30/6/21 Data - Structure & Algorithm

Topics we will learn today →

(1) DSA for interviews vs for Competitive Programming.

(2) Time / Space Complexity

(3) Coding Problem : Trapping Rain Water

(4) Some Problems on Basic Mathematics.

(5) Some Problems on Array / String.

— X —

* Time / Space Complexity →

Que-1 → find the sum of first N natural numbers.

→ { if n = 5,
then, 1+2+3+4+5 = 15. }

So, Method 1: →

```
int fun(int n)
{
    int sum = 0;
    for(int i = 1; i ≤ n; i++)
    {
        sum = sum + i;
    }
    return sum;
}
```

So, [Time Complexity means that how long a program takes to process a given input, or we can say time taken to execute each statement of code in an algorithm.]

Now, we code in time Method-

* So, we can compare two methods for any given problem".

→ So, when talking about worst-case Complexity of the method 1 for the problem of 'N' natural no's- if

that is → [O(n), since the loop is running for the 'n' times so that's why it is "O(n)".]

So, when talking about space-complexity for the problem is → O(1)

Since we are not allocating any space or extra space, like → arr[n]

↓

if we would have used this then our space complexity would be "O(n)"

Now, we are trying to optimize our code in order to achieve better time complexity.

Method-2 :→ (Better) than method 1:-

```
int fun (int n)
{
    return (n * (n+1)/2)
}
```

if n = 5

then, $(5 * \frac{(5+1)}{2})$

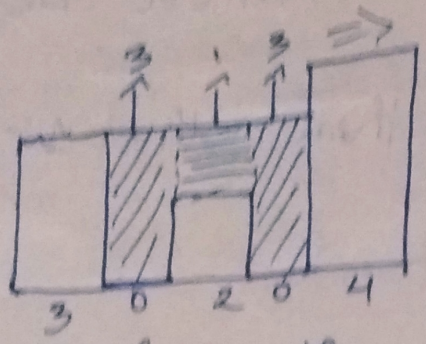
$= 5 * \frac{6^3}{2_1} = 5 * 3 = \boxed{15}$

So, now the Time Complexity is O(1) since we are not iterating something for 'n' times, and space is also O(1) since we are not using any extra space.

* Trapping Rain Problem → We will be given with an array representing the heights of the wall and we have to find the total water trapped between them.

Ex → arr[] = {3, 0, 2, 0, 4}

3+1+3 = 7



we will have the left max and right max for the index we want to calculate.

suppose, if I want to calculate for '2'

then

left max = 3

right max = 4

} min(left max, right max) - height.

min (3, 4) - 2

3 - 2 = 1 → This is amount of water which will be trapped

If now I want to calculate for '0' at index 3 then →

min (3, 4) - 0

= 3 - 0

= 3

```
{
    int ans = 0;
    for(int i = 1; i<n-1; i++)
    {
        toto ans = ans *
O(n) ← int left_max = find_left_max(arr, n, i);
O(n) ← int right_max = find_right_max(arr, n, i);
        ans = ans + min(left_max, right_max) -
                                        arr[i];
    }
    return ans;
}

int find_left_max (int arr[], int n, int i)
{
    int left_max = 0;
    for(int j=0; j<=i; j++)
    {
        left_max = max(left_max, arr[j]);
    }
    return left_max;
}
```

first & last index can be ignored.

↳ Time Comp — (O(n))

we are including the ith value so that we don't face -ve values in some test cases

Total Time Complexity = O(n*n) = O(n²)

```
int find_right_max(int arr[], int n, int i)
{
    int right_max = 0;
    for(int j = n-1; j >= i; j--)
    {
        right_max = max(right_max, arr[j]);
    }
    return right_max;
}
```

Try optimizing it ....

Que-2 → Check whether a no. is Prime or not. →

Eg: n = 5, ans = true
n = 24, ans = false.

```
→ bool is_prime(int n)
{
    for(int i = 2; i <= n; i++)
    {
        if(n % i == 0)
        {
            return false;
        }
    }
    return true;
}
```

Time Complexity = O(n)

or

```
bool is_prime(int n)
{
  for(int i=2; i<=sqrt(n); i++)
  {
    if(n%i==0) return false;
  }
  return true;
}
```

Time Complexity
⇓
O(sqrt(n))
↓
little better than O(n)

Que-3 find all the prime numbers from 1 to n...

→ n=100

from 1 to 100

⇒ 2 3 5 7 11 13 17 - - - -

So, in order to achieve this we have something known as → "Sieve of eratosthenes"

Method 1: (without using sieve)

```
for(int i=2; i<=n; i++)        // O(n)
{
  if(is_prime(i))              // O(sqrt(n))
    cout<<i;
}
```

Total ⇒ O(n * sqrt(n))

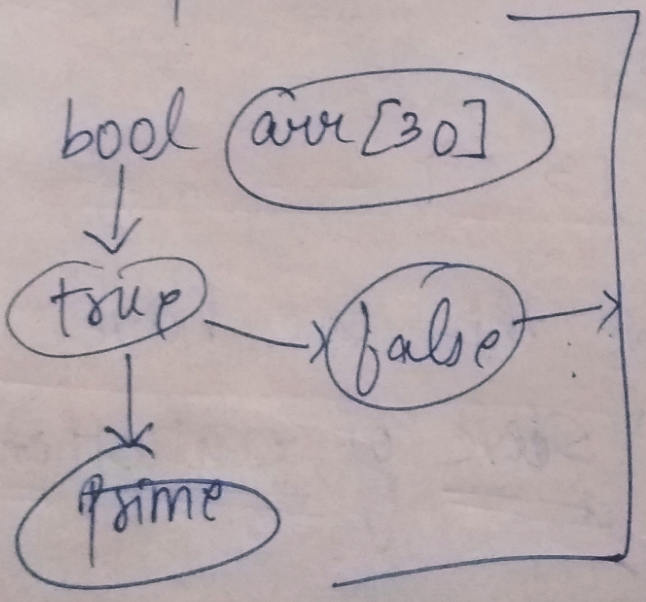
Method 2: (Using Sieve)

Que-4 →

So, in this we will have an array → bool arr[30], i.e., n = 30

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

So, as it is bool, that means all 30 values will be true or considered as prime.



So, we will start with 2 as it is the first prime no, and multiple of 2 will not be prime so we remove them, now, we go to 3 and remove all multiples of 3, now 5.