The Recursion Tree Method

- In this method, recursion tree is constructed.
- Using the tree, recurrence relation is solved.

- In the tree, each node represents the cost of single subproblem somewhere in the set of recursive function invocation.

- The method is effectively used in divide and conquer strategy.

- It is used to generate good guess.
- After solution is found by recursion tree method, it is verified by substitution method.

Examples Steps:

1. Draw the Recursion Tree
2. Cost of each level
3. Find the depth of tree
4. Find the number of leaves (No. of nodes at last level)
5. Find Total cost last level.
6. Total cost for entire tree.

3 Cases:

1. Cost of Root node is Maximum
2. Cost of Leaf node is Maximum
3. Cost of Each ~~node~~ level is Same.

→ All leaf nodes are at same level

Problem on Case1

Ex: $T(n) = 3T(n/4) + \theta(n^2)$

$\therefore T(n) = 3T\left(\frac{n}{4}\right) + cn^2$

The above statement says that the complex problem of size 'n' is divided into three subproblems of size 'n/4' and the operations cost is cn^2.

Step 1: Recursion Tree

Recursive call	Number of Nodes	Tree	Row Sum
$T(n)$	1	cn^2	cn^2
$T(n/4)$	3	$\frac{cn^2}{16}$ $\frac{cn^2}{16}$ $\frac{cn^2}{16}$	$\frac{3}{16}cn^2$
$T(n/16)$ $=T(n/4^2)$	3^2	$c\left(\frac{n}{16}\right)^2$ ...	$\left(\frac{3}{16}\right)^2 cn^2$
$T\left(n/4^i\right)$	3^i	$c\left(\frac{n}{16^i}\right)^2$	$\left(\frac{3}{16}\right)^i cn^2$

Step 2: Cost of each level $= \left(\frac{3}{16}\right)^i cn^2$

Step 3: Depth of tree i.e last level of tree, where

$$\frac{n}{4^i} = 1$$

$$n = 4^i$$

$$\log_4 n = \log_4 4^i$$

$$\log_4 n = i$$

$$\boxed{i = \log_4 n}$$

Step 4: Number of leaves i.e No. of nodes in the last level

$$= 3^i$$

$$= 3^{\log_4 n}$$

$$= n^{\log_4 3} \qquad = n^{0.79248}$$

Step 5: Total cost at last level

$$= n^{\log_4 3} * T(1)$$

$$= \Theta(n^{\log_4 3})$$

Step 6: Total cost of entire tree,

$$T(n) = cn^2 + \frac{3}{16} cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \Theta(n^{\log_4 3})$$

$$T(n) = \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \quad \text{--- (I)}$$

Approximate equation (I),

$$T(n) < \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

$$= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3})$$

$$\because \left[\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}\right]$$

$$T(n) = \frac{16}{13} cn^2 + \Theta(n^{\log_4 3})$$

First term has higher growth on comparison with second term.

$$\therefore \boxed{T(n) = \Theta(n^2)}$$

n

n
n/2
1
n/4
1

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

$$n = 2^i$$

$$\log_2 n = \log_2 2^i$$

$$= i$$

Problem on case 2

Ex. $T(n) = 4T(n/2) + \theta(n)$

$\therefore T(n) = 4T\left(\frac{n}{2}\right) + c(n)$

Step 1 : Recursion Tree

Recursive call	No. of nodes	Tree	Row sum
$T(n)$	1	cn	cn
$T(n/2)$	4	$\frac{cn}{2}$ $\frac{cn}{2}$ $\frac{cn}{2}$ $\frac{cn}{2}$	$2cn$
$T\left(\frac{n}{4}\right)$ $= T(n/2^2)$	4^2	$\frac{cn}{2^2}$ $\frac{cn}{2^2}$ $\frac{cn}{2^2}$ $\frac{cn}{2^2}$ $\frac{cn}{2^2}$ ...	$2^2 cn$
$T(n/2^i)$	4^i	$\frac{cn}{2^i}$	$2^i cn$

Step 2 : Cost of each level $= 2^i cn$

Step 3 : Depth of tree i.e last level of tree,

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

$$\log_2 n = \log_2 2^i$$

$$i = \log_2 n$$

Step 4 : No. of leaf nodes i.e No. of nodes at last level $= 4^i$

$$= 4^{\log_2 n}$$

$$= n^{\log_2 4}$$

$$= n^{\log_2 2^2}$$

$$= n^2$$

Step5 : Total cost at last level. $= n^2 \times T(1)$

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

Step6 : Total cost of entire tree,

$$T(n) = cn + 2cn + 2^2 cn + \cdots + 2^{\log_2 n - 1} cn + \Theta(n^2)$$

$$= \sum_{i=0}^{\log_2 n - 1} 2^i cn + \Theta(n^2) \quad \longrightarrow (I)$$

Approximating equation (I),

$$< \sum_{i=0}^{\infty} 2^i cn + \Theta(n^2)$$

$$= \frac{1}{1-2} cn + \Theta(n^2) \qquad \left[\because \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}\right]$$

$$= -cn + \Theta(n^2)$$

Second term has higher growth in comparison to first

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

OR

Step 6 : Total cost of entire tree,

$$T(n) = cn + 2cn + 2^2 cn + \cdots + 2^{\log_2 n} cn$$

$$= \sum_{i=0}^{\log_2 n} 2^i cn = cn \sum_{i=0}^{\log_2 n} 2^i$$

$$= cn\left[\frac{2^{\log_2 n} - 1}{2-1}\right]$$

[using Geometric progression $a=1$ $r=2$]

$$\left[S_n = \frac{a(r^n - 1)}{r-1}\right]$$

$$= cn\left[n^{\log_2 2} - 1\right]$$

$$= cn[n - 1]$$

$$T(n) = cn^2 - cn$$

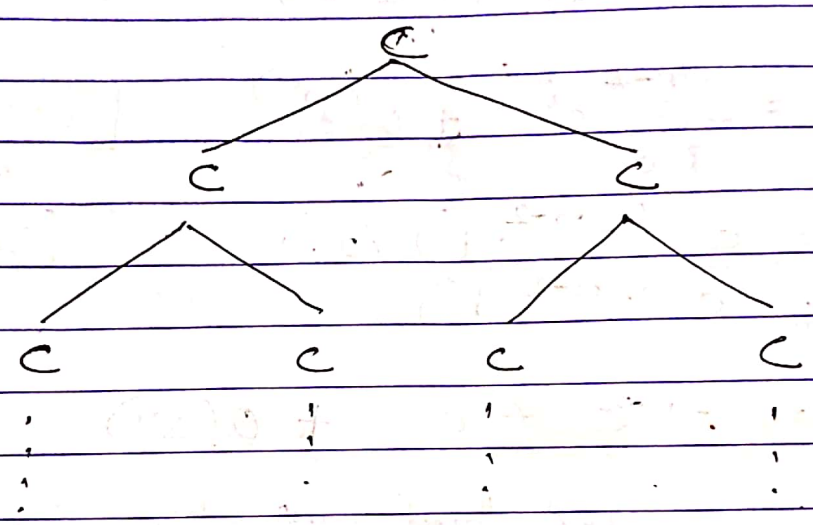
First term has higher growth i

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

Ex 3 $T(n) = 2T(n-1) + 1$

$\therefore T(n) = 2T(n-1) + C$

Step 1 : Recursion Tree

Recursive call	No. of nodes	Tree	Row Sum
$T(n)$	1		C
$T(n-1)$	2^1		$2C$
$T(n-2)$	2^2		2^2C
$T(n-i)$	2^i	C	2^iC

Step 2 : Cost at each level $= 2^iC$

Step 3 : Depth of tree i.e last level of tree

$n - i = 1$

$\boxed{i = n-1}$

Step 4 : No. of leaf nodes $= 2^i$
$= 2^{n-1}$

Step 5 : Total cost at last level $= 2^{n-1} * T(1) = \Theta(2^{n-1})$
$= \Theta(2^n)$

Step 6 : Total cost of entire tree,

$$T(n) = c + 2c + 2^2c + 2^3c + \cdots 2^{n-2}c + \Theta(2^n)$$

$$= c \sum_{i=0}^{n-2} 2^i + \Theta(2^n)$$

[using Geometric progression]

$$\left[S_n = \frac{a(r^n - 1)}{r - 1}\right]$$

$$= c\,(2^{n-2} - 1)/2 - 1$$

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

$$T(n) = 2^{n-2}c - c + \Theta(2^n)$$

First term and third terms have same order and second term is higher than second term.

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

OR

$$< c \sum_{i=0}^{\infty} 2^i + \Theta(2^n)$$

$$< c\left[\frac{1}{1-2}\right] + \Theta(2^n)$$

$$< -c + \Theta(2^n)$$

Problem on case 3

Ex: $T(n) = 2T(n/2) + \Theta(n)$

$\therefore T(n) = 2T\left(\frac{n}{2}\right) + c(n)$

Step 1: Recursion Tree.

Recursive call	No. of nodes	Tree	Row sum
$T(n)$	1	cn	cn
$T\left(\frac{n}{2}\right)$	2	$\frac{cn}{2}$ $\quad$ $\frac{cn}{2}$	cn
$T(n/2^2) =$ $T\left(\frac{n}{4}\right)$	2^2	$\frac{cn}{2^2}$ $\quad$ $\frac{cn}{2^2}$ $\quad$ $\frac{cn}{2^2}$ $\quad$ $\frac{cn}{2^2}$	cn
$T\left(\frac{n}{2^i}\right)$	2^i	$\frac{cn}{2^i}$	cn

Step 2: Cost at each level = cn

Step3 : Depth of tree i.e last level of tree,

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

$$\boxed{i = \log_2 n}$$

Step4 : No. of leaf nodes $= 2^i$

$$= 2^{\log_2 n}$$

$$= n^{\log_2 2}$$

$$= n.$$

Step5 : Total cost at last level $= n * T(1)$

$$= \Theta(n) = cn$$

Step6 : Total cost of entire tree,

$$T(n) = 1^0 cn + 1^1 cn + 1^2 cn + \cdots\cdots 1^{\log_2 n - 1} cn + \Theta(n)$$

$$= \sum_{i=0}^{\log_2 n - 1} cn + \Theta(n)$$

$$= cn \sum_{i=0}^{\log_2 n - 1} 1 + \Theta(n)$$

$$= cn (\log_2 n) + \Theta(n)$$

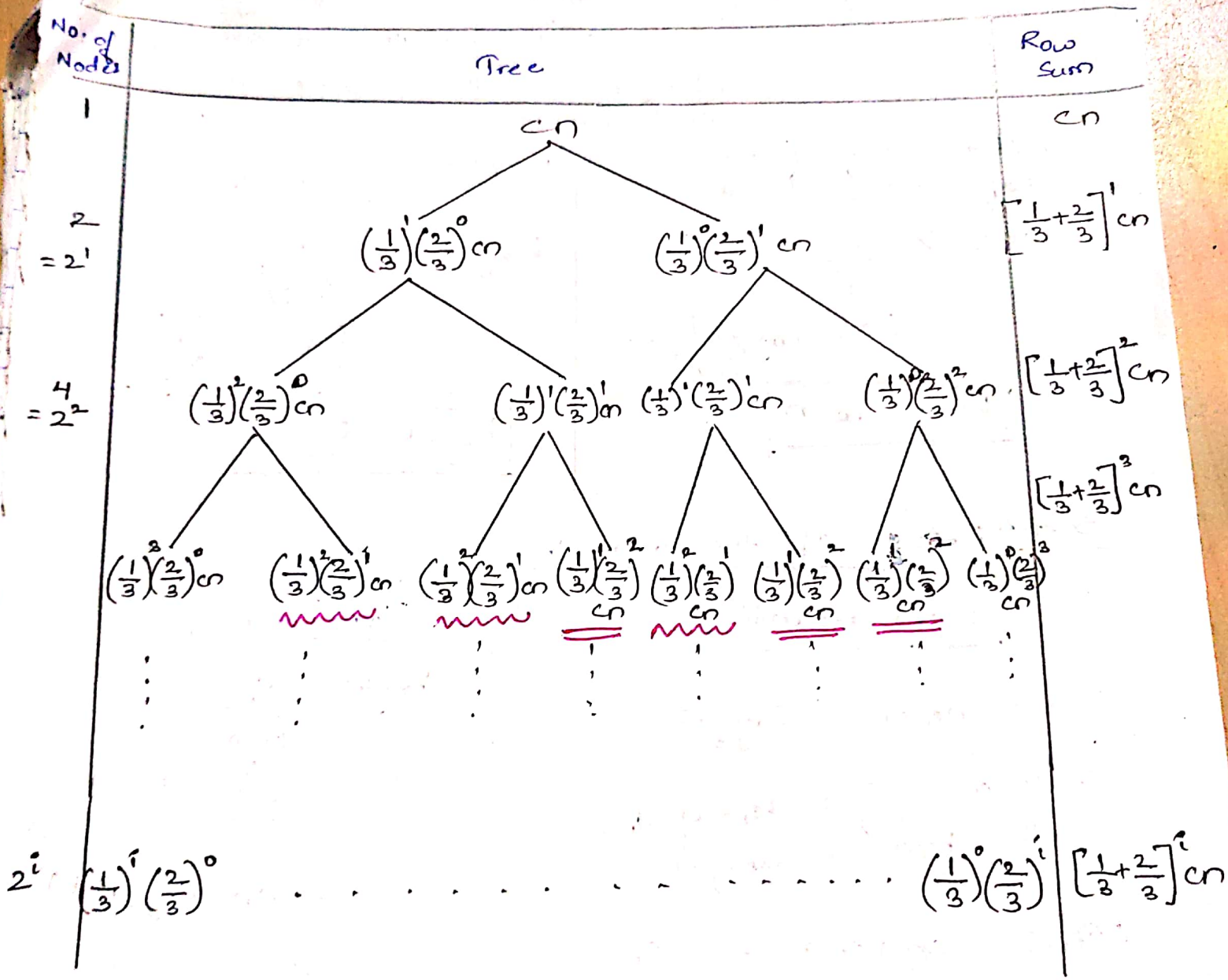
$$= cn \log_2 n + \Theta(n)$$

First term is higher than second term,

$$\therefore T(n) = \Theta(n \log_2 n)$$

$\therefore T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$ (Not in syllabus)

Step 1 : Recursion Tree



Step 2 : Cost at each level = cn

Step3 : Depth of the tree

$\because \frac{2n}{3} > \frac{n}{3}$ i.e Right subtree is doing more work than the left subtree.

∴ leaf nodes will not appear at the same level.

leftmost leaf of the leaft subtree is at.

$h_{left} \Rightarrow \frac{n}{3^i} = 1$

$\boxed{i = \log_3 n}$

$\boxed{h_{left} = \log_3 n}$

Rightmost leaf of the right subtree is at

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

$\boxed{i = \log_{\frac{3}{2}} n}$

$\boxed{h_{right} = \log_{\frac{3}{2}} n}$

Step4: Cost of the entire tree

Lower bound

$h_{left} = \log_3 n$

Consider a full binary tree with h_{left} as height.

$\therefore T(n) = cn + cn + \cdots cn$

$= \sum_{i=0}^{\log_3 n} cn$

$= cn(\log_3 n + 1)$

$= cn \log_3 n + cn$

$T(n) = \Omega(n \log_3 n)$

Upper bound

$h_{right} = \log_{\frac{3}{2}} n$

$\therefore T(n) = \sum_{i=0}^{\log_{\frac{3}{2}} n} cn$

$= cn(\log_{\frac{3}{2}} n + 1)$

$= cn \log_{\frac{3}{2}} n + cn$

$\therefore T(n) = O(n \log_{\frac{3}{2}} n)$

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

For larger values of n, (Not to syllabus)

$n - 1 \cong n$

$\therefore T(n) = T(n) + T(n/2) + n$

Step 1 : Recursion Tree

No. of Nodes	Tree	Row Sum
1	n	n
2	n, n/2	$\frac{3}{2}n$
$2^2 = 4$	n, n/2, n/2, n/4	$\left(\frac{3}{2}\right)^2 n$
2^i	n	$n/2^i$ $\left(\frac{3}{2}\right)^i n$

Step 2 : Cost at each level $= \left(\frac{3}{2}\right)^i cn$

Step 3 : Depth of the tree

$\because n > n/2$ i.e left subtree is doing more work than right subtree.

$\therefore$ leaf nodes will appear at the same level.

Leftmost leaf of the left subtree is at,

$h_{left} \Rightarrow n = i$

$\boxed{i = n}$

$\therefore \boxed{h_{left} = n}$

Rightmost leaf of the right subtree is at,

$h_{right} \Rightarrow \frac{n}{2^i} = 1$

$\therefore n = 2^i$

$\boxed{i = \log_2 n}$

$\therefore \boxed{h_{right} = \log_2 n}$

Step 4 : Cost of the entire tree

Lower Bound

$h_{right} = \log_2 n$

Consider a full binary tree with h_{right} as height

$\therefore T(n) = \sum_{i=0}^{\log_2 n} \left(\frac{3}{2}\right)^i cn$

$< \sum_{i=0}^{\infty} \left(\frac{3}{2}\right)^i cn$

$= \frac{1}{1 - \frac{3}{2}} cn$

$= -2cn$

$= \frac{\frac{3}{2}^{\log_2 n} - 1}{\frac{3}{2} - 1} [cn]$

$= 2cn\left[n^{\log_2 3/2} - 1\right]$

$= 2cn[n^{0.58} - 1]$

$= 2cn^{1.58} - 2cn$

$\therefore T(n) = \Omega(n^{1.58})$

Upper Bound

$h_{left} = n$

Consider a full BT with h_{left} as height.

$\therefore T(n) = \sum_{i=0}^{n} \left(\frac{3}{2}\right)^i cn$

$= cn\left[\frac{\frac{3}{2}^n - 1}{\frac{3}{2} - 1}\right]$

$= 2\left[\frac{3}{2}^n - 1\right] cn$

$= [3^n - 2]\, cn$

$= 3^n \cdot cn - 2cn$

$\therefore T(n) = O(n3^n)$