Shubham Lingwal
18-G

Tutorial-2

Ques-1 What is the time complexity of below code & how?

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

i=0 → i=0+1=1, j=2
i=1 → i=1+2=3, j=3
i=3 → i=3+3=6, j=4
i=K → i=K+n (sum), j=n

From the above pattern we can clearly see that its just a sum of first n- numbers.

OR

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

$\frac{K(K+1)}{2} > n \Rightarrow \frac{K^2+K}{2} > n$ (dominating term)

$K^2 = n$

$K = \sqrt{n}$

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

Q.2 Write recurrence relation for the recursive fn that prints fibonacci series. Solve the recurrence relation to get time complexity of the program. What will be the space complexity of this program & why?

Ans:
```
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

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

$$n - k = 1$$

$$k = n-1$$

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

$$T.C = O(2^n)$$

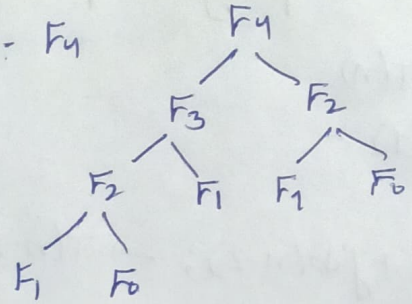
For Space Complexity

We know that

Space required $\propto$ 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)

S.C = O(n)

Q.3. Write programs which have complexity

1) $n(\log n)$

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

2) n^3

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

Que-4 Solve the following recurrence relation

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

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

So, we can write this eqn as

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

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

Comparing with Master theorem, we get

$a=2, b=2,$ ~~[illegible]~~ $K=2$ & $p=0$

now, $a < b^K$ so, $a < b^K$ & $p=0$

$$2 < 2^2$$

$$T.C = O(n^K \log^p n) = \mathbf{O(n^2)}$$

Que-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){
      statement;
    }}}
```

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

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

$T.C = O(n \log n)$

Que-6 what should be the time complexity ?

```
for(int i=2; i<=n; i=pow(i,k)
  { statement;
  }
```

Ans: i=2 $i = 2^k$ $i = (2^k)^k = 2^{k^2}$

2 to n-times 2^k to n-times $\cdots$ 2^{k^2} to n-times

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

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

Total, $T.C = O(\log \log n)$

OR

for $i=2 \mid i=2^k \mid i=2^{k^2} \ldots i=2^{k^l}$

$$2^{k^l} = n$$

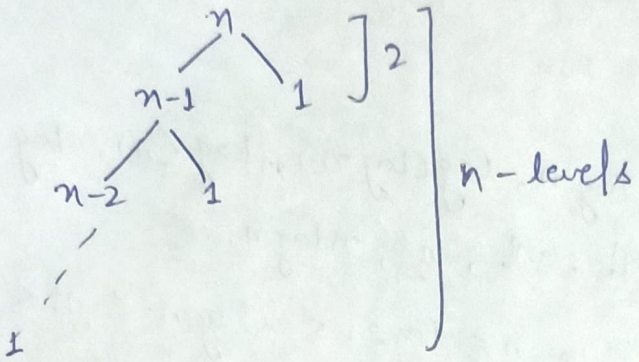
$$k^l = \log_2 n$$

$$l = \log_k \log_2 n$$

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

Que-7. Given algo divides array in 99% & 1% part

$$T(n) = T(n-1) + O(1)$$



→ n work is done at each level for merging

$$T(n) = [T(n-1) + T(n-2) + \ldots\ldots T(1) + O(1)] \times n$$

$$= n \times n$$

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

Lowest height = 2

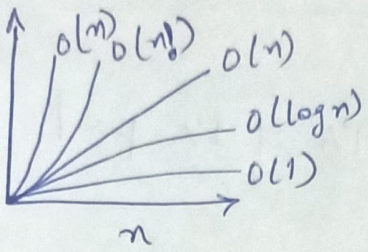
Highest height = n

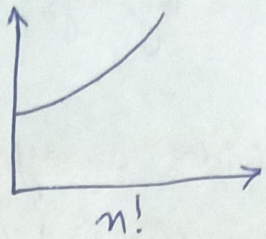
$$\boxed{\therefore \text{Difference} = n-2} \quad n > 1$$

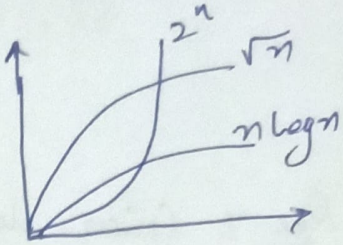
The given algo produces linear result.

Que-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, 2^{n2^n}, 4^n, 100.







$100 < \log\log n < \log^2(n) < \log(n) < \log n! < n\log n < \sqrt{n} < n <$

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

b) $2(2^n)$, 4^n, 2^n, 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, $8n^2$, $7n^3$, $5n$

$96 < \log_8 8^n < \log_2 n < \log(n!) < n\log_6 n < n\log_2 n < 5n <$

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