Sprint-2 [Day-4]

Time and Space Complexity-4

Let T: be the number of test cases

let T = 100

```
while(T>0)                          ---> T
{
        for(i=1;i<=n;i++)           ---> n
        {
            -> arr.sort()           ---> n·log2n
            -> j=1                      +
               while(j<=n)          ---> log2n
               {
                    j=j*2
               }
        }
        T--   (-1)
}
```

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

$\max = n \cdot \log_2^n$

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

Consider the program

```
void function(int n) {
int i, j, count=0;
for (i=n/2; i <= n; i++)
    for (j = 1; j <= n; j = j*2)
        count++;
}
```

The complexity of the program is

1. O(log n)
2. $O(n^2)$
3. $O(n^2 \log n)$
4. ✓ O(n log n)

What is the complexity of the following code?

```
i = n
while (i>=1){
    for j = 1 to n
        x=x +1
    i = i/2
}
```

a. $\Theta(n)$

b. $\Theta(\log_2 n)$

c. $\Theta(n/\log_2 n)$

d. $\Theta(n \log_2 n)$

→ Units of comp. mem.

→ function's ($\log_2^n$, n^2, n ---)

→ 2, T.c's are given, (identify best on
→ $n \cdot \log_2^n$ $n \cdot \sqrt{n}$

→ Big-oh / order of : OC)

→ code → How to find out it's T.C

Prob

T.C

S_1 S_2 O

```
function f1(n) ⇒ OC(log n)
{
    int x=n;
    while(x>=1)
    {
        print("*")
        x=x/2;
    }
}
```

$\log_2^n$ V/s $\sqrt{n}$

let $n = 2^{1024}$ → 2^{512}

1024

small → best

f_1 best

```
function f2(n)
{
  for(i=1;i<=Math.sqrt(n);i++)  ⇒ OC(√n)
  {
      print("*");
  }
}
```

20 1

Sol T.C

101/111

Nested loop

2D-arr

T.C

20 1

No Nested loop

Space Complexity [S.C]

S.C(p) = C + I.C

C : Constant space [all variables, data structures with fixed size]

I.C : Instance Characteristics

-> I.C includes space for all such variables and data structures whose size is not known before

NOTE:-

- > The space requirement of any Algorithm / Program is bifurcated into two parts

1. Space for inputs

2. work space requirements

w.s component is the space used during " computation " of the Algorithm/ Porgram

i.e , any space that you used, other than storing inputs

T.C O(1) :- Constant

```
function fun(a,b,c)
{
        var p=a
        var q=b
        var r=c
        console.log(p+q+r)
        return a+b+c
}
```

i/p

p, q, r ⇒ O(1)

* Space - comp

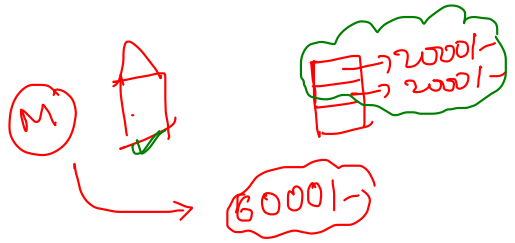
1. Apart from given i/p in the question, see what is the "extra space" that you've created to solve the question.

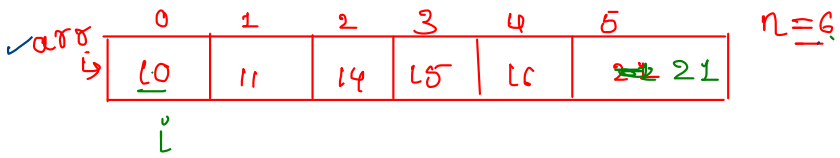
S.C

↳ extra space :- → Variables (i, j, a, b, c, temp .. -) ⇒ O(1)
only

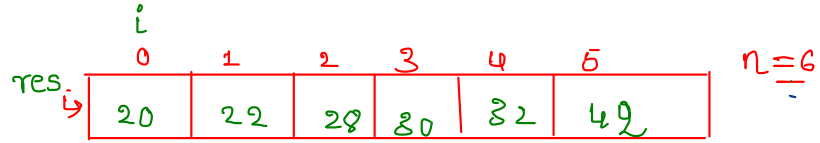
S.C

→ array of size n ⇒ O(n)






```
function fun(arr,n)
{
    let res=new Array(n) ==> // Good practice
                          ↑size

    for(let i=0;i<n;i++)
        res[i]=arr[i]*2

    return res
}
```



i → O(1)

res → n

n+1 ⇒ O(n)

Space :- extra space.

Apart from given i/p

∴ S.C : O(n)

$f(n) = n+1$; $O(n)$

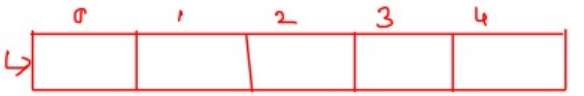
$f(n) \le c \cdot g(n)$

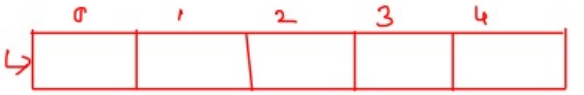
$n+1 \le c \cdot n$ ✓

c, n_0

```
function fun(arr,n)
{
    let res=new Array(n) ==> // Good practice

    for(let i=0;i<n;i++)
        res[i]=arr[i]*2

    let temp=new Array(n)

    for(let i=0;i<n;i++)
        temp[i]=res[i]*2

    return temp
}
```

res ⇒ n

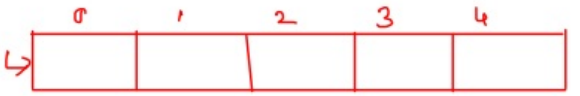




⇒ temp ⇒ n

n + n = 2n

⇒ O(n)



S.C

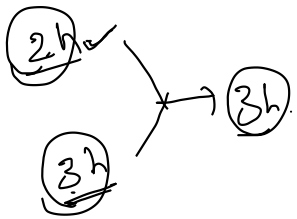
Ex:-

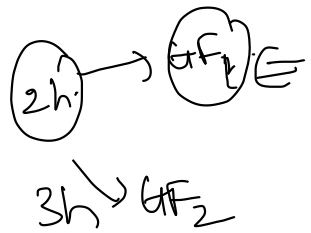
Ex:-

0	1	2	3	4	5	6	7	8	9	10	11
1	1	2	1	1	3	2	3	4	1	1	2

arr

n = 12





obj

Key	Value
1	6
2	3
3	2
4	1

op

w.r.t S.C

① Best case i/p:-

All elements are same in the array.

1 entry (obj) ⇒ O(1)

② worst case i/p:-

All elements are different / distinct

n entry (obj) ⇒ O(n)

arr.length

Always → worst case

O(n)

O(1) / O(n)

Note :-

In general work space requirements for Algorithm is the order of time complexity

$$W.S(Algorithm) = O(T.C)$$

→ Time Limit Exceed

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.

$1 \le N \le 10^9$

MAX value of N	Time complexity
10^8	O(N) Border case (code)
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.

Ex:- 6 Sit

XYZ → PF

1 → 2sec (2·1sec)

2

3

:-C : O(n)

Java Script In-built Methods and it's Time Complexity

Mutator Methods.

1. push() - O(1)
2. pop() - O(1)
3. shift() - O(n)
4. unshift() - O(n)
5. splice() - O(n)
6. sort() - O(n log(n))

Accessor methods

1. concat() - O(n)
2. slice() - O(n)
3. indexOf() - O(n)

Iteration methods

1. forEach() - O(n)
2. map() - O(n)
3. filter() - O(n)
4. reduce() - O(n)

THANK - YOU