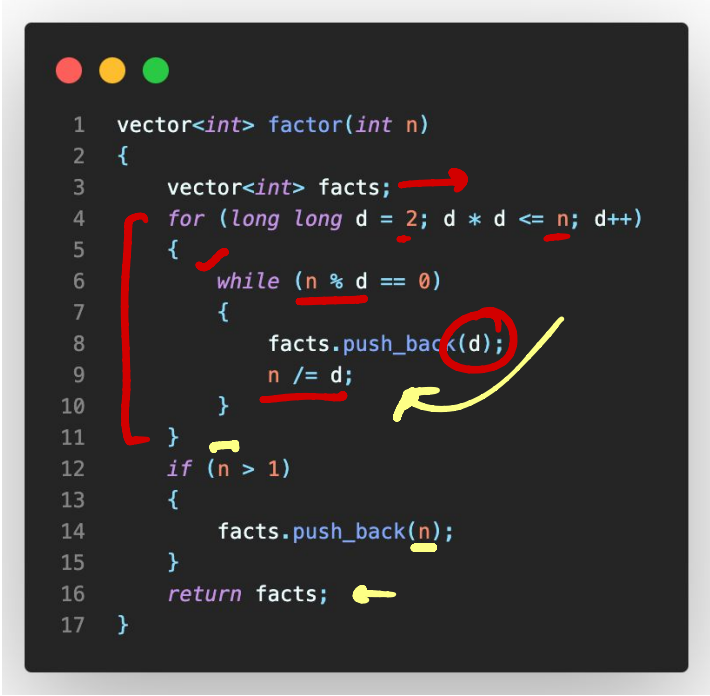
$$N = p_1^{a1} \times p_2^{a2} \times p_3^{a3} \times \ldots p_n^{an}$$

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**

We will iterate through the range [2, $\sqrt{N}$] to find p_1 first, then p_2 and so on.

Trial Division Method



```cpp
vector<int> factor(int n)
{
    vector<int> facts;
    for (long long d = 2; d * d <= n; d++)
    {
        while (n % d == 0)
        {
            facts.push_back(d);
            n /= d;
        }
    }
    if (n > 1)
    {
        facts.push_back(n);
    }
    return facts;
}
```

$N = 24$

$d = 2$

Best TC : $O(\log_2 N)$

Worst TC / TC : $O(\sqrt{N})$

$$N = 2^{5} \times 3^{3} \times 5^{-1} \ldots\ldots$$

$$= 2^{a_1} \times 3^{a_2} \times 5^{a_3} \ldots\ldots$$

$$TC = a_1 + a_2 + a_3 + \ldots\ldots$$

a_1 is relatively larger for most N

$$N \approx 2^{a_1}$$

$$\log_2 N \approx a_1$$

$$N/2 > N/5$$

$n = 52 = 2 \times 2 \times 13$ ← prime

2 to 7

$52/2 \rightarrow 26/2 \rightarrow 13$ Break for →

$n = 24 = 2^3 \times 3^1$

2 to 4

$2 \rightarrow 24/2 \rightarrow 12/2 \rightarrow 6/2 \rightarrow 3$

Important Links

- Computations Modulo P: https://www.youtube.com/watch?v=-OPohCQqi_E
- Primality Test: https://en.wikipedia.org/wiki/Primality_test
- Sieve: https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes