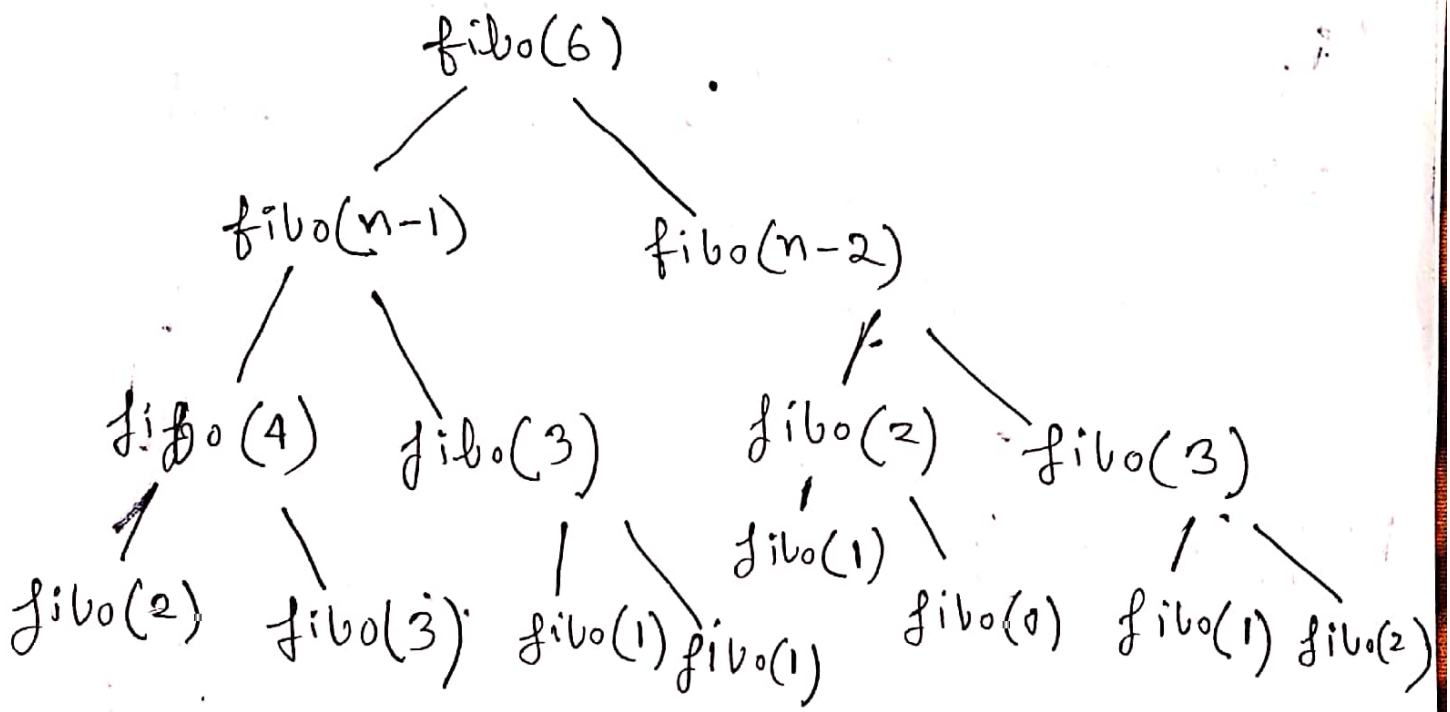
2030166

Problem - 2 :-

Implementation - 1 :

To find the time complexity we have find how many time the loop will run.

Example for 6th fibonacci number.



6th fibonacci is = $2^0 + 2^1 + 2^2 + 2^3$

n-th " $n = 2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^n$

$\therefore$ time complexity $= 2^{n+1} - 1$

$= 2^n$

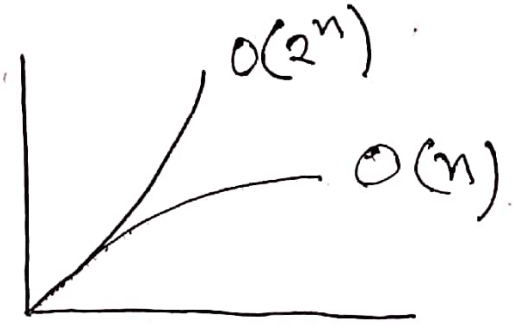
Implemention - 2 : For constant time like if condition, single line code the time complexity is O(1) which is upper ~~bond~~ bound.

But 'for' loop time complexity for nth time is O(n). As, both 'if' and 'for' loop is present the time complexity is

$$O(1) + O(n)$$

$$= O(n)$$

Difference :



Now, we can see $O(2^n)$ is has most worst time than $O(n)$.

$O(2^n) = 2^{10} = 1024$ s

$O(n) = 10 = 10$ s

Problem 4 : To know the time complexity

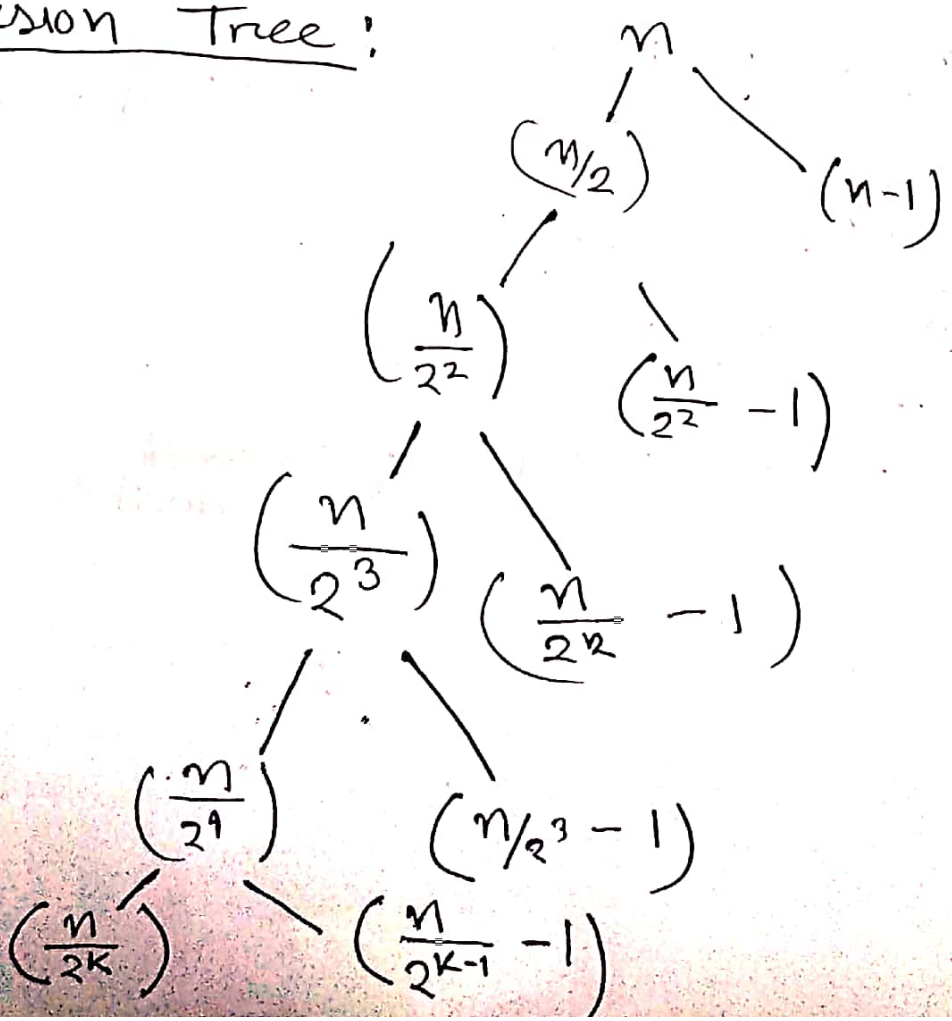
We have to know how many time the loop will run. From problem 4 We see there are 3 'for' loop and each loop will run 'n' times.

So, time complexity = $n \times n \times n$
$= n^3$

Problem 5:

1) $T(n) = T(n/2) + (n-1)$, $T(1) = 0$

Recursion Tree:



Now,

$$T(n) = (n-1) + \left(\frac{n}{2}-1\right) + \left(\frac{n}{2^2}-1\right) + \left(\frac{n}{2^3}-1\right) + \left(\frac{n}{2^4}-1\right) + \cdots$$

$$\Rightarrow \left(n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \cdots\right)$$

Here,

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

$$\Rightarrow 2^k = n$$

$$k = \log n$$

$$= n\left(1 + 1/2 + 1/2^2 + 1/2^3 + \cdots\right) - \log n$$

Again,

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

$$T(n) = \theta(n)$$

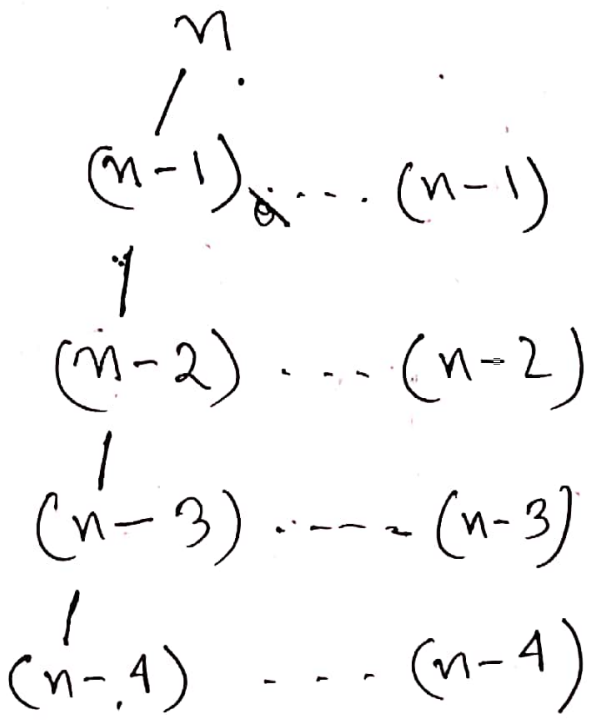
Also, $T(n) = O(n)$

∴ The time complexity in ~~worst~~ Worst case = $O(n)$

Problem 5

2] $T(n) = T(n-1) + (n-1)$, $T(1) = 0$

Recursion tree,



$$T(n) = (n-1) + (n-2) + (n-3) + (n-4) + \ldots + (n-(n-1))$$

$$= (n+n+n+\ldots(n-1)) - (1+2+3+\ldots+(n-1))$$

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

W.K

Series $= 1+2+3+\ldots+n = \frac{n(n+1)}{2}$

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

$$= \frac{n^2}{2} - \frac{n}{2} = \frac{1}{2}(n^2-n)$$

$$= \Theta(n^2) \,/\, O(n^2)$$

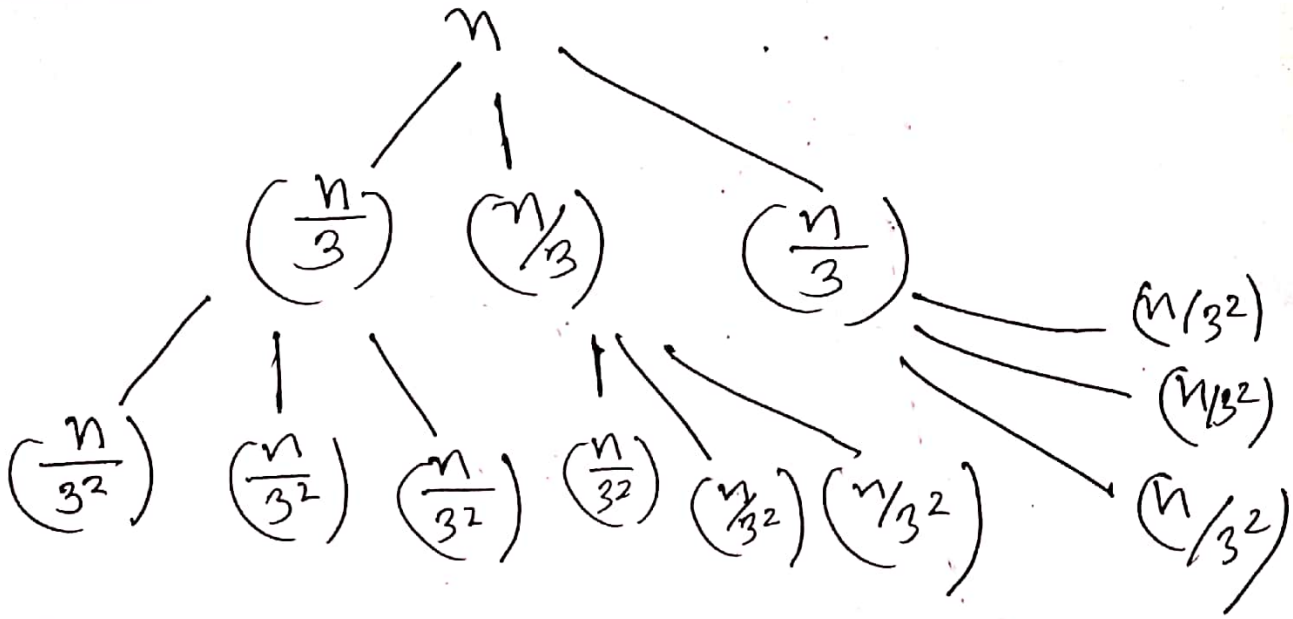
$T(n) = \Theta(n^2)$

∴ Time complexity in worst case $= O(n^2)$

3| $T(n) = T(\frac{n}{3}) + 2T(\frac{n}{3}) + n$

$= 3T(\frac{n}{3}) + n$

Recursion tree:



Assume,

$$\frac{n}{3^k} = 1$$

$\Rightarrow 3^k = n$

$k = \log_3 n$

$T(n) = (n + n + n + n + \dots \log n)$

$= n \log n$

$= \Theta(n \log n) \;/\; O(n \log n)$

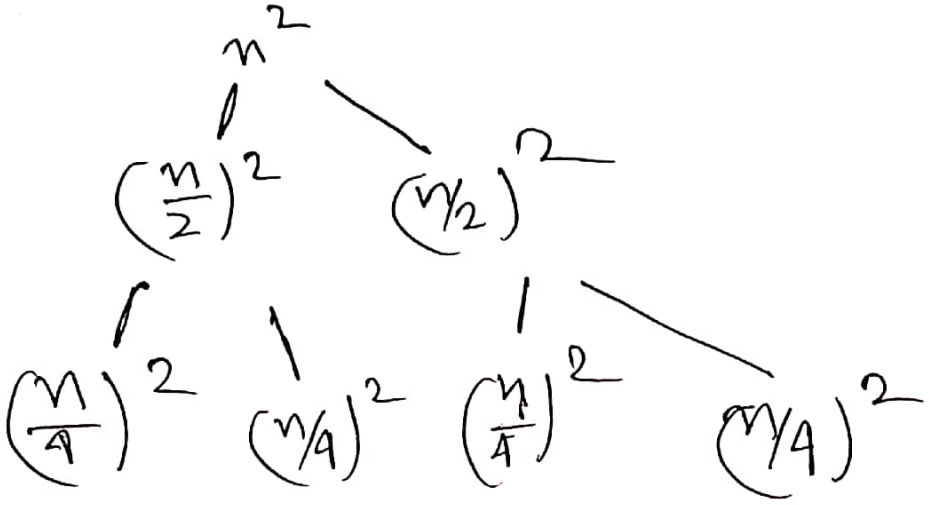
∴ The worst-case time complexity

$= \Theta(n \log n)$

Problem 5

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

Recursion tree,



Now, Using Master theorem,

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

Here, $\log_2^2 = 1$, $K = 2$,

W.K, $\log_2^2 = 1 < K = 2$

For case(3) the time complexity will be $O(n^2)$

∴ The worst case complexity will be n^2.

[Proved]