Solutions for Recurrence Relations using Recurrence Tree Method

CSE2003 Data Structures and Algorithms

Instructor : Dr. C. Oswald

Students : S. Girish and Harshanth K Prakash

Vellore Institute of Technology Chennai

(An Institution of Eminence recognized by MHRD)

chennai.vit.ac.in

1.) $T(n) = \begin{cases} T\left(\frac{n}{2}\right) + cn^2, & n \geq 2 \\ c, & n = 1 \end{cases}$

Recursion Tree Method :

$$
T(n) \qquad
\begin{array}{c} cn^2 \\ | \\ T\left(\frac{n}{2}\right) \end{array}
\qquad
\begin{array}{c} cn^2 \\ | \\ c\left(\frac{n}{2}\right)^2 \\ | \\ T\left(\frac{n}{4}\right) \end{array}
\qquad
\begin{array}{c} cn^2 \\ | \\ c\left(\frac{n}{2}\right)^2 \\ | \\ c\left(\frac{n}{4}\right)^2 \\ | \\ T\left(\frac{n}{8}\right) \end{array}
$$

$$
\begin{array}{c|ccc}
 & cn^2 & \dashrightarrow & cn^2 \\
\text{Height} & | & & \\
 & c\left(\frac{n}{2}\right)^2 & \dashrightarrow & c\left(\frac{n}{2}\right)^2 \\
\log_2 n & | & & \\
 & c\left(\frac{n}{4}\right)^2 & \dashrightarrow & c\left(\frac{n}{4}\right)^2 \\
 & \vdots & & \\
 & \underbrace{T(1)}_{1 \text{ node}} & \dashrightarrow & \Theta(1) \\
\hline
 & & & \text{Total} : O(n^2)
\end{array}
$$

From the above recursion tree, we derive at the following :

* For convenience, we assume that n is an exact power of 2, so that all sub problem sizes are integers.
* Because subproblem sizes decrease by a factor of 2, each time we go down one level, we eventually must reach a boundary condition.
* The subproblem size for a node at depth i is $\frac{n}{2^i}$.
* Thus, the subproblem size hits $n = 1$ when $\frac{n}{2^i} = 1$ or equivalently, when $i = \log_2 n$.

* Thus, the tree has $\log_2 n + 1$ levels (at depths $0, 1, 2 \ldots \log_2 n$).
* Each level has only 1 node and so the number of nodes at depth i is 1.
* Because, subproblem sizes reduce by a factor of 4 for each level we go down from the root, each node at depth i, for $i = 0, 1, 2, \ldots \log_2 n - 1$, has a cost of $c\left(\frac{n}{4^i}\right)$.
* Thus, the total cost over all nodes at depth i for $i = 0, 1, 2 \ldots \log_2 n - 1$, is $(1)\left(c\left(\frac{n}{4^i}\right)\right) = c\left(\frac{n}{4^i}\right)$.
* The bottom level at depth $\log_2 n$ has $1^{\log_2 n} = 1$ node (or leaf), with cost $T(1)$, for a total cost of $\Theta(1)$.

$$T(n) = cn^2 + \left(\frac{1}{4}\right)cn^2 + \left(\frac{1}{4}\right)^2 cn^2 + \ldots \left(\frac{1}{4}\right)^{\log_2 n - 1} cn^2 + \Theta(1)$$

$$= \sum_{i=0}^{\log_2 n - 1} \left(\frac{1}{4}\right)^i cn^2 + \Theta(1)$$

$$= cn^2 \left[\frac{(1)\left(1 - \left(\frac{1}{4}\right)^{\log_2 n}\right)}{\left(1 - \frac{1}{4}\right)}\right] = cn^2\left[\left(\frac{4}{3}\right)\left(1 - \left(2^{-2}\right)^{\log_2 n}\right)\right]$$

$$= \frac{4}{3}cn^2\left[1 - \frac{1}{n^2}\right] + \Theta(1)$$

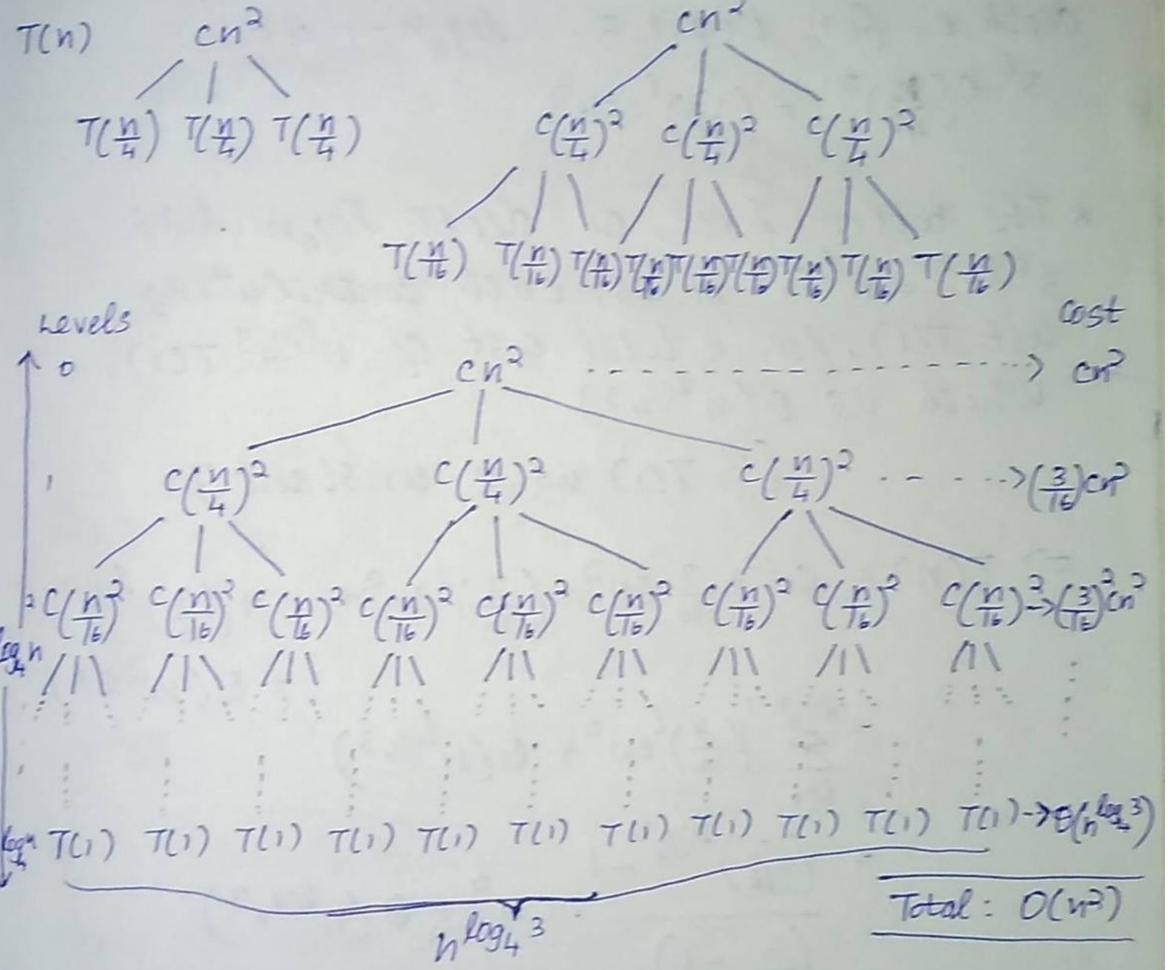
$$= \frac{4}{3}cn^2 - \frac{4}{3}c + \Theta(1)$$

$$\boxed{T(n) = O(n^2).}$$

2) $T(n) = \begin{cases} 3T(\frac{n}{4}) + cn^2, & n \geq 4 \\ 1, & n = 1 \end{cases}$

A:

Recursion Tree:



From the above recursion tree, we conclude the following:

* The subproblem size for a node at depth i is $\frac{n}{4^i}$.
* Thus, the subproblem size hits $n=1$ when $\frac{n}{4^i} = 1$ or equivalently, when $i = \log_4 n$.
* Thus, the tree has $\log_4 n + 1$ levels (at depths $0, 1, 2, \ldots, \log_4 n$)

* Because subproblem sizes reduce by a factor of 4 for each level we go down from the root, each node at depth i, for $i = 0, 1, 2 \ldots \log_4 n - 1$, has a cost of $c\left(\frac{n}{4^i}\right)^2$.

* Thus, the total cost over all nodes at depth i, for $i = 0, 1, 2 \ldots \log_4 n - 1$ is
$$3^i \cdot c\left(\frac{n}{4^i}\right)^2 = \left(\frac{3}{16}\right)^i \cdot cn^2.$$

* The bottom level, at depth $\log_4 n$ has $3^{\log_4 n} = n^{\log_4 3}$ nodes, each contributing cost $T(1)$, for a total cost of $n^{\log_4 3} \cdot T(1)$, which is $\Theta(n^{\log_4 3})$
[$\because T(1)$ is a constant.]

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

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

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

Since, we require upper bound, we can the infinite G.P. formula.

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

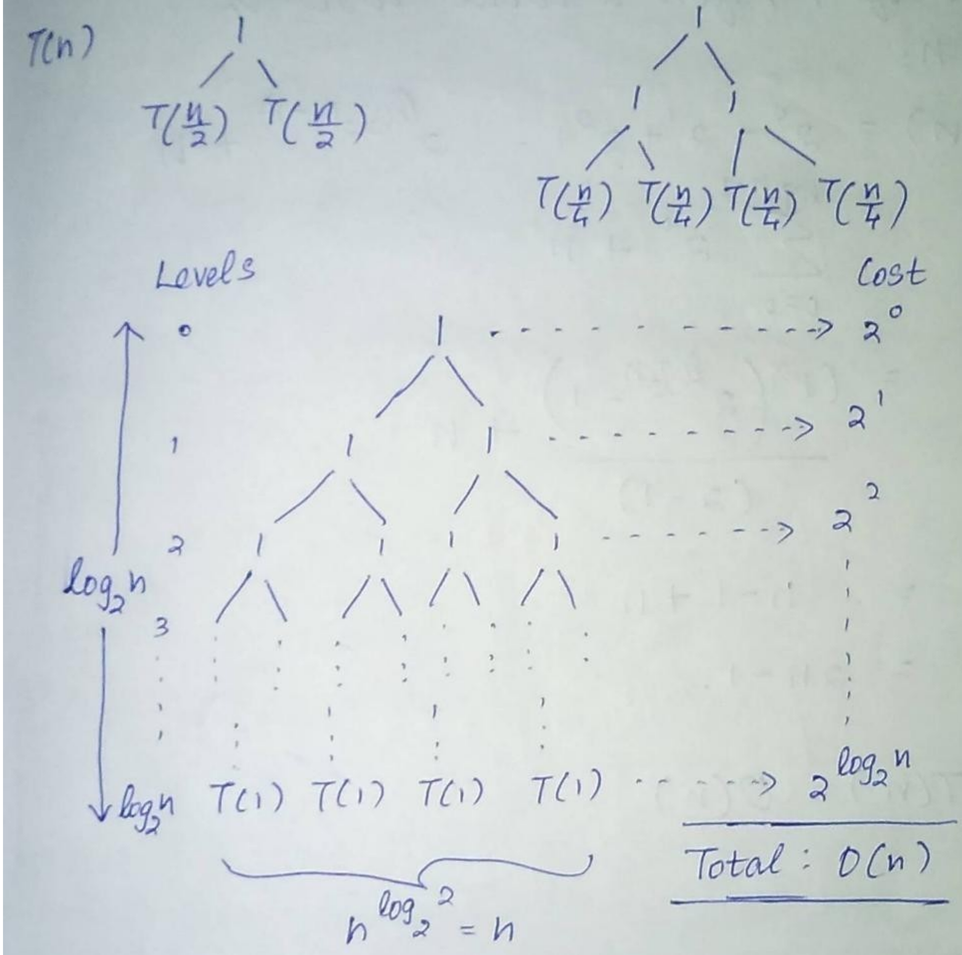
$$= \frac{1}{\left(1 - \frac{3}{16}\right)} cn^2 + \Theta(n^{\log_4 3}) = \frac{16}{13} cn^2 + \Theta(n^{\log_4 3})$$

$$\Rightarrow \boxed{T(n) = O(n^2).}$$

3) $T(n) = \begin{cases} 2T(\frac{n}{2}) + 1 & , n \geq 2 \\ 1 & , n = 1 \end{cases}$

A:

Recursion Tree:



From the above recursion tree, we conclude the following:

* The subproblem size for a node at depth i is 2^i.
* Thus, the subproblem size for a node hits $n = 1$ when $\frac{n}{2^i} = 1$ or $i = \log_2 n$.
* Thus, the tree has $\log_2 n + 1$ levels.
* Since, each subproblem reduce by a factor of 2 for each level we go down from the root, each node at depth i, for $i = 0, 1, 2 \ldots \log_2 n - 1$, has a cost of 2^i.

* Thus, total cost at depth $i = 2^i$.
* The bottom level, at depth $\log_2 n$ has $2^{\log_2 n} = n$ nodes, each contributing a cost of 1, for a total cost of $1 \cdot n = n$.

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

$$= \sum_{i=0}^{\log_2 n - 1} 2^i + n$$

$$= \frac{(1)(2^{\log_2 n} - 1)}{(2-1)} + n$$

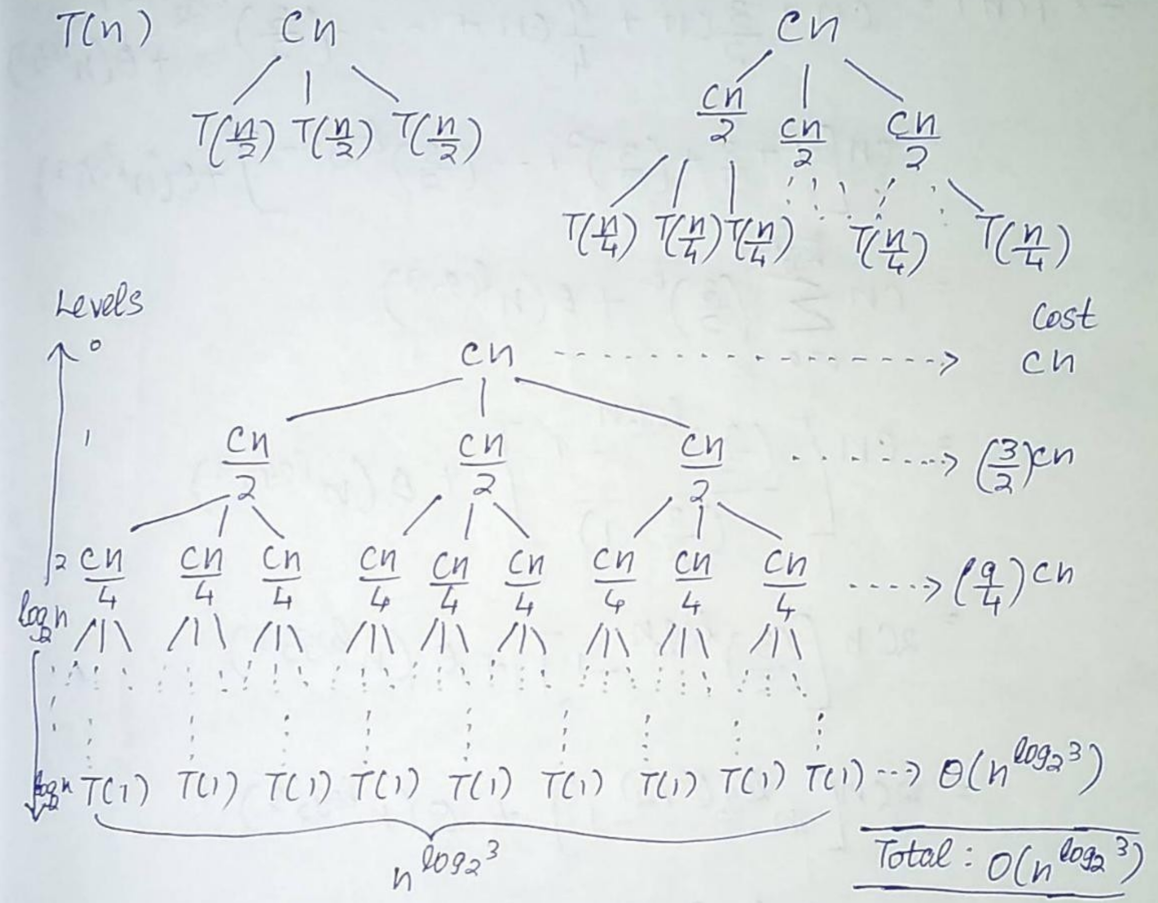
$$= n - 1 + n$$

$$= 2n - 1.$$

$$\Rightarrow \boxed{T(n) = O(n).}$$

4.) $T(n) = \begin{cases} 3T\left(\frac{n}{2}\right) + cn, & n \geq 2 \\ 1, & n = 1 \end{cases}$

A: Recursion Tree :



From the above recursion tree, we derive the following :

- The subproblem size at depth i is $\frac{n}{2^i}$.
- Thus, subproblem size $n = 1$ when $\frac{n}{2^i} = 1$ or $i = \log_2 n$.
- Thus, the tree has $\log_2 n + 1$ levels.
- Thus, the total cost over all nodes at depth i, for $i = 0, 1, 2 \ldots, \log_2 n - 1$ is $3^i\left(\frac{n}{2^i}\right) = \left(\frac{3}{2}\right)^i \cdot n$.

★ The bottom level, at depth $\log_2 n$ has $3^{\log_2 n} = n^{\log_2 3}$ nodes, each contributing cost $T(1)$, for a total cost of $n^{\log_2 3} \cdot T(1)$, which is $\theta(n^{\log_2 3})$.

$$\Rightarrow T(n) = cn + \frac{3}{2}cn + \frac{9}{4}cn + \cdots \left(\frac{3}{2}\right)^{\log_2 n - 1} + \theta(n^{\log_2 3})$$

$$= cn\left[1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \cdots \left(\frac{3}{2}\right)^{\log_2 n - 1}\right] + \theta(n^{\log_2 3})$$

$$= cn \sum_{i=0}^{\log_2 n - 1} \left(\frac{3}{2}\right)^i + \theta(n^{\log_2 3})$$

$$= cn\left[\frac{\left(\frac{3}{2}\right)^{\log_2 n} - 1}{\left(\frac{3}{2} - 1\right)}\right] + \theta(n^{\log_2 3})$$

$$= 2cn\left[\left(\frac{3}{2}\right)^{\log_2 n} - 1\right] + \theta(n^{\log_2 3})$$

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

$$= 2cn\left[n^{\log_2 3 - \log_2 2} - 1\right] + \theta(n^{\log_2 3})$$

$$= 2cn\left[n^{\log_2 3 - 1} - 1\right] + \theta(n^{\log_2 3})$$

$$= 2c\left[n^{\log_2 3 - \cancel{1} + \cancel{1}} - n\right] + \theta(n^{\log_2 3})$$

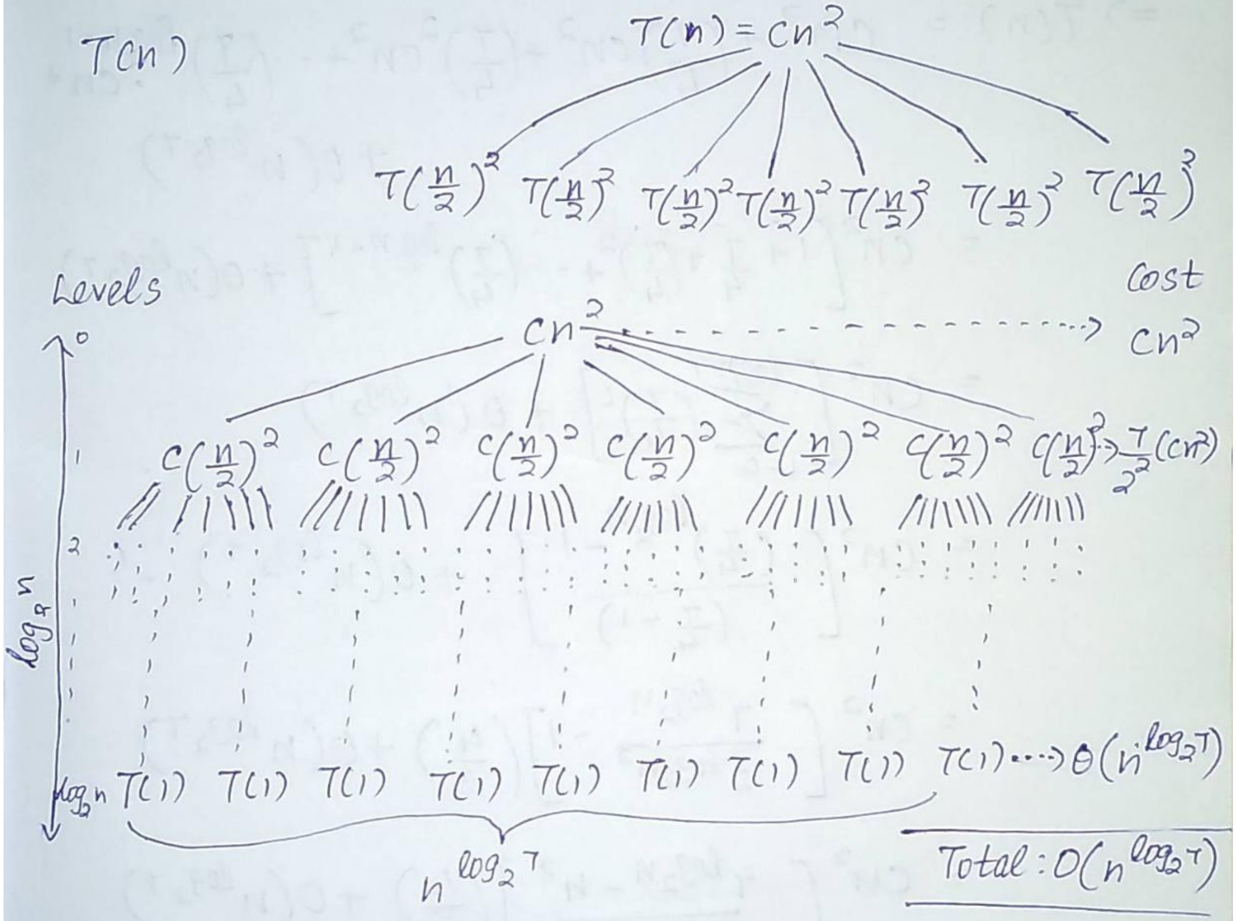
$$= 2cn^{\log_2 3} - 2cn + \theta(n^{\log_2 3})$$

$$= O(n^{\log 3})$$

$$\therefore \boxed{T(n) = O(n^{\log 3})}$$

5.) $T(n) = \begin{cases} 7T(\frac{n}{2}) + cn^2 & , n \geq 2 \\ 1 & , n = 1 \end{cases}$

A: Tree Method:



From the above recursion tree, we derive the following:

* The subproblem size at depth i is $\frac{n}{2^i}$.
* Thus at $n=1$; $\frac{n}{2^i} = 1 \Rightarrow i = \log_2 n$.
* Thus, the tree has $\log_2 n + 1$ levels.
* Thus, the total cost over all nodes at depth i, for $i = 0, 1, 2 \ldots \log_2 n - 1$ is $7^i \left(\frac{n}{2^i}\right)^2 = \left(\frac{7}{4}\right)^i \cdot n^2$

* The bottom level, at depth $\log_2 n$ has $7^{\log_2 n} = n^{\log_2 7}$ nodes, each of $T(1)$, thus total cost is $\theta(n^{\log_2 7})$.

$$\Rightarrow T(n) = cn^2 + \left(\frac{7}{4}\right)cn^2 + \left(\frac{7}{4}\right)^2 cn^2 + \cdots \left(\frac{7}{4}\right)^{\log_2 n - 1} \cdot cn^2 + \theta(n^{\log_2 7})$$

$$= cn^2\left[1 + \frac{7}{4} + \left(\frac{7}{4}\right)^2 + \cdots \left(\frac{7}{4}\right)^{\log_2 n - 1}\right] + \theta(n^{\log_2 7})$$

$$= cn^2\left[\sum_{i=0}^{\log_2 n - 1} \left(\frac{7}{4}\right)^i\right] + \theta(n^{\log_2 7})$$

$$= cn^2\left[\frac{\left(\frac{7}{4}\right)^{\log_2 n} - 1}{\left(\frac{7}{4} - 1\right)}\right] + \theta(n^{\log_2 7})$$

$$= cn^2\left[\frac{7^{\log_2 n}}{2^{\log_2 n^2}} - 1\right]\left(\frac{4}{3}\right) + \theta(n^{\log_2 7})$$

$$= c\cancel{n^2}\left[\frac{7^{\log_2 n} - n^2}{\cancel{n^2}}\right]\left(\frac{4}{3}\right) + \theta(n^{\log_2 7})$$

$$= \frac{4}{3}c \cdot 7^{\log_2 n} - \frac{4}{3}cn^2 + \theta(n^{\log_2 7})$$

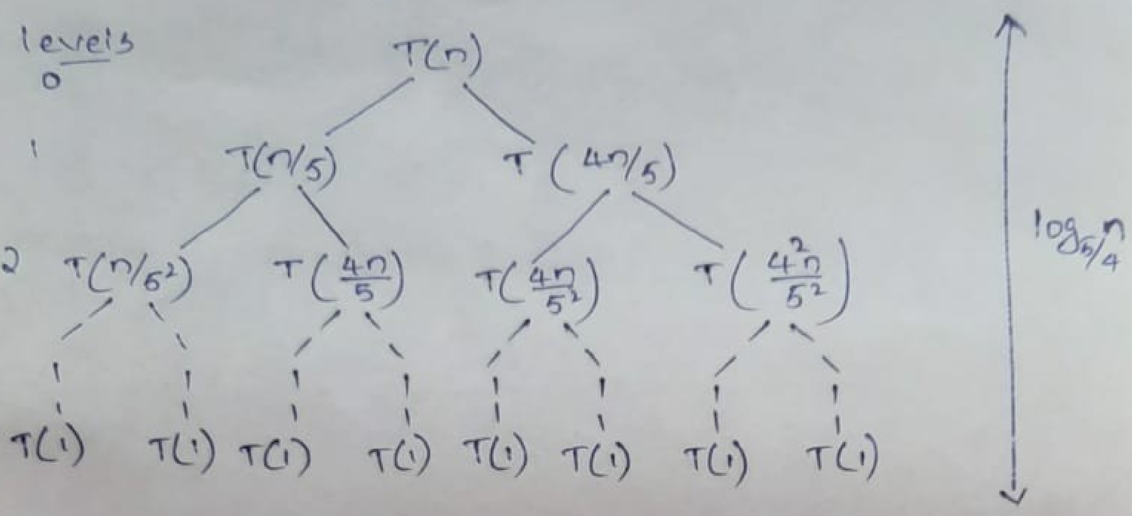
$$= O(n^{\log_2 7}) = O(n^{2.81})$$

$$\therefore \boxed{T(n) = O(n^{\log_2 7})}$$

Recursion Tree

Q. $T(n) = T(n/5) + T(4n/5) + n$

Sol:- Drawing a recursion tree and illustrating the following relation.



The given problem of size n is divided into two subproblems one of size $n/5$ and other of size $4n/5$

And then $n/5$ will be getting divided into two $\frac{n}{5^2}$ and $\frac{4n}{5^2}$, similarly $\frac{4n}{5}$ will also be getting divided into $\frac{4n}{5^2}$ and $\frac{4^2 n}{5^2}$

This process repeats and at the bottom most layer the size of subproblems will be approx (1).

The cost at each level will be n,

eg: cost of dividing problem of size $n/5$ into 2 sub problems & combining solution is $n/5$

In case of $4n/5$, it will be also $4n/5$ & so on.

$\therefore$ cost at level-0 $= n$

" level-1 $= n/5 + 4n/5 = n$

" level-2 $= n/5^2 + 4n/5^2 + 4n/5^2 + \frac{4^2 n}{5^2}$

$= \underline{n}$

The size of subproblems at each levels

At level-0 $= (4/5)^0 n$

It is calculated as rightmost sub-tree as it goes down to the deepest level

At level-1 $= (4/5)^1 n$, at level-2 will be $(4/5)^2 n$

$\therefore$ No of nodes will be of;

level-0 has 2^0 nodes $= 1$ node

level-1 " 2^1 nodes $= 2$ node

level-2 has 2^2 nodes $= 4$ node.

$\therefore$ level- $\log_{5/4} n$ $\rightarrow$ $2^{\log_{5/4} n}$ nodes

$\therefore$ cost at last level $= 2^{\log_{5/4} n} \times T(1)$

$= \Theta(n^{\log_{5/4} 2})$

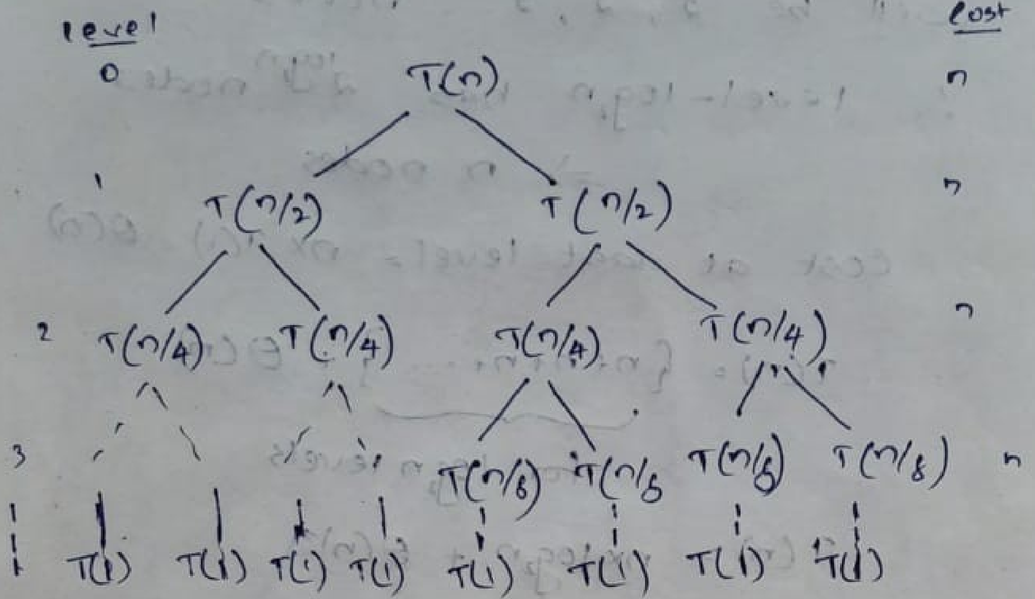
$T(n) = \underbrace{\{n + n + \cdots\}}_{\log_{5/4} n \text{ levels}} + \Theta(n^{\log_{5/4} 2})$

$[n \log_{5/4} n + \Theta(n^{\log_{5/4} 2})]$

$T(n) = \underline{\underline{\Theta(n \log_{5/4} n)}}$

Q. $T(n) = 2T(n/2) + n$

Sol:- Recursive tree



Here the given problem is divided into 2 subproblems, i.e. $n/2$ & $n/2$ & then combining solution

Cost at each level;

At level-0 $= n$

At level-1 $= n/2 + n/2 = n$

At level-2 $= n/4 + n/4 + n/4 + n/4 = n$

so, on

Total no. of levels will be;

size of subproblems at each levels will be $n/2^0$, $n/2^1$, $n/2^2$ ...

$\|^{ly}$ size of sub problem at level-i $= n/2^i$.

$\therefore$ at level-n

$\frac{n}{2^n} = 1$, $2^n = n$ $[\log_2 2^n = \log_2 n]$

$\therefore n = \log_2 n$

$\therefore$ Total no of levels $= \log_2 n + 1$

For number of nodes at each levels will be $2^0, 2^1, 2^2 \dots$ nodes

$\therefore$ level $-\log_2 n$ has $2^{\log_2 n}$ nodes

$\Rightarrow n$ nodes

Cost at last level $= n \times T(1) = \Theta(n)$

$$T(n) = \underbrace{\{n + n + \dots\}}_{\text{for } \log_2 n \text{ times}} + \Theta(n)$$

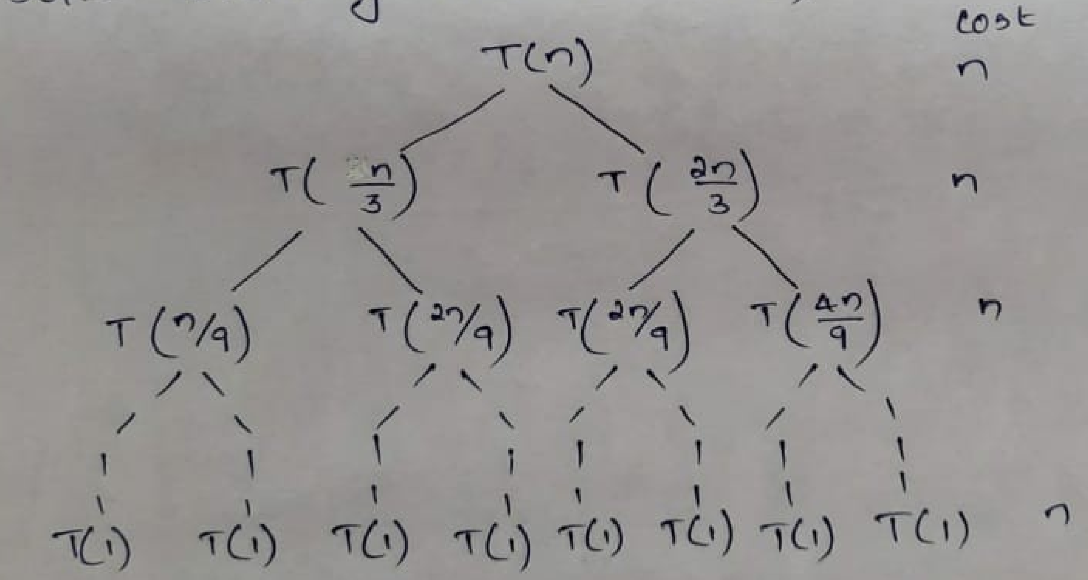
$$T(n) = n \times \log_2 n + \Theta(n)$$

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

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

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

Sol:- Drawing recursive tree;



In this problem, it is divided into two subproblem of size $n/3$ and $2n/3$, As the same process, we can focus on the rightmost subproblem at each level to get the upper bound of the algorithm

The cost of each level will be equal to cn and considering only the longest path ie the number of levels (nodes) in the longest path.

In case of the size of subproblems

$n \rightarrow \left(\frac{2n}{3}\right)n \rightarrow \left(\frac{2n}{3}\right)^2 n \rightarrow \cdots \rightarrow 1$, so the size of sub problems keeps on decreasing

So at any level i, the size of sub problem $\Rightarrow \left(\frac{2}{3}\right)^i n$

For last level,

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

$$n = (3/2)^i, \quad i = \log_{3/2} n$$

We know that there are $1 + \log_{3/2} n$ levels in the longest path and each level will take atmost cn time ie,

$$T(n) = \{cn + cn + \cdots cn\} \rightarrow \text{For } 1 + \log_{3/2} n \text{ times}$$

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

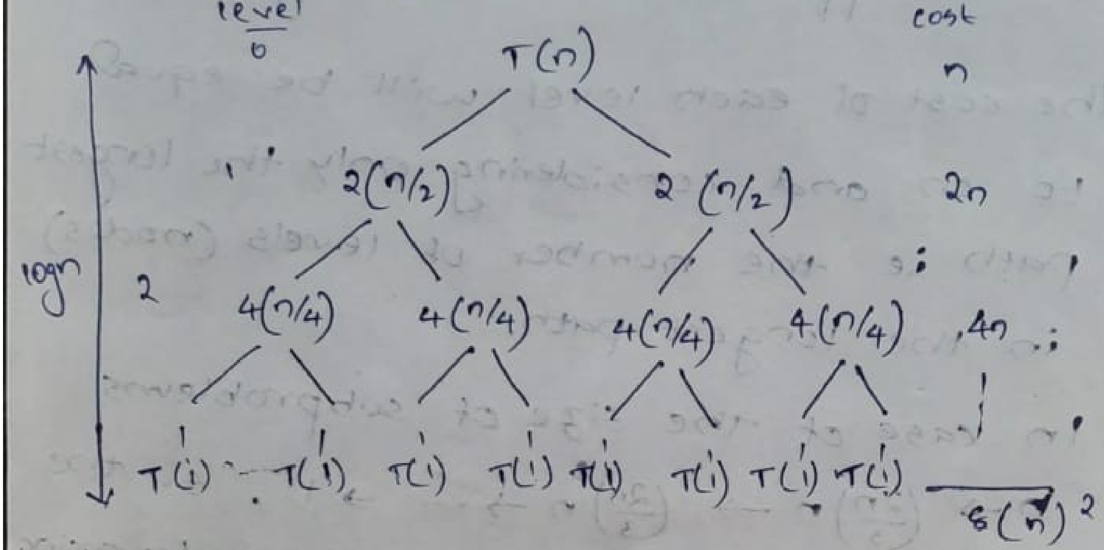
$$= cn + cn \frac{\lg n}{\lg 3/2}$$

[ignoring lower order & constants]

$$\therefore T(n) = O(n \lg n)$$

Q. $T(n) = 4T(n/2) + n$

Sol:- Drawing a recursive tree,



Here the given problem is divided into 2 subproblems, ie $2(n/2)$, $2(n/2)$.

Cost at each levels will be $n, 2n, 4n \ldots$

Total size of subproblems will be of $\log n$

$\therefore$ we have;

$$n + 2n + 4n + \cdots \log_2 n \text{ times}$$

$$= n\,(1 + 2 + 4 + \cdots + \log_2 n)$$

$$\Rightarrow n\,\frac{(2\log_2 n - 1)}{(2-1)}$$

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

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

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

References :

1. E.Horowitz, S.Sahni, S.Rajasekaran, Fundamentals of Computer Algorithms, Galgotia Publications.

2. T.H. Cormen, C.E. Leiserson, R.L.Rivest, C.Stein,
   Introduction to Algorithms, PHI.

3. Sara Baase, A.V.Gelder, Computer Algorithms, Pearson.

4. Mark Allen Weiss, Data Structures and Algorithms in C ,
   Pearson Publications.

5. Alfred V. Aho, Jon E. Hopcroft, Jeffrey D. Ullman, Design & Analysis of Computer Algorithms, PEarson Publications.

6. www.cse.iiitdm.ac.in/lectnotes.html