Design and Analysis of Algorithms
Tutorial-2

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

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

values after execution of while loop

1st time $i = 1$

2nd time $i = 1 + 2$

3rd time $i = 1 + 2 + 3$

4th time $i = 1 + 2 + 3 + 4$

let for i^{th} time $i = (1 + 2 + 3 + \cdots i) < n$

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

$$i^2 < n$$

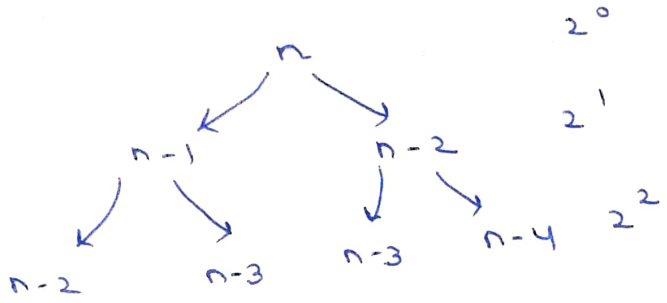
$$i = sqrt(n) = \sqrt{n}$$

$\therefore O(\sqrt{n})$

2. 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 & why?

Recurrence Relation for Fibonacci →

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



1, 2, 4, 8, ...

$r = 2$

$$S_n = \frac{a(r^n - 1)}{r-1} = 1 \cdot \frac{(2^n - 1)}{2-1} = 2^n - 1$$

$\Rightarrow O(2^n)$

For fibonacci series recursive implementation or any recursive algorithm the space required is proportional to maxm depth of tree (i.e. that is the maxm no. of elements present in implicit func. call stack)

$\Rightarrow$ Space Complexity $\rightarrow O(n)$

3. write programs which have complexity

n(logn) , n^3 , log(logn)

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

O(nlogn)

```
int A (int n) {
    for (int j = 1 ; j < i ; j * = 2) {
        for (int k = j ; k >= 1 ; k /= 2) {
            if (n == 0) return 1
            else
                for (int z = 0; z < n ; z++) {
                    // do something
                }
        }
}
```

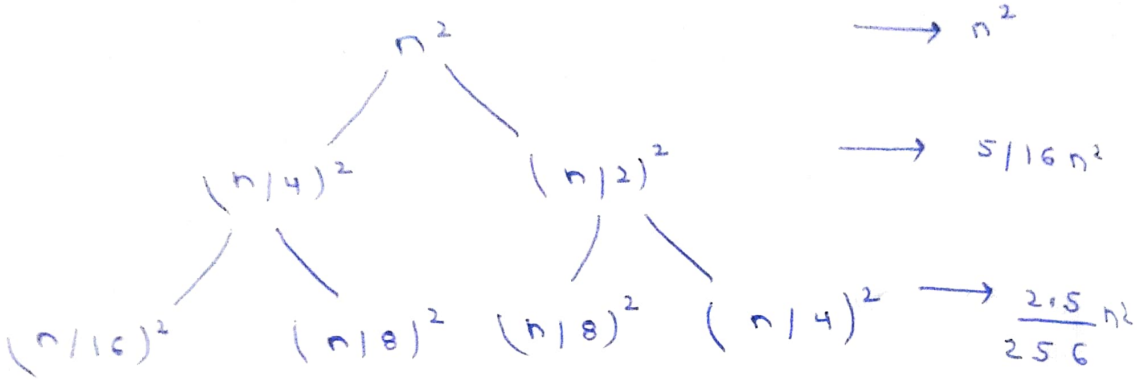
∴ O(log(logn))

```
for (i = 0 ; i < N; i++) {
    for (j = 0 ; j < N ; j++) {
        for (j = 0 ; j < N ; j++) {
            // do something          ∴ O(n^3)
        }
}}
```

4. Solve the following recurrence relation

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + cn^2$$

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + cn^2$$



$O(1)$

Total

$$n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

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

5. What is the time complexity of the following

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

This loop will run n/i times

$\therefore O(n \log n)$

6. What should be time complexity

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

If $n = 16$

2	$2 <= 16$
4	$4 <= 16$
16	$16 <= 16$

$\therefore \hookrightarrow O(2^n)$

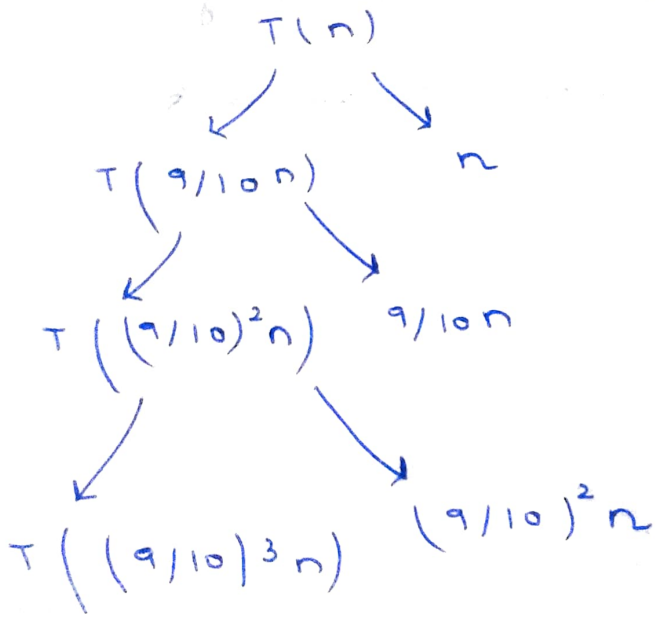
7

For case1

$$F(N) = F(9N/10) + F(N/10) + N$$

Since $F(N/10)$ which is 10% of the sublist always end earlier we can thus simplify problem to

$$F(N) = F(9N/10) + N$$



$\therefore O(n \log n)$

8.

Arrange the following in inc order of rate of growth

a. n, n!, logn, loglogn, rootn, log n!, nlogn, log^2n, 2^n, 2^(2^n), 4^n, n^2, 1

1 < loglogn < log^2n < logn, √n < n < log n! < nlogn, n^2 < 2^n < 2·2^n < 4^n < n!

b. 2(2^n), 4n, 2n, 1, logn, loglogn, √logn, log^2n, 2logn, nlog n, n!, n^2, nlogn

1 < loglogn < √logn < logn < log^2n < 2logn < 2n < 4n < nlogn < nlog n! < n^2 < 2^2n < n!

c. 8^2n, logn, nlogn, nlog^2n, n!, log_8n, 96, 8n^2, 7n^3, 5n

96 < log_8n < logn < 5n < log_6n < log_2n < 8n^2 < 7n^3 < 8^2n < n!