1. Prime Number: Prime numbers are positive integers greater than 1 that have no divisors other than 1 and themselves.

n >1

2 is the smallest prime number

Note: The number 1 is not considered a prime number because prime numbers, by definition, are positive integers greater than 1 that have exactly two distinct positive divisors: 1 and the number itself.

However, the number 1 has only one positive divisor, which is 1 itself. It does not meet the requirement of having exactly two distinct positive divisors, so it is not classified as a prime number.

Que: n is prime number or not?

Ans: let i = 2……..9, do (n%i), if n%i ==0 not a prime number

**Edge case:**

**If(n<=1){**

**Return false;**

**}**

For(let I = 2; i<=9; i++) {

If(n%i == 0){

Return false;

}

}

Return true;

Que: count primes, let say in input u have n =46, so count numbers which are prime from 1 to 46.

Constraint: 0<=n>=50\*10^5

TC = O(n^2)

N^2 = (50\*10^5)^2 => (5\*10^6)^2 => (10^7)^2/(2^2) => (10^14)/4 => 25\*10^12

So, machine will give TLE, cos in it could take 10^12 seconds, but now a days machine srf 1 sec me 10^8 computations karti hai.

So, that’s why we have to optimize this algorithm.

So will use Sieve of eratosthenes

Find Prime number using Sieve method

1. First mark every number as prime number from (2……39)
2. Then check in table, which number is coming, whichever is coming mark them as non-prime

Segmented Seive

The segmented sieve is an optimized version of the traditional sieve of Eratosthenes algorithm used to generate prime numbers. The sieve of Eratosthenes is a method to identify all prime numbers up to a given limit, but it can become memory-intensive for large ranges.

The segmented sieve breaks down the range into smaller segments, reducing the memory requirements. The basic idea is to find the primes in each segment individually and then combine the results to obtain all primes within the given range.

Here's a general outline of the segmented sieve algorithm:

Choose a segment size, let's say 'sqrt(N)' where 'N' is the upper limit of the range you want to generate primes for.

Generate all primes up to the square root of the upper limit using the traditional sieve of Eratosthenes algorithm.

Iterate over the segments of size 'sqrt(N)' within the range, starting from the lower limit. For each segment:

a. Create a boolean array of size 'sqrt(N)' to track the prime status of numbers in the segment.

b. Mark all multiples of primes found in step 2 within the current segment as non-prime.

The remaining unmarked numbers within each segment are prime numbers within that segment.

Repeat step 4 for all segments in the range.

Combine the primes from each segment to obtain all prime numbers within the desired range.

The segmented sieve improves memory efficiency by sieving primes in smaller segments, rather than storing all numbers within the range in memory. This allows for efficient generation of prime numbers even for very large ranges.

The segmented sieve algorithm is a powerful technique for generating prime numbers and is commonly used in practice.

GCD/HCF: ek esa maximum(greatest) number jo in dono hi number ko perfectly divide kar skta hai and remainder 0 la skta hai. That number is called highest common factor or greatest common factor

In JavaScript, 1e9 is a shorthand notation for the number 1 followed by 9 zeros, which represents 1 billion.

The letter 'e' in 1e9 stands for exponentiation, and it indicates that the number should be multiplied by 10 raised to the power of the exponent that follows it. In this case, 1e9 is equivalent to 1 \* 10^9, which results in 1 billion.

Here are a few examples to illustrate the usage of 1e9 in JavaScript: