Time and Space Complexity-3

5 – 7 min

Q1) What is the time complexity ?

Q2) What is the return value of q interms of Big-Oh?

```
function fun(n)
{
        q=0
        for(i=1;i<=n;i++)
        {
                p=0
                for(j=n; j>1; j=j/2)
                {
                        ++p
                }

                for(k=1; k<p; k=k*2 )
                {
                        ++q
                }
        }
        return q;
}
```

TC

```
function fun(n)
{
    q=0
    for(i=1;i<=n;i++)
    {
        p=0
        for(j=n; j>1; j=j/2)
        {
            ++p
        }
        for(k=1; k<p; k=k*2 )
        {
            ++q
        }
    }
    return q;
}
```

$q=0$ ✓

$\rightarrow \log_2^n$

$n * [\, L_1 + L_2 \,]$

$\rightarrow n$ times

$p = 0 \times x \cdots \log_2^n$

$L_1 \rightarrow p = \log_2^n$ ✓

$L_2 \rightarrow \log_2^p = \log_2 \log_2^n$

Here.

Separate

$O(n)$

```
q=0
for(k=1; k<p; k=k*2 )
{
    ++q
}
```

$\rightarrow O(\log_2^p)$

Q1) T.C

Sol) $n * [\, \underbrace{\log_2^n}_{①} + \underbrace{\log_2 \log_2^n}_{②} \,] = n * [\log_2^n] = O(n \log_2^n)$

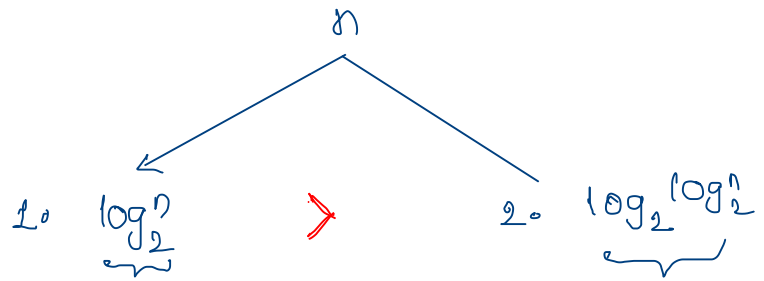
bigger ?

1st one

Q2) $= O(n \cdot \log \log_2^n)$

Ex:- $\log_b b = 1$; $\log_b b^n = n \cdot \underbrace{\log_b b}_{1} = n$.

Ex:- $n = \underbrace{2^{100}}_{\text{very big value}}$ —(Apply $\log_2^n$)→ $= \underbrace{100}_{\text{small}}$ —($\log_2^n$)→ very small.



Q2) given

Assume arr.sort() will take T.C as nlog(n) ✓

↳ Study later classes.

```
function fun(arr,n)
{
    arr.sort()
    for(i=1;i<=n;i++)
    {
        print(arr[i])
    }
}
```

arr.sort() → $n \cdot \log_2 n$

+

for(i=1;i<=n;i++) → n.

print(arr[i]) // $O(1)$

$O(n^2)$

extra

$n\log_2 n + n = O(n\log_2 n)$

Assume arr.sort() will take T.C as nlog(n)

```
function fun(arr,n)
{
    for(i=1;i<=n;i++)
    {
        arr.sort()
        print(arr[i])
    }
}
```

$\rightarrow n$

$\rightarrow n\log_2^n$

$\rightarrow 1$

$n * [n\log_2^n + 1]$

$= n * n\log_2^n$

$= O(n^2 \cdot \log_2^n)$

dep → un-rolling process.

```
for(i=n; i>=1; i=i/2)
{
    for(j=1; j<=i; j=j*2)
    {
        print("DSA Anna")
    }
}
```

→ $\log i$

inner loop runs $\log i$

$i = n \rightarrow \log n$

$+$

$\frac{n}{2} \rightarrow \log \frac{n}{2}$

$+$

$\frac{n}{4} \rightarrow \log \frac{n}{4}$

$+$

$\frac{n}{8} \rightarrow \log \frac{n}{8}$

$\vdots$

$2 \rightarrow \log 2 = 1$

$+$

$1 \rightarrow \log 1 = 0$

$+$

$0 \rightarrow x$

$\log_b(x \cdot y) = \log_b x + \log_b y$

(or)

```
for(j=1; j<=i; j=j*2)   → log i
{
    print("DSA Anna")
}
```

$= \log n + \log \frac{n}{2} + \log \frac{n}{4} + \log \frac{n}{8} + \cdots + 1$

$= \log_2 (n * \frac{n}{2} * \frac{n}{4} * \frac{n}{8} \cdots 1)$

$= n * \frac{n}{2} * \frac{n}{3} * \frac{n}{4} * \frac{n}{5} * \cdots * \frac{n}{n} 1 = n!$

$\log_2(n \to \frac{n}{2} \to \frac{n}{4} \to \frac{n}{8} \cdots \to 1)$

how many such n's we have?

total logn n's are present

$\log_2(n^{\log_2^n}) = \log_2^n \cdot \log_2^n$

$= O((\log_2^n)^2)$

$n \times \frac{n}{2} \times \frac{n}{3} \times \frac{n}{4} \times \frac{n}{5} \times \cdots 1 = \frac{n^n}{\sim}$

we've $n - n's$.

$\therefore n^n$

$5 \times 5 \times 5 \times 5 = 5^4$

T.C :- HW

```
function fun(n)
{
        for(i=n; i>=1; i--)
        {
                for(j=n; j>=i; j--)
                {
                        for(k=1; k<=n^5; k=20*k)
                        {
                                print("Masai");
                        }
                }
        }
}
```

Q1 :-

Qn)

```
function fun(arr,n)
{
        j=0;
        for(i=0; i<n; i++)
        {
                while(j<n and arr[i]<arr[j])
                {
                        j++;
                }
        }
}
```

j=0, i=0	j=0 1, i=1	j=1 2, i=2	j=2 3, i=3	j=3 4, i=4	j=4, i=5
w() ✗	w() ✓	w() ✓	w() ✓	w() ✓	

a[1] < a[0] → F

a[1] < a[1] → F

a[2] < a[1] → T

a[3] < a[2] → T

a[4] < a[3] → F

O(1)

# of times w() runs.

i = 1 → 1

2 → 1

3 → 1

4 → 1

5 → 1

n → 1

1 + 1 + 1 + ---- 1 (n's are there)

n(1) = O(n)

```
function fun(arr,n)
{
        j=0;
        for(i=0; i<n; i++)
        {
            while(j<n and arr[i]<=arr[j])
            {
                j++;
            }
        }
}
```

C_2

$i = 0$:

$j = 0, 1, 2, 3, \ldots, n-1, n$

$a[0] \leq a[j] \rightarrow T$ (j = 1, 2, 3, ... n-times), then F

n-times

i = 1	i = 2	i = 3
j = n.	j = n	j = n
arr-values		
x	x	x . . . x

$n + 0 + 0 + \ldots 0 = n$

$= O(n)$

Note:- Always remember, when you are analyzing the T.C of any program, assume worst case scenario

Best qn:-

```
function(n)
{
      sum=0
      for(i=1;sum<=n;i++)
      {
            sum=sum+i
      }
}
```

1+2+3+4+...n=n(n+1)/2

→ most Imp:
How many # of times Loop runs.? ✓
if it runs for k-times, inside : O(1) ⇒ O(k)

```
for(i=1;i<=k;i++) → O(k)
{
      print(i) → O(1)
}
```

```
function(n)
{
    sum=0
    for(i=1;sum<=n;i++)
    {
        sum=sum+i
    }
}
```

→ OL

sum<=n → fail

sum<=n → T

O_1

runs for

i = 1 2 3 4 5 ..... k k+1 → fail

O(k) ?

sum = 0+1+2+3+4+5+6+7+...+k

sum [1+2+3+...+k] = n → fail

$\frac{k(k+1)}{2} = n$

$k(k+1) = 2n$

$k^2 + k = 2n$

$k = \sqrt{2n}$

↑ const ignore.

$k = \sqrt{n}$ → T.C : $O(\sqrt{n})$

Why TLE comes?

- **Online Judge Restrictions:** TLE comes because the Online judge has some restriction that it will not allow to process the instruction after a certain Time limit given by Problem setter the problem(1 sec).
- **Server Configuration:** The exact time taken by the code depends on the speed of the server, the architecture of the server, OS, and certainly on the complexity of the algorithm. So different servers like practice, CodeChef, SPOJ, etc., may have different execution speeds. By estimating the maximum value of N (N is the total number of instructions of your whole code), you can roughly estimate the TLE would occur or not in 1 sec.

```
MAX value of N                          Time complexity
   10^8                                 O(N) Border case
   10^7                       O(N) Might be accepted
   10^6                                 O(N) Perfect
   10^5                                 O(N * logN)
   10^4                                 O(N ^ 2)
   10^2                                 O(N ^ 3)
   10^9                                 O(logN) or Sqrt(N)
```

- So after analyzing this chart you can roughly estimate your Time complexity and make your code within the upper bound limit.
- **Method of reading input and writing output is too slow:** Sometimes, the methods used by a programmer for input-output may cause TLE.

Space Complexity :-

1. Constant Space Complexity - $O(1)$

Example 1: Finding the Maximum Element

python

```python
def find_max(arr):
    max_val = arr[0]  # constant space
    for val in arr[1:]:
        if val > max_val:
            max_val = val
    return max_val
```

2. Linear Space Complexity - $O(n)$

Example 2: Copying an Array

python

```python
def copy_array(arr):
    new_array = arr[:]  # linear space
    return new_array

# Example usage:
arr = [1, 2, 3, 4, 5]
copied_arr = copy_array(arr)
print(copied_arr)  # Output: [1, 2, 3, 4, 5]
```

Quadratic Space Complexity - $O(n^2)$

Example 4: 2D Array (Matrix) Multiplication

python Copy code

```python
def multiply_matrices(a, b):
    n = len(a)
    result = [[0 for _ in range(n)] for _ in range(n)]  # quadratic space
    for i in range(n):
        for j in range(n):
            for k in range(n):
                result[i][j] += a[i][k] * b[k][j]
    return result

# Example usage:
a = [[1, 2], [3, 4]]
b = [[5, 6], [7, 8]]
result = multiply_matrices(a, b)
print(result)  # Output: [[19, 22], [43, 50]]
```

O(logn) and O(2^n) space , you will see in recursion chapter