Tutorial - 2

Name:- Dalesh Goel
Section:- F
Roll no.:- 51
Univ. Roll no:- 2016713

1> what is time complexity of below code and how?

```
void func(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-level

for(i)

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

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

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

$m \approx \sqrt{n}$

By summation method

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

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

Dalesh.

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

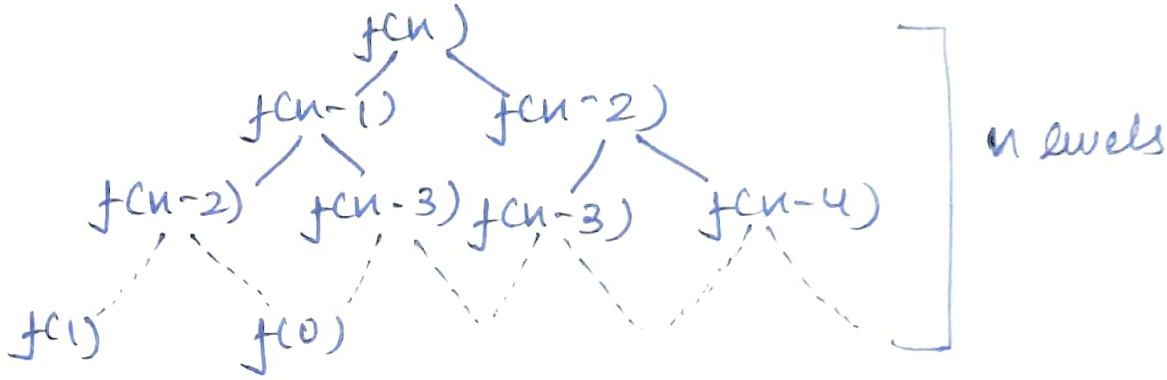
⇒ For Fibonacci series

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

$f(0) = 0$

$f(1) = 1$

By forming a tree



∴ At every function call we get 2 function calls

∴ for n levels

we have = 2 × 2 ... n times

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

MAXIMUM SPACE

considering Recursive

Stack: no. of calls maximum = n

For each call we have space complexity O(1)

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

without considering Recursive stack:

each call we have time complexity O(1)

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

Daksh.

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

(i) $n(\log n)$ → Quick sort

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

(ii) n^3 → Multiplication of 2 square Matrix

```
for(i=0; i<n1; i++)
{
    for (j=0; j<c2 ;j++)
        for(k=0; k<c1; k++)
          res[i][j] += a[i][k] * b[k][j];
}
```

Thank U.

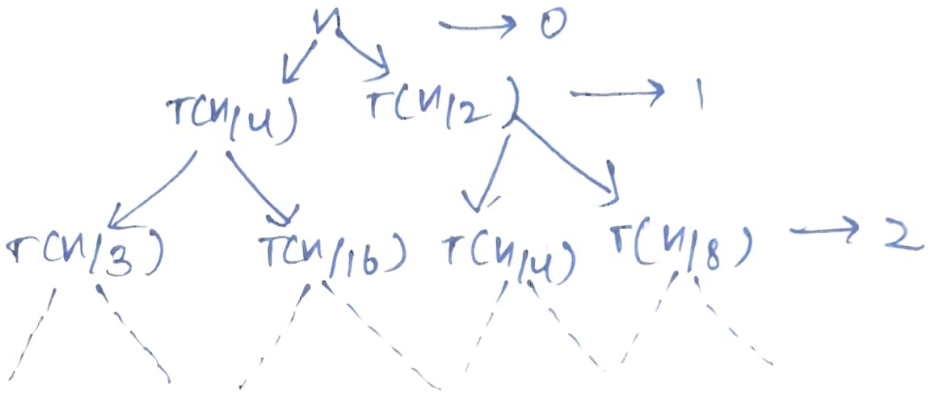
(iii) log(log(n))

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

4.) solve the following recurrence relation

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

⇒



At level

$0 \rightarrow cn^2$

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

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

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

$= k = \log_2 n$

$T(n) = c\left(n^2 + \left(\frac{5}{16}\right)n^2 + \left(\frac{5}{16}\right)^2 n^2 + \cdots + \left(\frac{5}{16}\right)^{\log n} n^2\right)$

$T(n) = cn^2\left[1 + \left(\frac{5}{16}\right) + \left(\frac{5}{16}\right)^2 + \cdots + \left(\frac{5}{16}\right)^{\log n}\right)$

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

Takes n.

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

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

$\boxed{O(cn^2)}$

5) what is complexity of fun()? (5)

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

⇒ for

i	j
1	1
2	1+3+5
3	1+4+7
⋮	
n	1+5+9

$j = (n-1)/i$ times

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

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

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

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

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

6) what should be complexity of

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

⇒ for

i
2^1
2^k
2^{k^2}
2^{k^3}
⋮
2^{k^m}

where

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

$k^m = \log_2 n$

$m = \log_k \log_2 n$

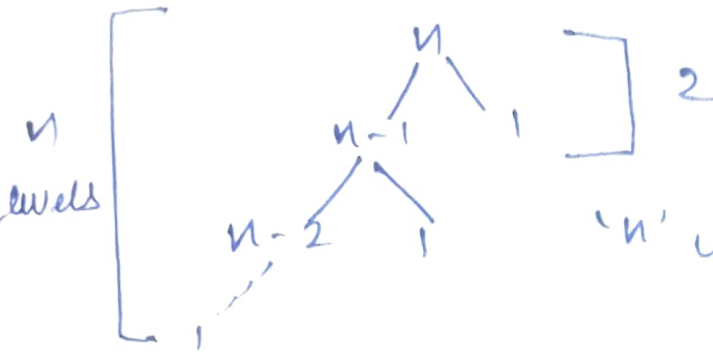
Taken.

$\because \sum_{n=1}^{m} 1$ $\quad 1+1+1 \cdots$ m times

$$\boxed{T(n) = O(\log_k \log n)}$$

7) Given algorithm divides array in 99% and 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 = n

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

The given algorithm produces linear result

8)

(a) $100 < \log\log n < \log n < (\log n)^2 < \sqrt{n} < n < n\log n < \log(n!_0) < n^2 < 2^n < 4^n < 2^{2^n}$

(b) $1 < \log\log n < \sqrt{\log n} < \log n < \log 2n < 2\log n < n < n\log n < 2n < 4n < \log(n!_0) < n^2 < n!_0 < 2^{2^n}$

(c) $ab < \log_3 n < \log 2n < 5n < n\log_{10}(n) < n\log_2 n < \log(n!_0) < 8n^2 < 7n^3 < n!_0 < 8^{2n}$

Daksh