DESIGN AND ANALYSIS OF ALGORITHMS

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

$x = \text{sqrt}(n)$

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
        return 1
    return fib(n-1) + fib(n-2)
```

Time complexity

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

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

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

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

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

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

$$= 8T(n-3) + 7c.$$

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

$$n - 2k = 0$$

$$n = 2k.$$

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

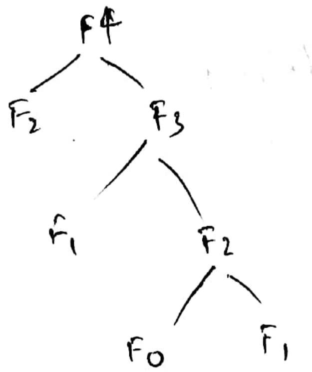
$$2^n * 1 + 2^n c - c$$

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

$$\simeq 2^n$$

$$O(2^n)$$

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), n3, log(logn).

Sol Merge sort - nlogn.

⇒ For time complexity - n^3

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

⇒ For time complexity - log(logn)

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

where k is constant

⇒ For time complexity nlogn.

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

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

using master method $T(n) = aT(n/b) + f(n)$

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

We get $c = \log_2^2 = 1$

$f(n) > n^c$

$T(n) = O(f(n))$

$\Rightarrow O(n^2)$

(5) What is the complexity of following function.

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

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

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

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

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

$n/1 + 1/2 + 1/3 + 1/4 + \ldots)$

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

$n \log n$.

$\therefore$ Time complexity $= n\log n$//.

(6) What should be the time complexity of

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

where k, is a constant.

for first iteration $i = 2$.

2nd iteration $i = 2^k$.

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

⋮

nth 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 $\log(k^i) = \log n$.

$\Rightarrow i = \log_k(\log n)$

Q. Write a recurrence relation when Quick sort repeatedly divides the array in to two parts of 99% & 1%. Drive the time complexity in this case. Show the recursion tree while driving time complexity & find the difference in heights of both the extreme paths. What do you understand by this analysis?

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

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

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

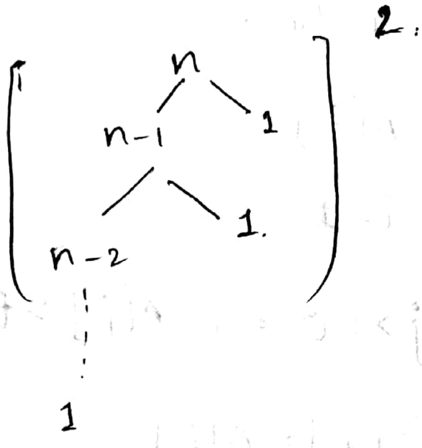
$= n \times n$.

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

lowest height = 2.

Highest height = n.

$\therefore$ diff = n − 2 $n > 1$.



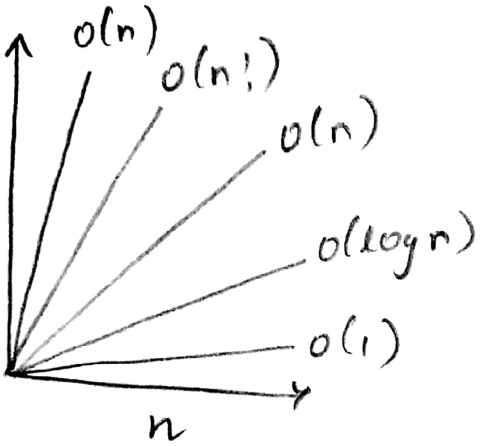
$\therefore$ The given Algorithm provide linear result.

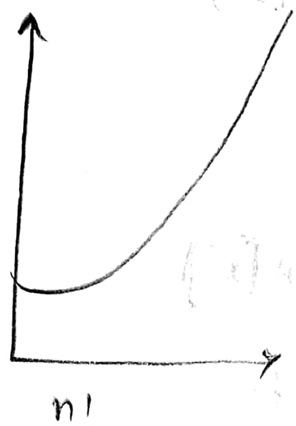
(1) Arrange the following in increasing order of rate

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

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

Sol:





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

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

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

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

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

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

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

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

Sol:- $96 < \log_8 n < \log_2 n < \log(n!) < n\log_6 n < n\log_2 n <$

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