Date. _____
Page No. _____

Ajay Pratap Kardari
Section D
Roll No: 62

Tutorial :02

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

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

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

or $k^2 < n$

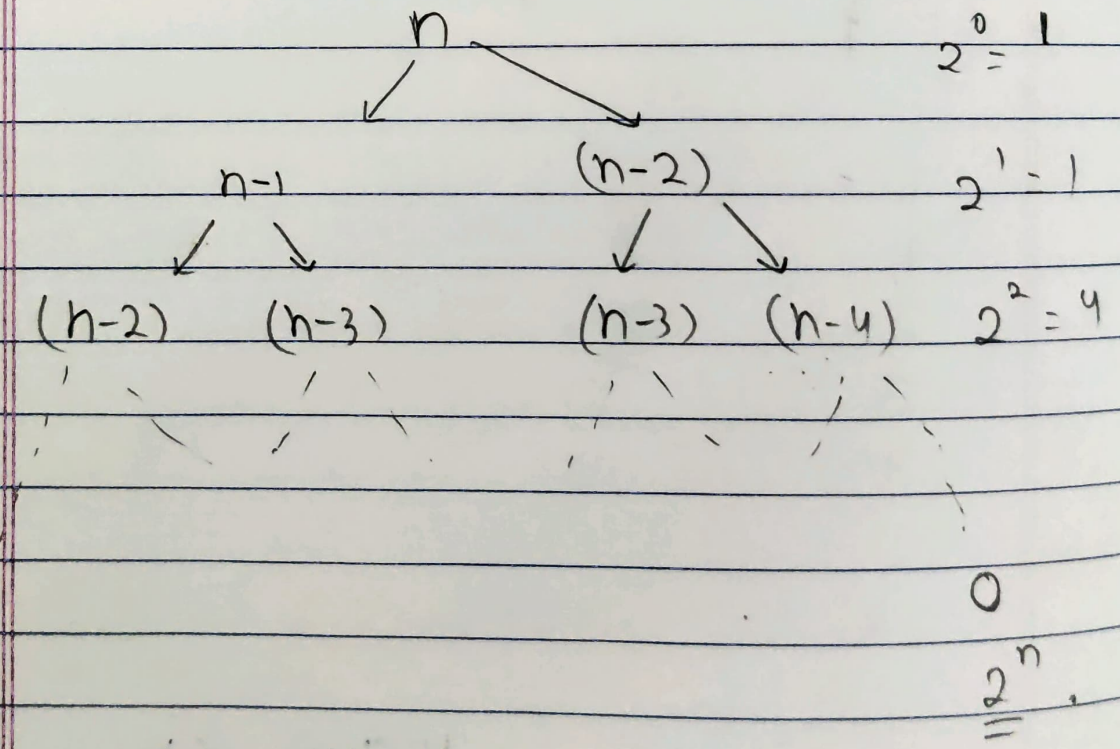
$k < \sqrt{n}$

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

Date.
Page No.

2. Fibbonacci Series with recursion

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



∴ T.C of function is $O(2^n)$

Space complexity $O(1)$

∵ no extra spac

Date. _____
Page No. _____

Using Recursion function

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

for $n = 0$

$T(0) = 1$

for $n = 1$

$T(1) = 1$

∵ only one condition has 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.

Date.
Page No.

3-) i) Merge Sort $n \log n$

ii) n^3.

```
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))
  {
    // O(1)
  }
```

k ∈ constant.

iv) for $n \log n$

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

Date.
Page No.

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

for $n = 0$

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

$T(0) = 0$

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

$\therefore$

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

Using Master Theorem

$f(n) = cn^k$

$k = 2$

$a = 2 \quad b = 2$

$\because a < b^k$

$2 < 2^2$

$2 < 4$

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

$= n^2$

Data. _____
Page No. _____

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

i	j (inner loop)
1	n
2	n/2
3	n/3
4	n/4
:	:
k	n/k
:	:
n	1

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 = $O(n \log n)$.

Date. _____
Page No. _____

6) for (int i=2 ; i<=n; i = pow(i,k))
{
// O(1) k ∈ constant
}

i can take values as i^k

for iteration 1st $i = 2^1$
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 $2^{(k)^i} <= n$
$k^i \log 2 <= n$
$k^i <= \frac{\log n}{\log 2}$
$k^i <= \log_2 n$

$i \log k <= \log(\log_2 n)$
$i \log k <= \log(\log n)$

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

$i \leq \log(\log n)$

TC = $\log(\log n)$

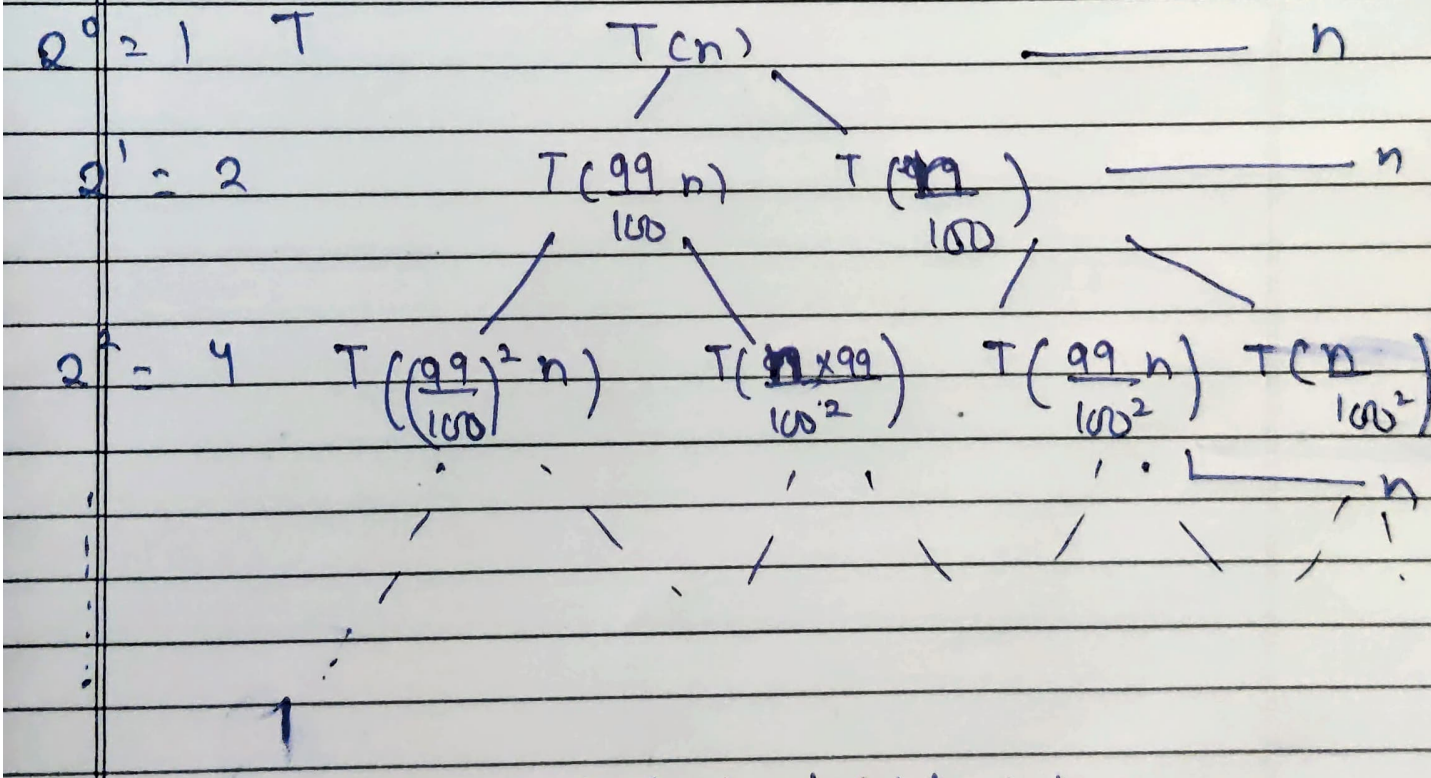
(7) Quick Sort analysis for 99:1 partition

Possibly when pivot is either start from front or end always

Partition function

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

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



So for finding the height of tree

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

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

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

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

for n time

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