Tutorial - 2

Name: HARSH RASTOGI Class Roll No- 6

Section: F University Roll No- 2016761

1. 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++;
    }
}
```

Ans:-

$j=1$	$i=1$	m-level
$j=2$	$i=1+2$	
$j=3$	$i=1+2+3$	

for(i)

$\because 1+2+3+\ldots\ldots + < n$

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

$m \approx \sqrt{n}$

By summation method,

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

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

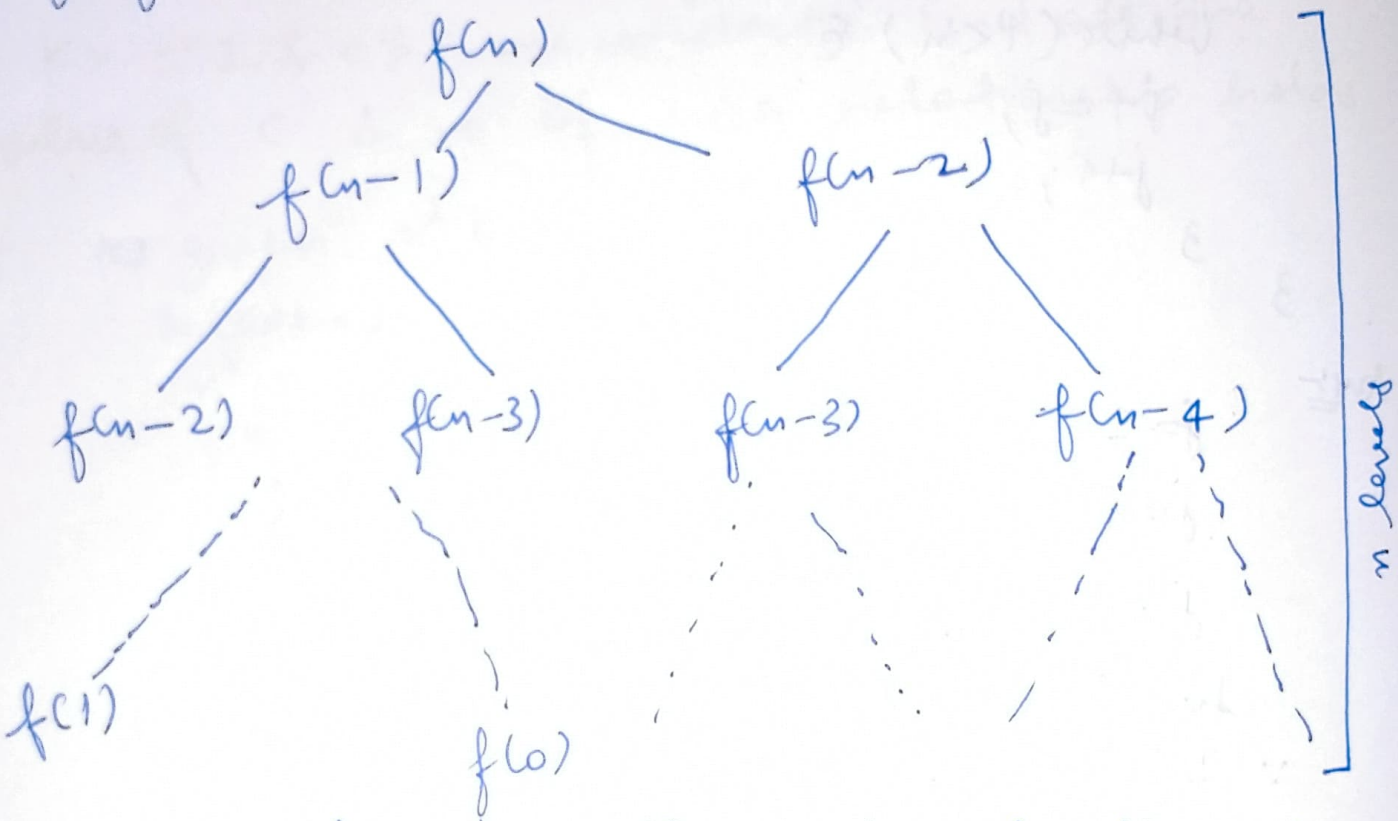
2. Write recurrence relation for function that prints Fibonacci series. Solve it to get the time complexity. What will be the space complexity and why?

Ans= For Fibonacci series

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

$$f(0) = 0$$
$$f(1) = 1$$

By forming a tree,



∵ At every function call, we get 2 function calls

∴ for levels

we have = 2×2 . . . . . n times

$$T(n) = 2^n$$

Maximum space:

considering residance

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)

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

3. Write programs which have complexity :
$n(\log n)$, n^3, $\log(\log n)$

Ans:-

1) $n(\log n)$ → 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];
    }
```

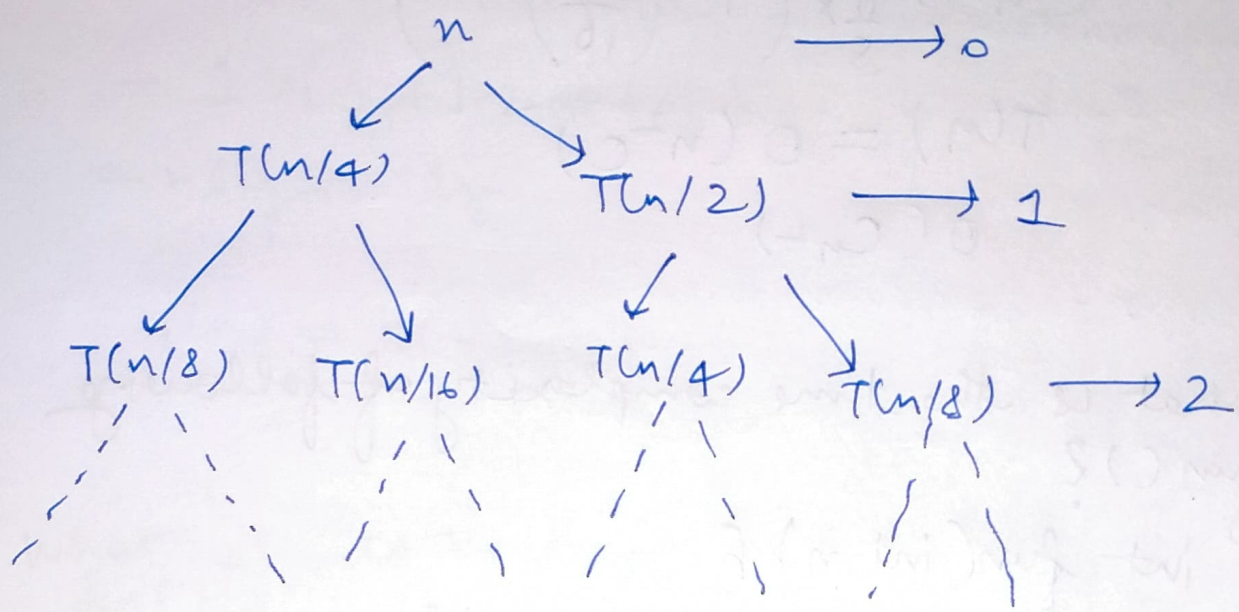
3.) log(logn)

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

4. Solve the following recurrence relation.

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

Ans.



At level

$0 \rightarrow Cn^2$

$1 \rightarrow \frac{n^2}{4^2} + \frac{n^2}{2^2} = \frac{5n^2}{16}$

$2 \rightarrow \frac{n^2}{8^2} + \frac{n^2}{16^2} + \frac{n^2}{4^2} + \frac{n^2}{8^2} = \left(\frac{5}{16}\right)^2 n^2 C$

$\vdots$

max level $= \frac{n}{2^k} = 1$

$= k = \log_2 n$

$T(n) = C(n^2 + (5/16)n^2 + (5/16)^2 n^2 + \ldots\ldots + (5/16)^{\log n} n^2)$

$T(n) = Cn^2[1 + (5/16) + (5/16)^2 + \ldots\ldots + (5/16)^{\log n}]$

$$T(n) = (n^2 \times 1 \times \left(\frac{1-(5/16)^{\log n}}{1-(5/16)}\right)$$

$$T(n) = Cn^2 \times \frac{11}{5} \times \left(1-\left(\frac{5}{16}\right)^{\log n}\right)$$

$$T(n) = O(n^2 C)$$

$$O(Cn^2)$$

5. What is the time complexity of following fun()?

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

Ans:- for

i	j	
1	1	$j = (n-1)/i$ times
2	1+3+5	
3	1+4+7	
⋮	⋮	
n	1+5+9	

$$\sum_{i=1}^{n} \frac{(n-1)}{i}$$

$$\because T(n) = \frac{(n-1)}{1} + \frac{(n-1)}{2} + \frac{(n-1)}{3} + \cdots + \frac{(n-1)}{n}$$

$$T(n) = n[1 + 1/2 + 1/3 + \cdots\cdots\cdots 1/n] - 1 \times [1 + 1/2 + 1/3 + \cdots\cdots 1/n]$$

$$= n\log n - \log n$$

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

6. What should be time complexity of

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

where, k is a constant.

Ans:- for i

2^1
2^k
2^{k^2}
2^{k^3}
$\vdots$
2^{k^m}

where,

2^{k^m}

$k^m = \log_2 n$

$m = \log_k \log_2 n$

$\because \sum_{i=1}^{m} 1$

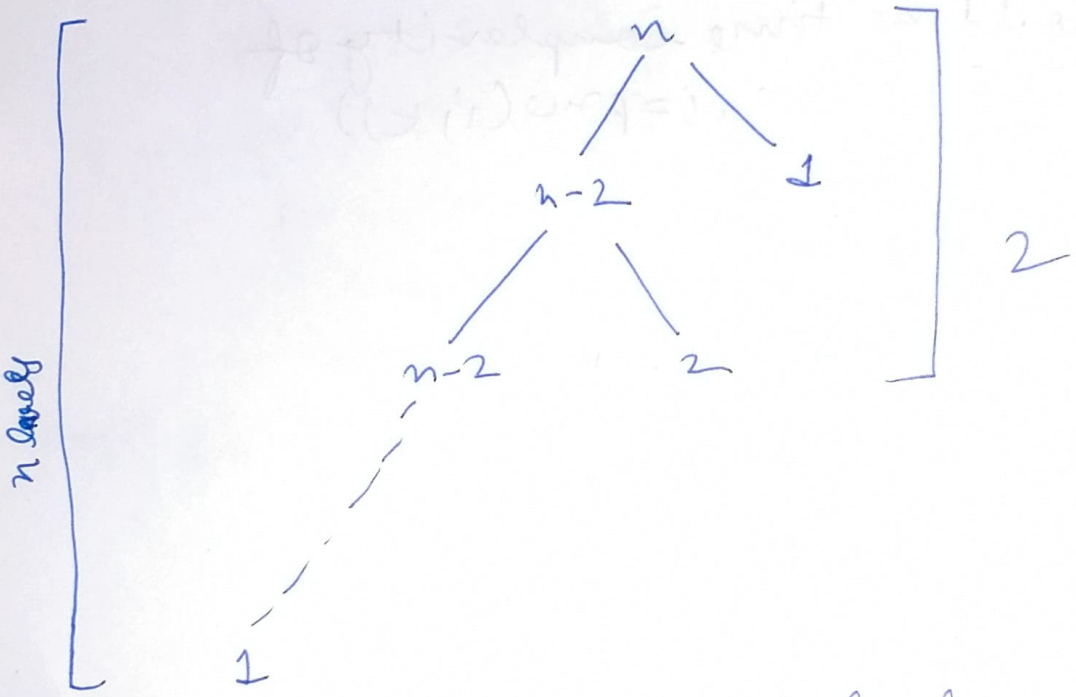
$1 + 1 + 1 + \cdots\cdots$ m times

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

Q.7 Write a recurrence relation when quick sort repeatedly divides array into 2 parts of 99% and 1%. Derive time complexity in this case. Show the recurrence time while deriving time complexity. Find difference in heights of both extreme parts.

Ans: Given algorithm divides, array in 99% & 1% part.

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



'n' work is done at each level.

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

$= n \times n$

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

Lowest height = 2
Highest height = n

∴ Difference = n − 2 $\quad n > 1$.

The given algorithm produces linear result.

8. Arrange the following in increasing order of rate of growth:

a) n, n!, $\log n$, $\log(\log n)$, root(n), $n \log n$, $\log^{2n}$, 2^n, 2^{2^n}, 4^n, n^2, 100

Ans a) $100 < \log(\log n) < \log n < (\log n)^2 < \sqrt{n} < n < n \log n < \log(n!) < n^2 < 2^n < 4^n < 2^{2^n}$.

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

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

c) 8^{2n}, $\log_2(n)$, $n \log(n)$, $n \log_2(n)$, $\log(n!)$, n!, $\log_8(n)$, 96, 8^{2n}, $7n^3$, $5n$

Ans c) $96 < \log_8 n < \log 2n < 5n < n \log_2(n) < n \log(n)$ $\log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$