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).

⋮

n^{th} 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 cromplexity 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

③ Write programs which have complexity - n(logn), n^3, log(logn) ④ Sc

Sol :- → Merge sort - nlogn

Sol

→ 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.
      }
    }
  }
```

⑤

→ for time complexity - log(log n)

We can use the fdlowing fonction

```
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 fenction

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

③ Write programs which have complexity - n(logn), n^3, log(logn) ④ Sc

Sol :- → Merge sort - nlogn

Sol

→ 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.
      }
    }
  }
```

⑤

→ for time complexity - log(log n)

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

Sol $T(n) = 2T(n/2) + cn^2$ $\quad T(\frac{n}{2}) \geq T(n/4)$

using masters 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)

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

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

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

$n \log n$.

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