Ajay Pratap Kardari
Section D
Roll No : 62

Tutorial :-02

① void fun (int n)
    Σ

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

j	i
1	1
2	3
3	6
4	10
⋮	
k	$1+2+3+\cdots k = \dfrac{k(k+1)}{2}$

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

or $\quad k^2 < n$

$$k < \sqrt{n}$$

TC: $\underline{O(\sqrt{n})}$.

2i   Fibbonacci Series with recursion

```
int fibbo (int n )
{
        if (n==1)
            return 1;
        else if (n == 0)
            return o;
        else
            return fibbo (n-1) + fibbo(n-2);
}
```


$2^0 = 1$

$2^1 = 1$

$2^2 = 4$

$\vdots$

0

$\underline{2^n}$

∴   T.C of function is $O(2^n)$

Space Complexity $O(1)$

∵ no extra spau

Using Recursion function

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

for $n = 0$
$T(0) = 1$
for $n = 1$
$T(1) = K$ $\Bigg\}$ ∵ only one condition no recursive calls.

3→ Program for complexity $(n \log n)$, (n^3), $\log (\log n)$.

① $n (\log n)$
Merge sort.

② (n^3)
Printing elements of 3-D array.

③ $(\log (\log (n)))$
pow function inside some loop.

3-) i→ Merge Sort   n log n

ii→ n³.

```
for (i=0 ; i<n ; i++)
{   for(j=0 ; j<n ; j++)
    {  for(k=0 ; k<n ; k++)
       {
          arr[i][j][k] = i+j+k;
       }
    }
}
```

iii -)   for log (log n)

```
for (int i=2 ; i<n ; i=pow(i,k))
{
      (100)
}
          k ∈ constant.
```

iv)   for   n logn
```
int abcd (int n)
{
    for (int i =1 ; i<=n ; i++)
    {
        for (int j=1 ; j<=n ; j++)
        { ε ||O(1) }
    }
```

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

for $n = 0$

$T(0) = 2T(0) + 0$

$T(0) = 0$

$\therefore$

Clearly $\frac{n}{2} > \frac{n}{4}$

$\therefore$

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

Using Master Theorem

$f(n) = cn^k$

$k = 2$

$a = 2 \qquad b = 2$

$\therefore \quad a < b^k$

$2 < 2^2$

$2 < 4$

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

$= n^2$

5)
```
int fun (int n)
{
    for (int i=1; i<=n; i++)
        for (int j=1; j<m; j+=i)
        {
            //O(1) Task)
        }
}
```

i	j (inner loop)
1	n
2	n/2
3	n/3
4	n/4
k	$\frac{n}{k}$
n	1

$$\text{total} = n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots 1$$

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

$$= n + \sum_{k=1}^{n} \left(\frac{1}{k}\right)$$

$$= n * \log(n)$$

$$TC = O(n \log n).$$

6→ -for (int i=2 ; i<=n; i = pow (i,k))
i
|| O(1)   k ∈ constant
}

i can take values as i^k
for iteration 1st $i = 2^\pi$
              2nd $i = 2^k$
              3rd $i = (2^k)^k = 2^{k^2}$
              4th $i = (2^k)^k = 2^{k^3}$
              :
              :

              ith $i = 2^{k^{i-1}}$

, Loop ends for $2^{(k)^{i-1}} <= n$
        or $2^{(k)^i} <= n$
              $k^i \log 2 <= n$
              $k^i \quad <= \dfrac{\log n}{\log 2}$
              $k^i <= \log_2 n$

              $i \log k <= \log(\log n)$
              $i \log k <= \log (\log n)$

              $i \leq \dfrac{\log (\log n)}{\log k}$

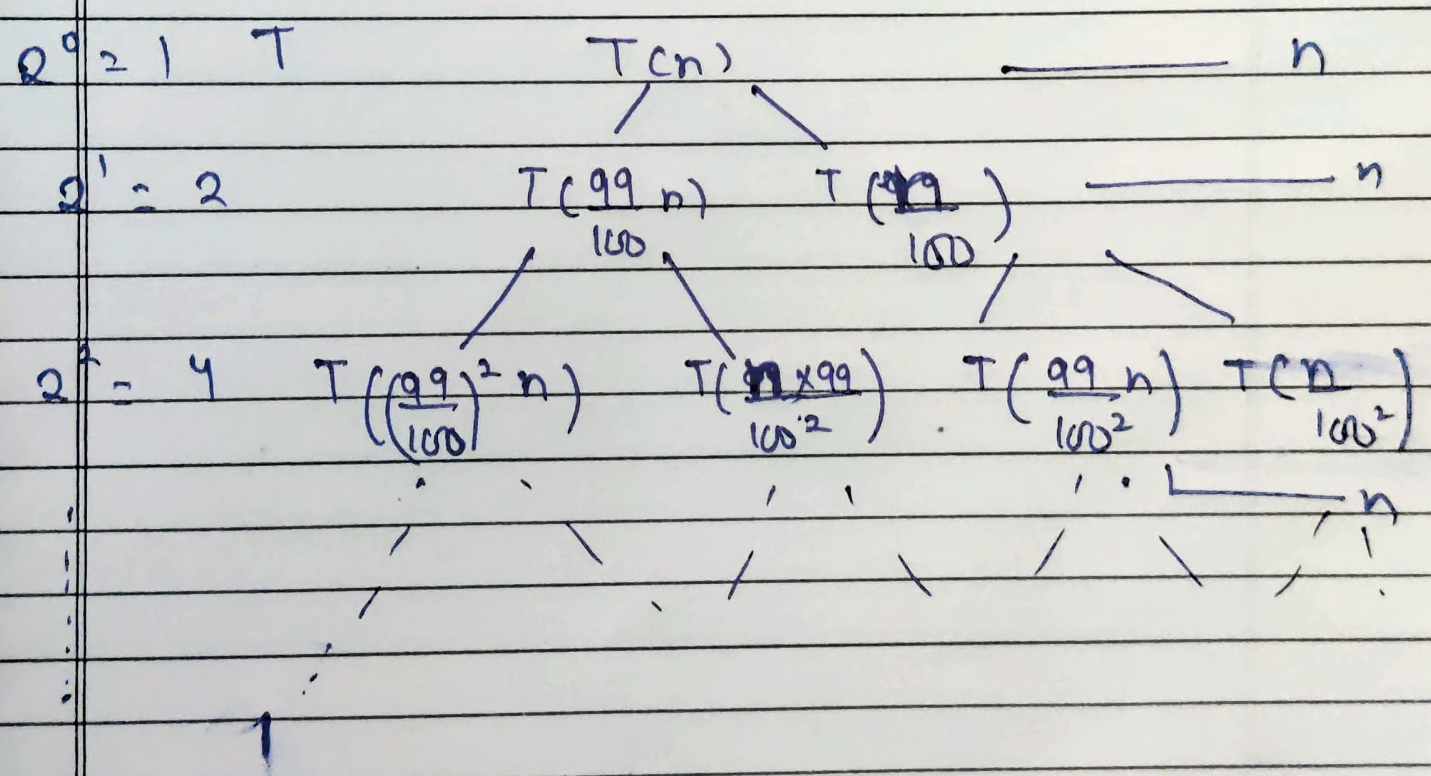 $\boxed{i \leq \log (\log n)}$

6→ $T \leq \log(\log n)$

(7) Quick Sort analysis for 99:1 partition

Possibly when pivot is either last from front or end always

Partition function

so $T(n) = T(\frac{99}{100} n) + T(\frac{n}{100}) + O(n)$

Recursive calls for division $\frac{99}{100}$ division.

$2^0 = 1$ T $T(n)$ ⎯⎯⎯ n

$2^1 = 2$ $T(\frac{99}{100} n)$ $T(\frac{n}{100})$ ⎯⎯⎯ n

$2^2 = 4$ $T((\frac{99}{100})^2 n)$ $T(\frac{n \times 99}{100^2})$ $T(\frac{99 n}{100^2})$ $T(\frac{n}{100^2})$ ⎯ n

1

so for finding the height of tree

$n \times (\frac{99}{100})^k = 1 \Rightarrow \frac{n}{(\frac{100}{99})^k} = 1$

$k = \log_{100/99} n$

$n = (\frac{100}{99})^k$

for n time

$TC = O(n \times \log_{100/99} n)$

$\log n = k \log \frac{100}{99}$

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

$100 < \log\log(n) \leq \log^2 n \quad \leftarrow \quad \log n < \log n!$

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