Sieve of Eratosthenes

Monday, July 15, 2024 11:10 AM

➢ **Whenever you have to find the range of prime numbers always think of sieve of eratosthenes**

1) Optimized Prime no. Program

```
public static boolean isPrime(int n) {
        if (n == 2) {
            return true;
        }
        for (int i = 2; i <= Math.sqrt(n); ++i) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }
```

Time complexity → $O(N \times \sqrt{N})$

Space complexity → $O(1)$

→ But if we have to find the range of prime no. let say 10^6 than this method is very bad.

So we use sieve of eratosthenes

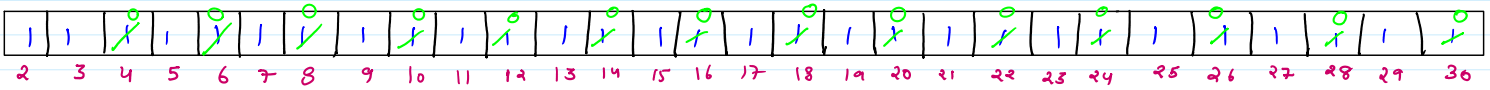
Time complexity → $O(N \times \sqrt{N})$ → $O(1)$ → by sieve of eratosthesis

Space complexity → $O(1)$

Lets say we have to find the prime number less 30

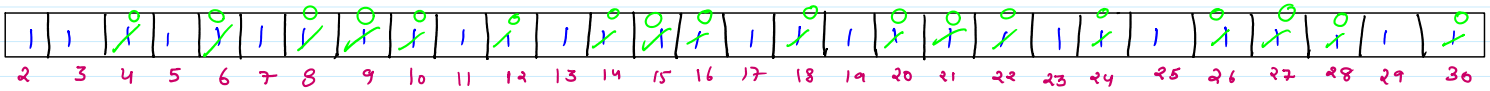
first, declare an array of size (n+1) and mark from 2 to N as 1 (which denotes all are prime no.)

Mark multiples of 2 as 0

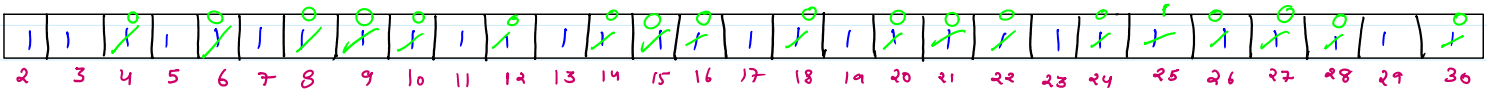


Mark multiples of 3 as 0

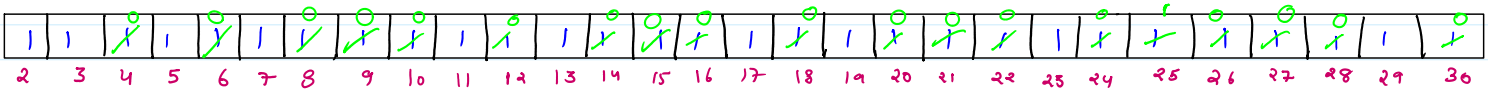
We cannot Mark multiples of 4 as 0 because somebody mark 4 as 0 which means 4 is not a prime no.



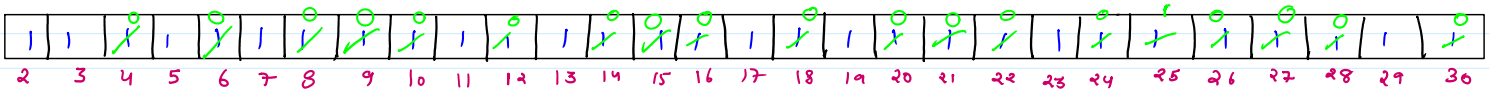
Mark multiples of 5 as 0



Similarly we cannot Mark multiples of 6 as 0
Mark multiple of 7 as 0



Similarly mark multiple of 11,13,17 ,19 as 0



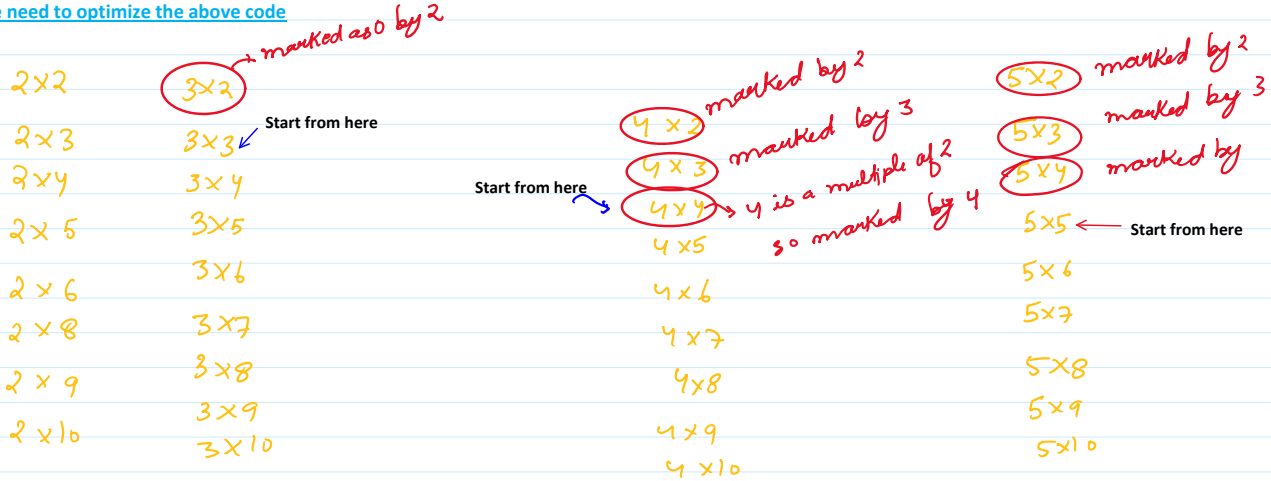
In this whole array who are marked as 1 are prime no. that 2, 3, 5,7,9, 11, 13, 17,19,

Pseudo code :-

```
1) arr[ ] = new int [n+1]
2) for( i = 2 ⟶ n) {
       arr[i] = 1;
   }
3) for (i=2 ⟶ n) {
       if (arr[i]==1){
           for ( j= 2*i ; j<= n; j+=i) {
               arr[i]= 0
           }
       }
4) count = 0
   for (j= 2 ⟶ n){
       if (arr[i] == 1){
           count++;
       }
   }
   return count;
```

If i = 2
j = 2 + i = 4
2 → 4

We need to optimize the above code



Pseudo code :-

```
1)  arr[ ] = new int [n+1]
2)  for(i = 2 ⟶ n) {
        arr[i] = 1;
    }
3)  for (i = 2 ⟶ n) {
        if (arr[i] == 1){
            for (j = i*i ; j <= n; j += i) {
                arr[i] = 0
            }
        }
4)  count = 0
    for (j = 2 ⟶ n){
        if (arr[i] == 1){
            count++;
        }
    }
    return count;
```

i*i

We need to optimized further

Pseudo code :-

```
1)  arr[ ] = new int [n+1]
2)  for(i = 2 ⟶ n) {
        arr[i] = 1;
    }
3)  for (i = 2 ⟶ n) {
        if (arr[i] == 1){
            for (j = i*i ; j <= n  j += i) {
                arr[i] = 0
            }
        }
4)  count = 0
    for (j = 2 ⟶ n){
        if (arr[i] == 1){
            count++;
```

i*i <= n ⇒ i ≤ √n

```
    }
}
return count;
```

Time complexity → $O(N) + O(N \log(\log N)) + O(N)$

space complexity → $O(1)$