Tutorial-2

1) What is the time complexity of below code & how.

```
void fun (int n)
{ int j=1; i=0;
While (i<n){
  i=i+j;
  j++;}
}
```

Time Complexity - O(sqrt n)

1st time i=1

2nd time i=3 (. i=1+2)

3rd time i=6 (i=1+2+3).

nth time $i = \frac{x(x+1)}{2} = x^2 < n$

$x = sqrt(n)$

2) Write recurrence relation for the recursive function that Prints fibonacci series. solve the recurrence relation to get complexity of the program. what will the space complexity of this program & why.

Let T(0)=1

Sol:- * fib(n) = fib(n-1)+ fib(n-2)

```
fib(n):
    if n<=1
        return1
    return fib(n-1) + fib(n-2)
```

Time Complexity

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

$= 2T(n-2) + c$ $\quad$ (let $T(n-1) \simeq T(n-2)$)

$T(n-2) = 2*(2T(n-2-2)+c)+c$

$= 2*(2T(n-4)+c)+c$

$= 4T(n-4)+3c$

$T(n-4) = 2*(4T(n-4)+3c)+c$

$= 8T(n-6)+7c$

$= 2^k \times T(n-2k) + (2^k-1)c$

$n-2k=0 \Rightarrow n=2k \quad K = n/2$

$T(n) =$ [illegible]

$T(n) = 2^n * T(0) + (2^n-1)c$

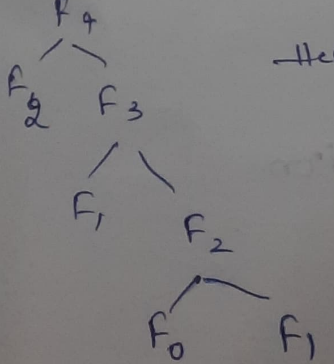
$2^n * 1 + 2^n c - c$

$2^n(1+c) - c$

$\simeq 2^n$ //constants can be ignored

$O(2^n)$

Space complexity The space is proportional to the maximum depth of the recursion tree



Hence the space complexity of Fibonacci recursive is O(N)

③ Write programs which have complexity - n(logn), n^3, log(logn).

Sol :- ⇒ Merge sort - nlogn.

→ for time complexity - n^3

We can use three nested loops - $O(n^3)$

```
for (int i=0; i<n; i++)
{
    for(int j=0; j<n; j++)
    {
        for (int k=0; k<n; k++)
        {
            some O(1) expressions.
        }
    }
}
```

⇒ for time complexity - log(logn)

We can use the following function

```
for(int i=2 ; i<n; i= pow(i,c))
{
// some O(1) expressions
}
```

where k is constant.

→ for time complexity nlogn

We can use the following function

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

(4) Solve the following recurrence relation $T(n) = T(n/4) + T(n/2) + cn^2$

$\frac{T(n)}{2} \geq T(n/4)$

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

using master's method $T(n) = aT(n/b) + f(n)$

$a \geq 1, b > 1, c = \log_b a$ comparing n^c & $f(n)$

we get $c = \log_2 2 = 1$

$f(n) > n^c$

$T(n) = \theta(f(n))$

$\Rightarrow \theta(n^2)$

(5) what is the time complexity of the following function

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

Sol:- for $i = 1 \rightarrow j = 1, 2, 3, 4 ---- n$ (run for n times)

for $i = 2 \rightarrow j = 1, 3, 5 ----$ (run for $n/2$ times)

for $i = 3 \rightarrow j = 1, 4, 7 ---$ (run for $n/3$ times)

$\therefore T(n) = n + n/2 + n/3 + n/4 + ----$

$n(1 + 1/2 + 1/3 + 1/4 + ----)$

$\Rightarrow n \sum_{x=1}^{n} \int \frac{1}{x} \Rightarrow n \int_1^n \frac{dx}{x} \Rightarrow \log x \Big]_1^n$

$n \log n$.

$\therefore$ The time complexity of following function is $n \log n$.

(6) What should be the time complexity of following funcs

for (int i = 2 ; i < n ; i = pow(i, k))

{

// some O(1) expressions or statements.

}

where k is a constant

Sol:- for first iteration $i = 2$

2nd iteration $i = 2^{\wedge k}$

3rd iteration $i = (2^k)^k = 2^{k^2}$

⋮

n^{th} iteration $i = 2^{k^i}$ loop ends at $2^{k^i} = n$

apply log $\log n = \log 2^{k^i} \Rightarrow k^i = \log n$

again apply log $\log(k^i) = \log n \Rightarrow i = \log_k(\log n)$

(7) Write a recurrence relation when Quick Sort repeatedly divides the array in to two parts of 99 % & 1 %

Derive the time complexity in this case. show the recursion tree while deriving time complexity & find the difference in heights of both the extreme parts. what do you understand by this analysis?

Sol:- Array is divided into 99 % & 1 %

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

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

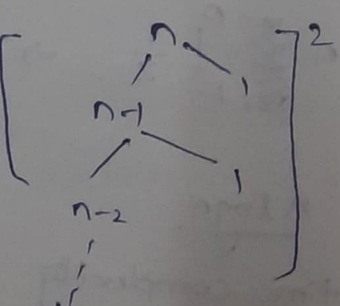
$= n \times n$

$\therefore T(n) = O(n^2)$

Lowest height = 2

Heightest height = n

$\therefore$ diff = n − 2 $n > 1$

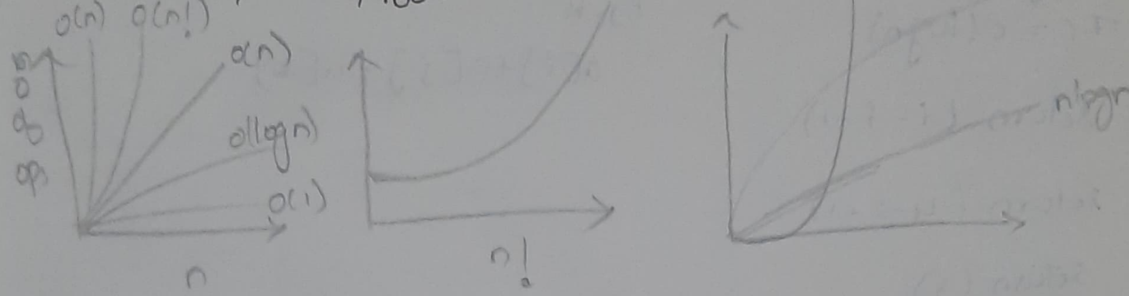


The given algorithm provide linear result.

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



Sol :- $100 < \log\log n < \log^2(n) < \log(n) < \log n! < n\log n$

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

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

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

Sol:- $1 < \log(\log n) < \sqrt{\log(n)} < \log n < \log 2n < 2\log n < n!$

$< \log(n!) < n\log n < n < 2n < 4n, n^2 < 2(2^{\wedge})$.

③ $8^{\wedge}(2n)$, $\log_2(n)$, $n\log_6(n)$, $n\log_2(n)$, $\log(n!)$, $n!$

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

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

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

—— x ——