Tutorial - 2

Name → Muskan Agarwal
Section → F
Roll no. → 15
Univ. Roll no. → 2016860

Q1- What is the time complexity of below code and how?

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

⇒

j = 1	i = 1
j = 2	i = 1+2
j = 3	i = 1+2+3

m-levels

for (i)

∴ $1+2+3+ \cdots + < n$

$1+2+3+m < n$

∴ $\frac{m(m+1)}{2} < n$

$m \approx \sqrt{n}$

By summation method

$\sum_{i=1}^{m} 1 \Rightarrow 1+1+ \cdots + \sqrt{n}$ times

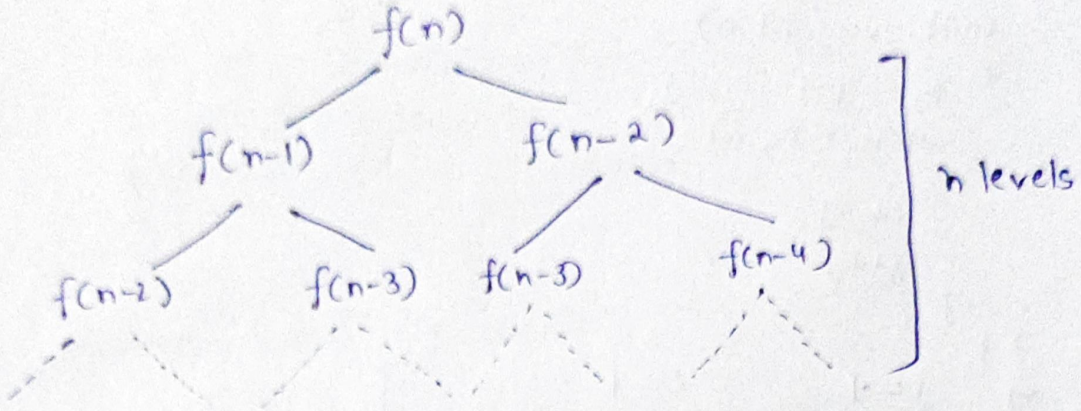
$$\boxed{T(n) = \sqrt{n}}$$

Q2- Write recurrence Relation for the recursive function that prints Fibonacci Series. Solve the RR to get time & space comp.

⇒ For Fibonacci Series

$f(n) = f(n-1) + f(n-2)$ $\quad f(0)=0$, $f(1)=1$

By forming a tree



∵ At every function call we get 2 f^n calls for n levels

we have = 2 x 2 ..... n times

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

Maximum Space

Considering Recursion

Stack: no. of calls maximum = n

For each call we have space complexity O(1)

$\therefore \boxed{T(n) = O(n)}$

Without considering Recursive stack:

each call we have time complexity O(1).

$\therefore \boxed{T(n) = O(1)}$

Q3- Write program which have complexity
$n(\log n)$, n^3 , $\log(\log n)$

Sol- $n \log n$ → Quick sort

```
void quicksort (int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition (arr, low, high);
        quicksort(arr, low, pi-1);
        quicksort(arr, pi+1, high);
    }
}
int partition (int arr[] , int low, int high)
{
    int pivot = arr[high];
    int i = (low -1);
    for(int j= low ; j <= high -1 ; j++)
    {
        if ( arr[j] < pivot)
        {
            i++;
            swap (&arr[i] , &arr[j]);
        }
    }
    swap (&arr[i+1] , &arr[high]);
    return (i+1);
}
```

2) n^3 → Multiplication of 2 square matrix

```
for(i=0; i<r1 ; i++)
for(j=0; j<c2 ; j++)
    for(k= 0; k<c1; k++)
    {
        res[i][j] += a[i][k] * b[k][j];
    }
```

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

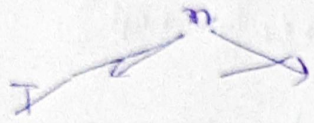
3) log(log n)

```
for (i = 2; i < n; i = i * i)
{
    count ++;
}
```

Q4- Solve the following Recurren Relation

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

⇒



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

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

So we can write.

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

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

Now, Solving by Maste's Theorem

$f(n) = n^2$

$n^c = n^{\log_b a} = n^{\log_2 2} = n^1$

$\therefore f(n) > n^c$

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

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

Q5- What is the time complexity of the following function?

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

⇒ For

i	j
1	1, 2, 3 . . . . n
2	1, 3, 5, 7 . . . n/2 times
3	1, 4, 7, 10 . . . n/3 times
⋮	
n	the inner loop will run for $\frac{n}{n}$ times

So the total time complexity = $n + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n}$

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

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

$= n \sum_{k=1}^{n} \frac{1}{k}$

$= n \log n$

Q6 - What should be the time complexity of

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

where k is a constant

⇒ for i

2^1
2^k
2^{k^2}
2^{k^3}
.
.
2^{k^m}

$2^{k^m} <= n$

applying log both sides

$k^m \log_2 2 = \log_2 n$

again app. $\log_k$ on both sides

~~$m \log k = \log$~~

$m \log_k k = \log_k (\log_2 n)$

$m = \log_k (\log_2 n)$

$\because T(n) = \sum_{i=1}^{m} 1$

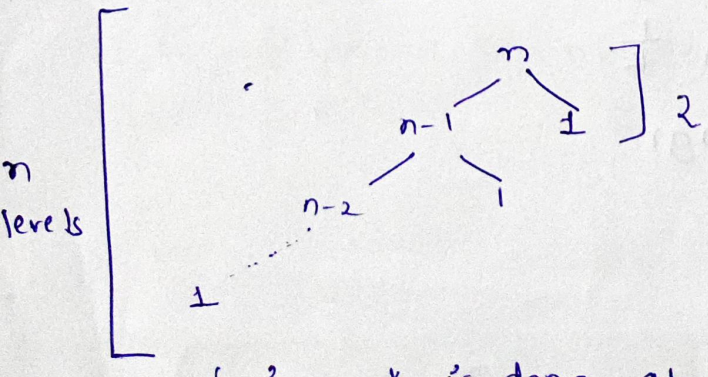
$= 1 + 1 + + 1 \ldots m$ times

$T(n) = O(\log_k \log n)$

Q7 - Write a Recurren Relation . . . . . analysis?

⇒ Given algorithm divides array in 99% & 1% part

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



'n' work is done at each level.

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

$= n \times n$

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

Lowest height = 2

highest height = 2

$\therefore$ difference $= n-2 \quad n \geq 1$

The given algorithm produces linear result.

Q8- Arrange the following in increasing order of rate of growth:

(a) n, n!, logn, loglogn, root(n), log(n!), nlogn, log^2(n), 2^n, 2^(2^n), 4^n, n^2, 100

⇒ $100 < \log\log n < \log n < (\log n)^2 < \sqrt{n} < n < n\log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n}$

(b) 2(2^n), 4n, 2n, 1, log(n), log(log(n)), $\sqrt{\log(n)}$, log2n, 2logn, n, log(n!), n!, ~~n2~~ n2, nlog(n)

⇒ $1 < \log\log n < \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n < 2n < 4n < \log(n!) < n^2 < n! < 2^{2^n}$

(c) 8^{2n}, $\log_2(n)$, $n\log_6(n)$, $n\log_2(n)$, log(n!), n!, $\log_8(n)$, 96, 8n2, 7n3, 5n

⇒ $96 < \log_2 n < \log 2n < 5n < n\log_6(n) < n\log_2 n < \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$