Recursive Tree Method

This method is used to solve the recurrence relation with multiple recursive calls (more than 1).

~~The~~ Same recurrence relation can be solved using substitution but for multiple recursive relations many mathematical calculations are required so we will get the result slower.

Recursive Tree Method

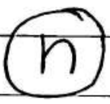
Example 1

Given recurrence relation is

$$T(n) = T(n/2) + T(n/2) + \underline{n}$$

Step 1:- Now other than recursive calls whatever is left, assign that as a root of the tree.

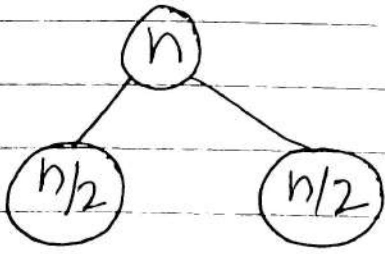
In our case we have 2 recursive calls $T(n/2)$ & $T(n/2)$ & what is remaining is n. So n becomes root of the tree.



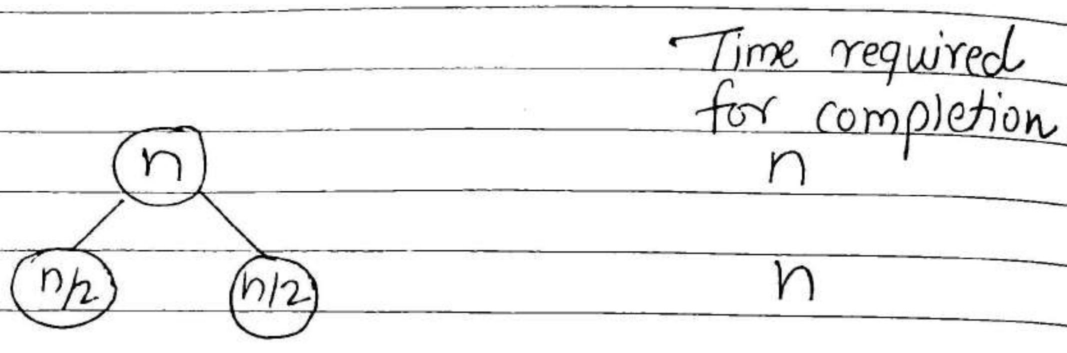
Step 2:- Now check how many recursive calls we have, we have 2 recursive calls so root node n will have 2 child nodes.

The smaller recursive call goes on left side of root node & larger recursive call goes to the right side.

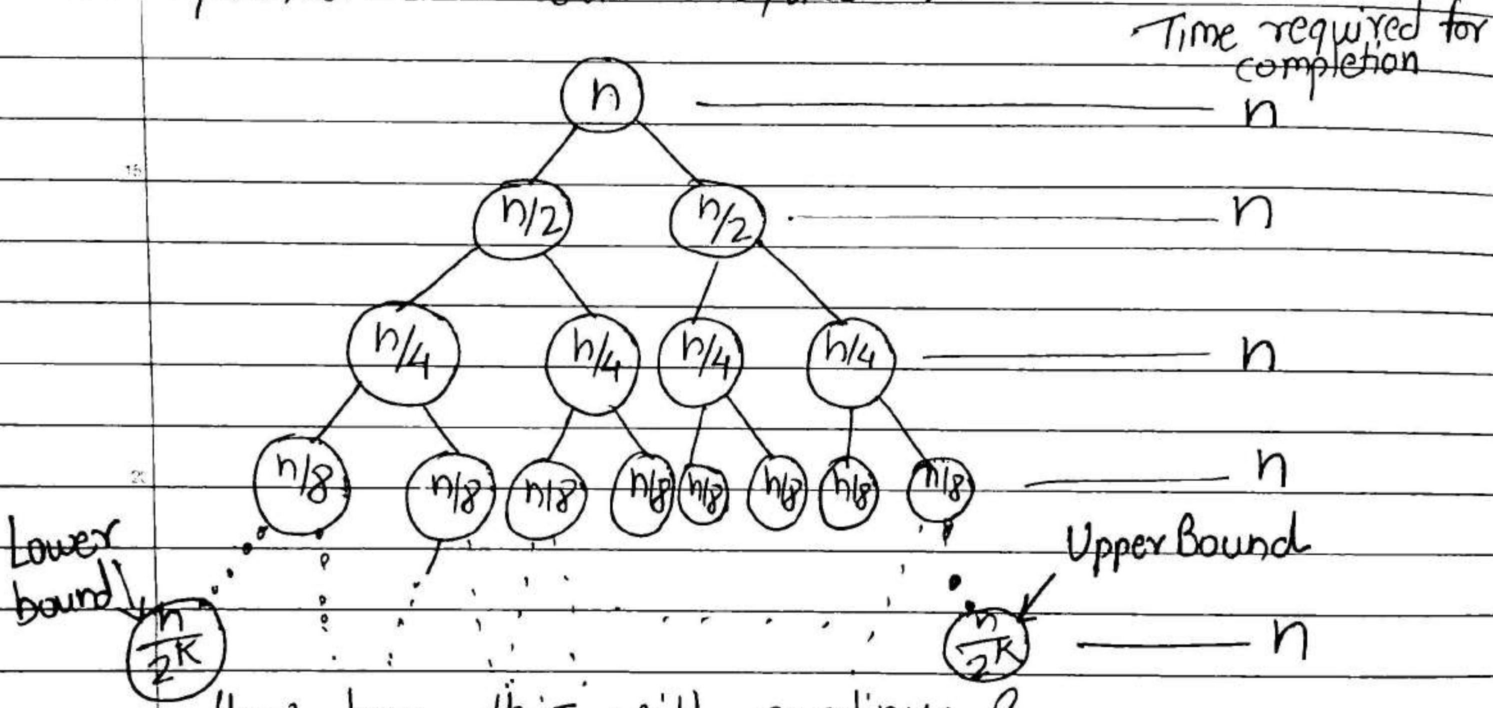
In our case both recursive calls are n/2



Step3:- ~~Compute the~~ calculate the computation time for each level of tree



Step4:- Go on substituting till we identify the pattern of our equation.



How long this will continue ?
This will continue till n becomes 1
i.e. till the recursive function terminates.

Step.5:- Total no of steps are $\boxed{k}$
Time required for completion of each step is $\boxed{n}$

So time required to solve the given recurrence relation = $n * k$

step6:- Let's find out value of k

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

So, $n = 2^k$

Taking log both sides

$\log n = k \cdot \log 2$ ~~As to~~

$\boxed{k = \log n}$ as $\log 2 = 1$

Step 7 :-
The time complexity of the given recurrence relation is
$n * k$
$n * \log n$

which is $\theta(n \log n)$

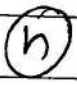
$\boxed{T(n) = \Theta(n \cdot \log n)}$

Example 2

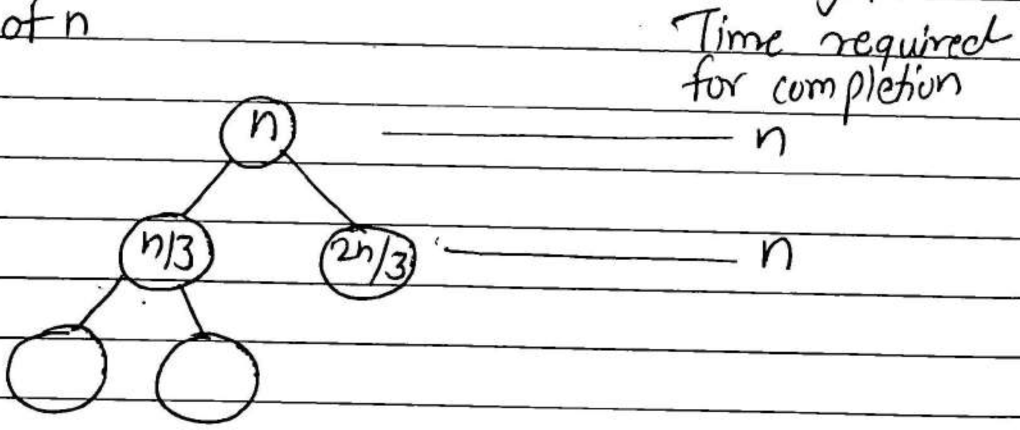
Given recurrence relation is
$T(n) = T(n/3) + T(2n/3) + n$
Solve this equation using recursive tree method.

$T(n) = T(n/3) + T(2n/3) + n$

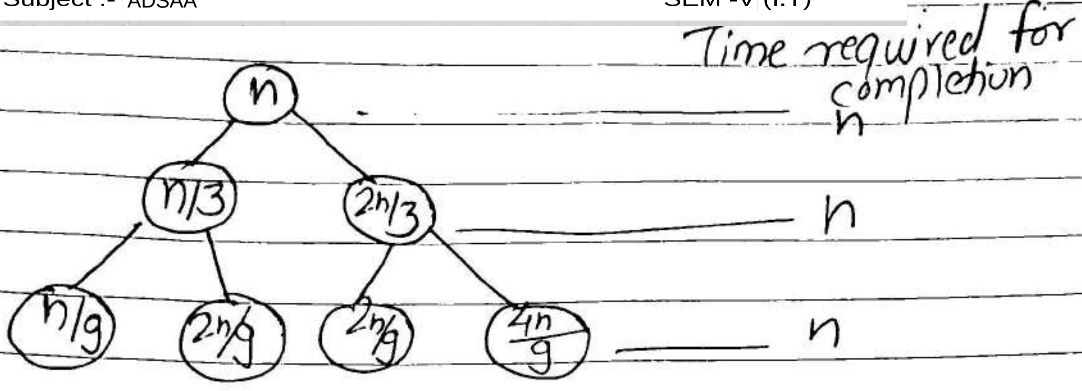
Step 1:- Other than recursive terms we have n so n is root node of tree



Step 2:- We have 2 recursive calls n/3 & 2n/3 so n/3 will be left child of n as it is smaller. And 2n/3 is a right child of n

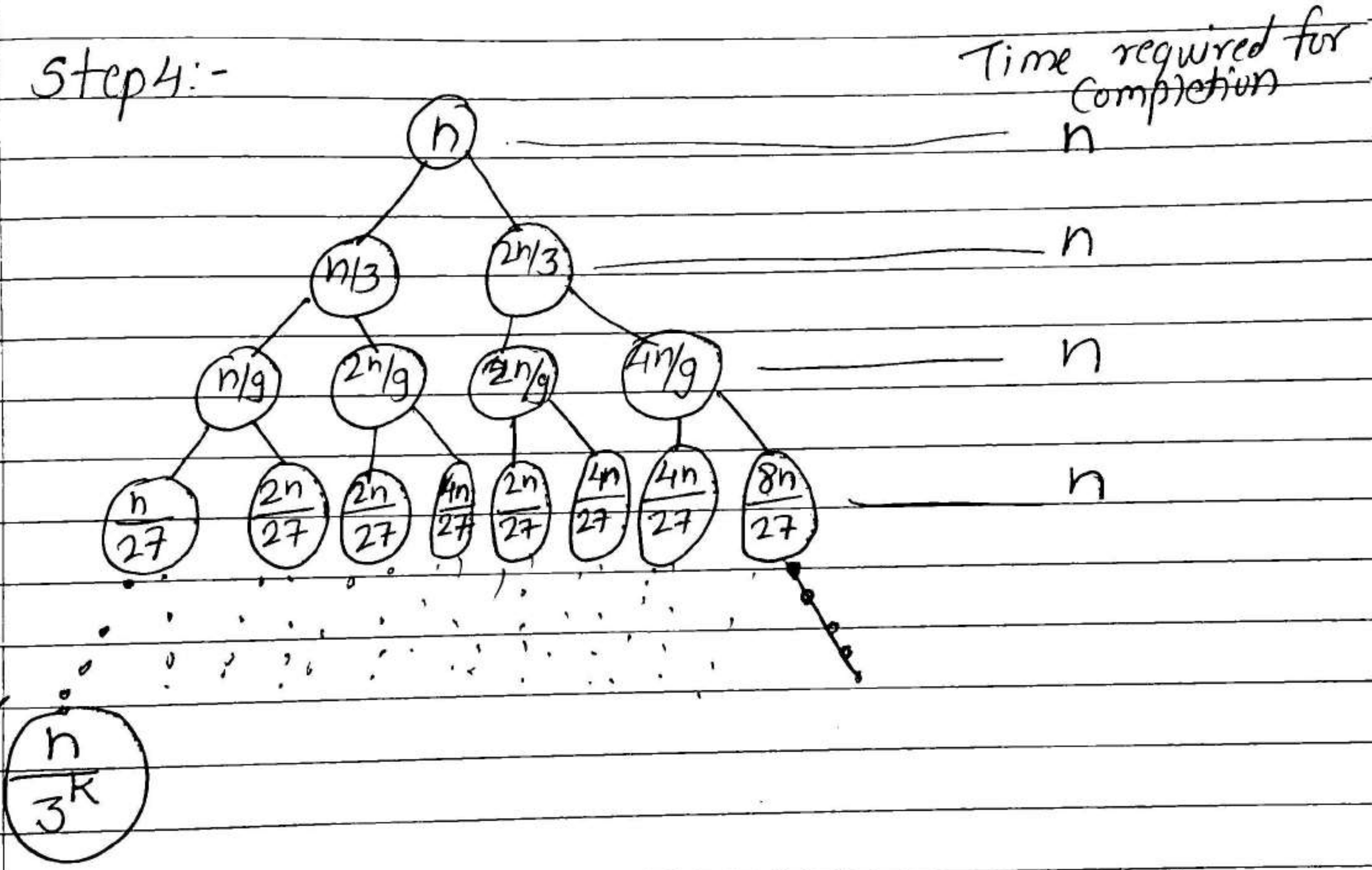


Step 3:- Go on substituting till we find a pattern of our equation.



~~Pleas~~ Remember left child of every node will always be in the terms of n/3 & right child of every node will always be in terms of 2n/3 for both left subtree and right subtree

Step 4:-



We are not suppose to find the time for intermediate nodes we just have to find the lower bound to give us minimum time and upper bound to find out maximum time of execution

Here lower bound which we have got is

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

Now let's find out upper bound which is $\frac{2n}{3}$

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

So for first step can we say

$$\frac{n}{\left(\frac{3}{2}\right)^0} = 0$$

For Second step we have $\frac{2n}{3}$ so we can say

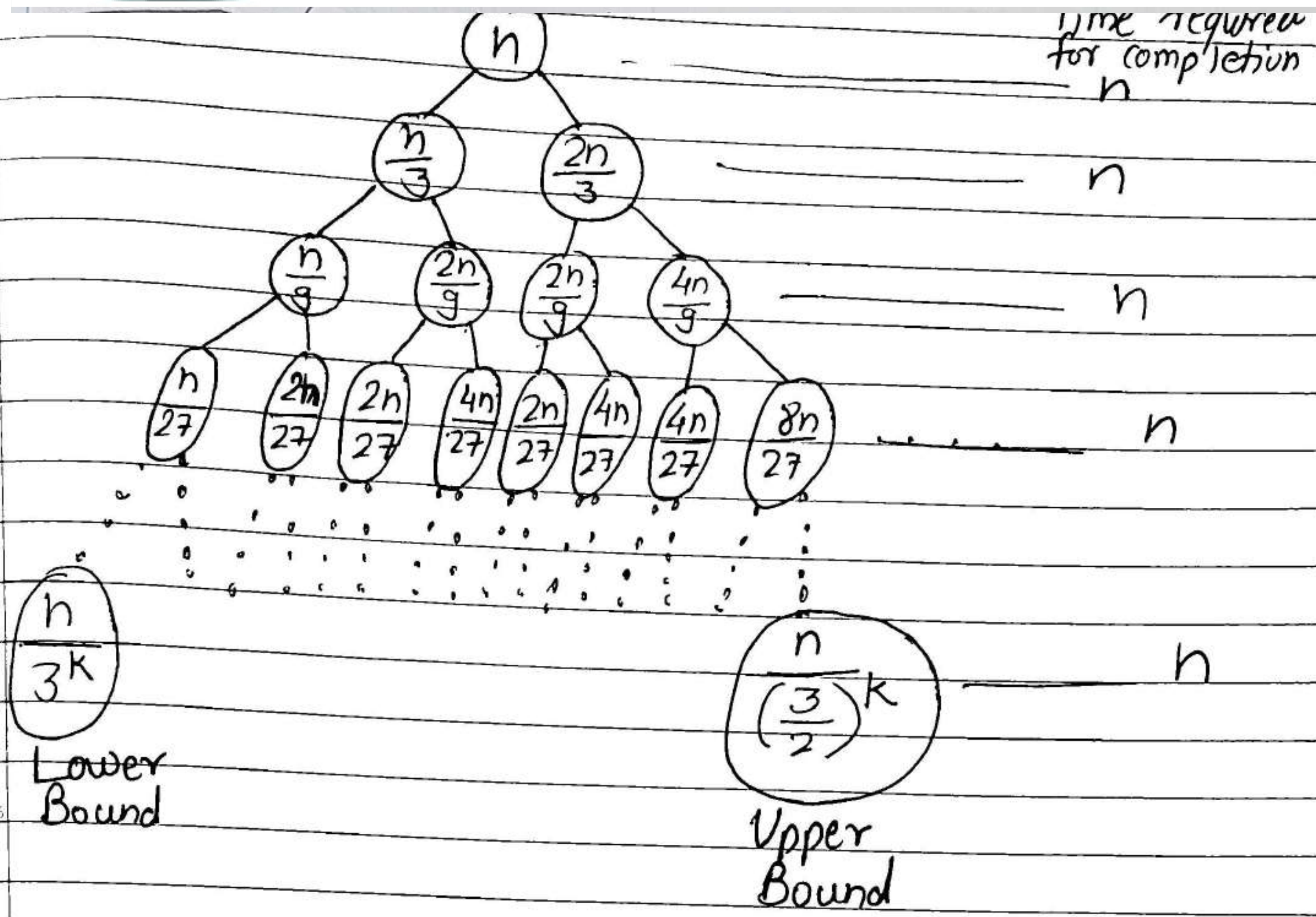
$$\frac{n}{\left(\frac{3}{2}\right)^1} = \frac{2n}{3}$$

For third step we have $\frac{4n}{9}$ so we can say

$$\frac{n}{\left(\frac{9}{4}\right)} = \frac{4n}{9}$$

$$\frac{n}{\left(\frac{3}{2}\right)^2} = \frac{4n}{9}$$

So for k^{th} step we will have $\frac{n}{\left(\frac{3}{2}\right)^k}$ as upper bound.



This tree continues till T(n) becomes 1
i.e. function terminates.

Step 5:- Total no. of steps are $\boxed{k}$
Time required for completion ~~is~~ of each step is $\boxed{n}$

~~So time required to solve the re~~

Let's find height of left subtree,
Assume

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

So $n = 3^k$

Taking log at both sides gives us

$$\log n = k \cdot \log 3$$

$$\boxed{k = \log_3 n}$$ Height of left subtree

Now let's find height of right subtree

Assume $\frac{n}{\left(\frac{3}{2}\right)^k} = 1$

So, $n = \left(\frac{3}{2}\right)^k$

Taking log at both sides

$$\log n = k \log\left(\frac{3}{2}\right)$$

$$\log_{\left(\frac{3}{2}\right)} n = k$$

$$\boxed{k = \log_{\frac{3}{2}} n}$$ Height of right subtree

⊛ $\log_{\frac{3}{2}} n$ this value is always bigger than $\log_3 n$

We know that total time required to solve the given recurrence relation is

$n * k$

For worst case the time required is

$T(n) = \boxed{n * \log_{\frac{3}{2}} n}$ This is max^m time required for execution.

For best case the time required is

$\boxed{T(n) = n * \log_3 n}$ This is the min^m time required for execution

$$\boxed{\begin{gathered} T(n) = O\left(n * \log_{\frac{3}{2}} n\right) \\ T(n) = \Omega\left(n \log_3 n\right) \end{gathered}}$$