Sakshi Rajvanshi
CST SPL-2
01

Tutorial - 2

Ques1. What is the time complexity of below code and how?

```
void fun (int n) {
    int j = 1, i = 0;          → O(1)
    while ( i<n) {
        i = i+j;
        j++; }}
```

$i=0 \rightarrow i = 0+1 = 1, \; j=2$

$i=1 \rightarrow i = 1+2 = 3, \; j=3$

$i=3 \rightarrow i = 3+3 = 6, \; j=4$

$i=6 \rightarrow i = 6+4 = 10, \; j=5$

$i=k \rightarrow i = \underbrace{k+x}_{sum}, \; j=n$

from the above pattern we can clearly see that it is just a sum of first n- numbers.

OR

j	1	2	3	4 ----	n
i	1	3	6	10 ----	k

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

$$\frac{\boxed{k^2}+k}{2} > n \xrightarrow{} \text{dominating term}$$

$$k^2 = n$$

$$k = \sqrt{n}$$

Time complexity $= O(k)$ or $O(\sqrt{n})$

$= O(\sqrt{n})$ Ans.

Ques2. Write recurrence relation for the recursive function that prints fibonacci series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity of this program and why?

```
int fib (int n)  → T(n)
{
    if (n<=1)  → O(1)
        return n;
    return fib(n-1) + fib(n-2);  → T(n-1)+T(n-2)
}
```

$$T(n) = \begin{cases} 1 , & n<=1 \\ T(n-1)+T(n-2) , & \text{otherwise.} \end{cases}$$

$T(n) = T(n-1) + T(n-2) + C$

$= 2T(n-1) + C$

[from the approximation $T(n-1) \sim T(n-2)$]

$T(n) = 2(2T(n-2)+C)+C$

$T(n) = 4T(n-2) + 3C$

$= 8T(n-3) + 7C$

$\vdots$

$= 2^k T(n-k) + (2^k - 1)C$

we know $T(1) = 1$, so

$T(n-k) = T(1) = 1$

$n-k = 1$

$\boxed{k = n-1}$

$T(n) = 2^{n-1} T(1) + (2^{n-1} - 1)C$

Time Complexity $= O(2^n)$

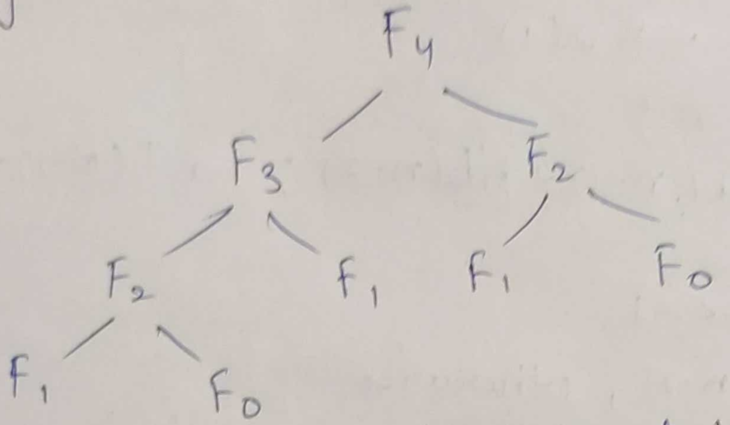
for space complexity

we know that:

Space required ∝ Max. depth of recursive tree.

because that is the max. no. of elements that can be present in the implicit function call stack.

eg. f_4



Space complexity = O(4)

for n-elements, $\boxed{S.C = O(n)}$

Ques 3. Write programs which have complexity

1) n(log n)

```
A()
{
int i,j;
for (i=1; i<=n; i++) ⟶ O(n)
  { for (j=1; j<=n; j=j/2) ⟶ O(log n)
      {
         printf("*");
      }
  }
}
```

2) n^3

```
    A()
 { int i, j, k;
 for (i= 0; i<= n; i++)  →  O(n)
  {
    for (j=0; j<=n; j++)  →  O(n)
     {
       for (k=0; k<=n; k++)  →  O(n)
         { printf("*");
            }
 } } }
```

Ques 4. Solve the following recurrence relation

$T(n) = T(n/4) + T(n/2) + cn^2$.

$T(n/4) \le T(n/2)$

So, we can write this equation as

$T(n) = T(n/2) + T(n/2) + cn^2$

$T(n) = 2T(n/2) + cn^2$

Comparing with Master equation, we get

$a = 2,\ b = 2,\ k = 2$ & $p = 0$

now $\boxed{a < b^k}$ So, $a < b^k$ & $p = 0$.

$2 < 2^2$

$T.C = O(n^k \log^p n)$

$= O(n^2 (\log n)^0)$

$\boxed{TC = O(n^2)}$

Ques 5. What is the time complexity of following function fun()?

```
int fun(int n){
for (int i=1; i<=n; i++) {
    for (int j=1; j<n; j+=i) {
        // some O(1) task
}}}
```

j incrementation depends on i, ∴ we unroll all loops

$i=1$	$i=2$	$i=3$	------	$i=n$
$j=1$ to n	$j=1$ to n	$j=1$ to n		$j=1$ to n
→ n-times	→ n/2 times	→ n/3 times		→ 1 time

So, $T.C = n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}$

$= n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$

$\boxed{TC = O(n \log n)}$ Ans.

Ques6. what should be the time complexity of

```
for (int i=2; i<=n; i= pow(i,k))
{
    some O(1) expressions or statements
}
```

where, k is a constant.

$i=2$ $\quad$ $i=2^k$ $\quad$ $i=(2^k)^k = 2^{k^2}$

'2 to n times' $\quad$ '2^k to n times' $\quad$ "2^{k^2} to n times"

- - - - $i = 2^{k^{\log_k \log n}}$

at that time, $2^{k \log_k(\log n)} = 2^{\log n} = n$

Total $\boxed{T.C = O(\log \log n)}$

OR

for

$i=2 \mid i=2^k \mid i=2^{k^2}$ - - - - $i=2^{k^l}$

$2^{k^l} = n$

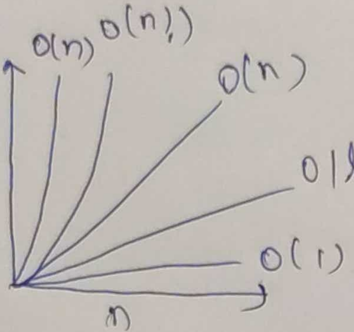
$k^l = \log_2 n$

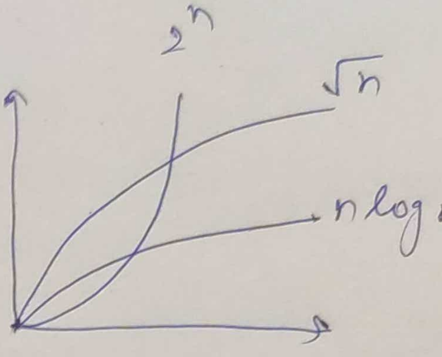
$l = \log_k \log_2 n$ $\quad$ $\boxed{T.C = O(\log_k \log_2 n)}$ Ans.

Ques 8. Arrange the following in increasing order of rate of growth.

a) $n, n!, \log n, \log\log n$, root(n), $\log(n!)$, $n\log n$, $\log^2(n)$

$2^n, 2n^{2n}, 4^n, 100.$





$100 < \log\log n < \log^2(n) < \log(n) < \log n! < n\log n <$

root $n < n < n! < (2)^{2^n} < 4n, n^2, 100$

b) $2(2^n), 4n, 2n, 1, \log n, \log(\log n), \sqrt{\log(n)}$,

$\log 2n, 2\log(n), n, \log(n!), n!, n^2, n\log n.$

$1 < \log(\log n) < \sqrt{\log(n)} < \log n < \log 2n, 2\log n < n! <$

$\log(n!) < n\log n < n < 2n < 4n < n^2 < 2(2^n)$

c) $8^{(2n)}, \log_2(n), n\log_6 n, n\log_2 n, \log(n!), n!$

$\log_8(8^n), 96, 8n^2, 7n^3, 5n.$

$96 < \log_8 8^n < \log_2 n < \log(n!) < n\log_6 n < n\log_2 n <$

$5n < 2n^2 < 7n^3 < n! < 8^{(2n)}.$