Tutorial-2

① - Lakshay Sharma
- AI and OS
- 10

1>

i	j
0	1
1	2
3	3
6	4
10	5

no. of time loop is running by k

$S_k = 1 + 3 + 6 + 10 - - - + T_k$

$S_{k-1} = 1 + 3 + 6 + - - - - + T_{k+1}$

Subtracting both.

$S_k - S_{k-1} = 1 + 2 + 3 + 4 - - - + (k-1)$

$T_k = \frac{(k-1)\,k}{2}$

given that k^{th} term is n

$T_k = n$

$\frac{(k+1)\,k}{2} = n \Rightarrow \frac{k^2}{2} - \frac{k}{2} = n.$

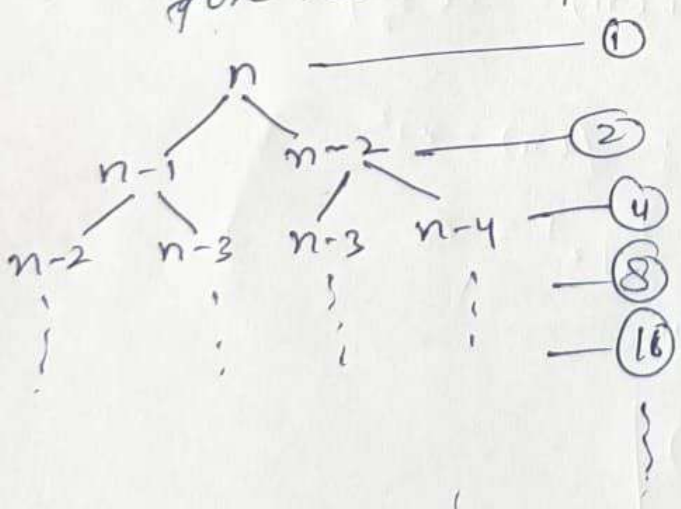
$k^2 = n$

$k = \sqrt{n}$

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

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

for recursive fibonacci solution



no. of times function is running will be sum of the series.

$S = 1 + 2 + 4 + - - - - + 2^n$

$= \frac{2^{n+1} - 1}{2 - 1} = 2^{n+1} - 1$

Time complexity → $\boxed{O(2^n)}$

← from recursion tree

height of the tree = n

$\boxed{\text{Space complexity} = O(n)}$

3) code having time complexity

O(nlogn):
```
for (int i=1; i<=n; i++)
    { for (int j=1; j<n; j=j*2)
        printf ("Hello");
    }
```

O(n^3):
```
for (int i=0; i<n; i++)
    { for (int j=0; j<n; j++)
        { for (int k=0; k<n; k++)
            printf("Hello");
        }
    }
```

O(log(logn)):
```
for (int i=2; i<=n; i=pow(i,3))
    { printf ("hello"); }
```

here n can be any positive integer.

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

Ignoring lower terms :- $T(n) = T(\frac{n}{2}) + cn^2$

using master's theorem :-

$a = 1 ; b = 2 ; f(n) = n^2$

$c = \log_b a = \log_2 1 = 0$

$0 < n^2$ True

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

5)

i	j
1	n
2	n/2
3	n/3
4	n/4

Time complexity will be sum of series

$S = \frac{n}{1} + \frac{n}{2} + \frac{n}{3} \cdots$

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

Complexity $\longrightarrow n \times \sum_{i=1}^{n} \left(\frac{n}{i}\right)$

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

6) Sequencing :- $2, 2^k, (2^k)^k \cdots$

Generalising :- $2^{k^0}, 2^{k^1}, 2^{k^2}, \cdots 2^{k^{\lambda-1}}$

Assumption :- Let no. of terms be λ

given :- last term is n

$2^{k^{\lambda-1}} = n$

$k^{\lambda-1} \log 2 = \log n$

$k^{\lambda-1} = \log n$ [Ignoring constant $(\log 2)$]

$(\lambda-1) \log k = \log n$

$\boxed{\lambda = \log(\log n)}$

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

8) (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) $1 < \log(\log n) < \sqrt{\log n} < \log n < \log 2n < 2 \log n < n$
$< n \log n < 2n * 4n < \log(n!) < n^2 < n! < 2^{2^n}$

(c) $96 < \log_8 n < \log 2n < 5n < n(\log_6 n) < n(\log_2 n)$
$< \log(n!) < 8n^2 < 7n^3 < n! < 8^{2n}$