(5) RECURSION TREE METHOD:

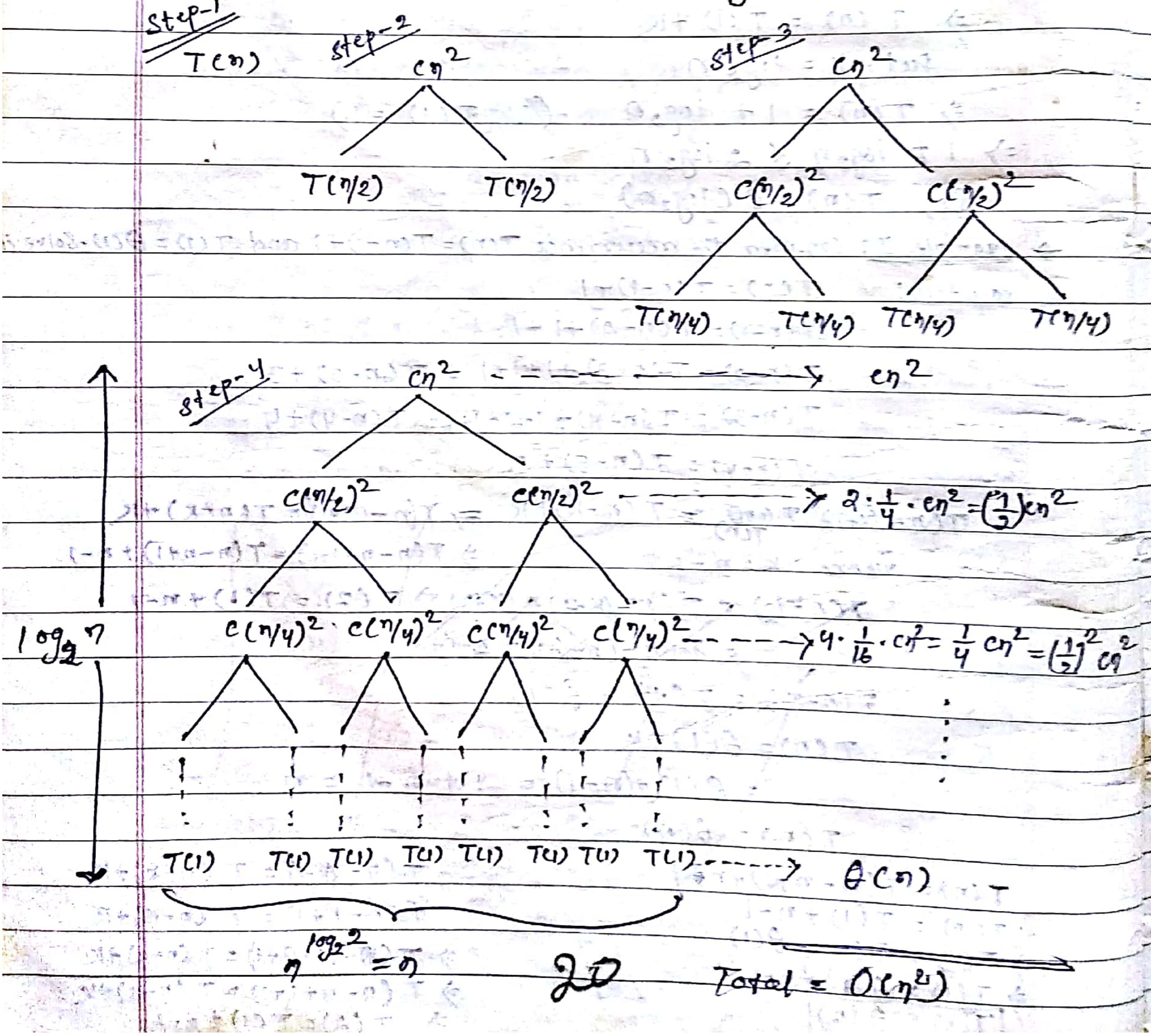
→ Recursion Tree method is a pictorial representation of an iteration method, which is in the form of a tree, where at each level nodes are expanded.

→ We consider second term in recurrence as root. It is useful when divide and conquer algorithm is used.

→ In a recursion tree, each node represents the cost of a single subproblem. We sum the costs within each level of the tree to obtain a set of per-level costs. Then we sum all the per level costs to determine the total cost of all levels of the recursion.

→ Example 1: The recurrence is $T(n) = 2T(n/2) + \theta(n^2)$.
We create a recurrence tree for the recurrence

$$T(n) = 2T(n/2) + cn^2, \text{ for any constant } c > 0.$$



The subproblem size decreases as we get further from the root. At last we must reach a boundary condition. The subproblem sizes of a node at depth i is $n/2^i$.

Thus, the subproblem size hit $n=1$

$\Rightarrow n/2^i = 1$

$\Rightarrow n = 2^i$

$\Rightarrow \log n = \log 2^i$

$\Rightarrow \log_2 n = i \cdot \log 2$

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

Thus the depth of the tree is $\log_2 n$ and the tree has $\log_2 n + 1$ levels $(0, 1, 2, \dots, \log_2 n)$

Now we should determine the cost at each level of the tree. Each level has two times more nodes than the above level. So, the number of nodes at depth i is 2^i. Because subproblem sizes reduce by factor of 2 for each level we go down from the root, each node at depth i, for $i = 0, 1, 2, \dots \log_2 n - 1$ has a cost of $c(n/2^i)^2$.

So, the total cost over all nodes at depth i, for $i = 0, 1, \dots, \log n - 1$ is $2^i c(n/2^i)^2 = (2/4)^i cn^2 = (1/2)^i cn^2$

The last level, at the depth $\log_2 n$ has:

$$2^{\log_2 n} = n^{\log_2 2} = n \text{ nodes, each contributing cost } T(1)$$

for a total cost of $n, T(1)$ which is $\Theta(n)$

Now we add costs of all the levels to determine the costs over all the levels to determine the cost of entire tree is =

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

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

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

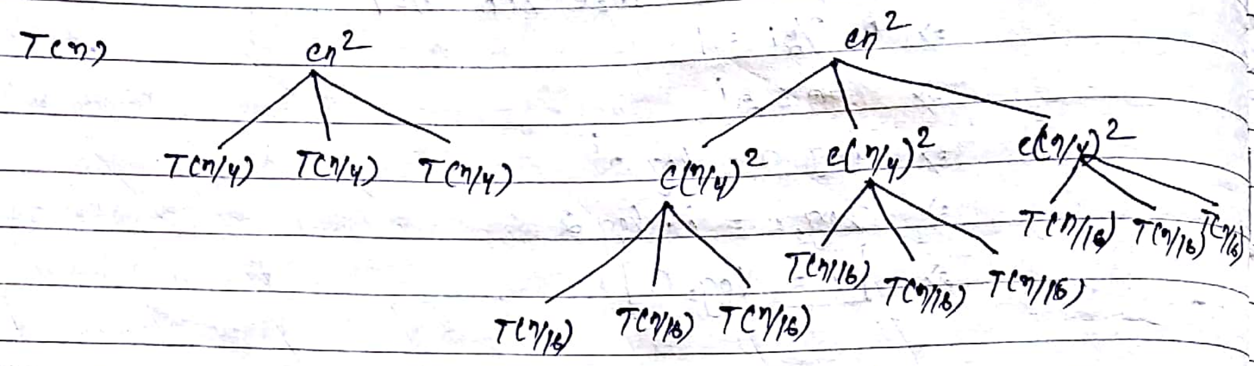
$$= 2cn^2 + \Theta(n)$$

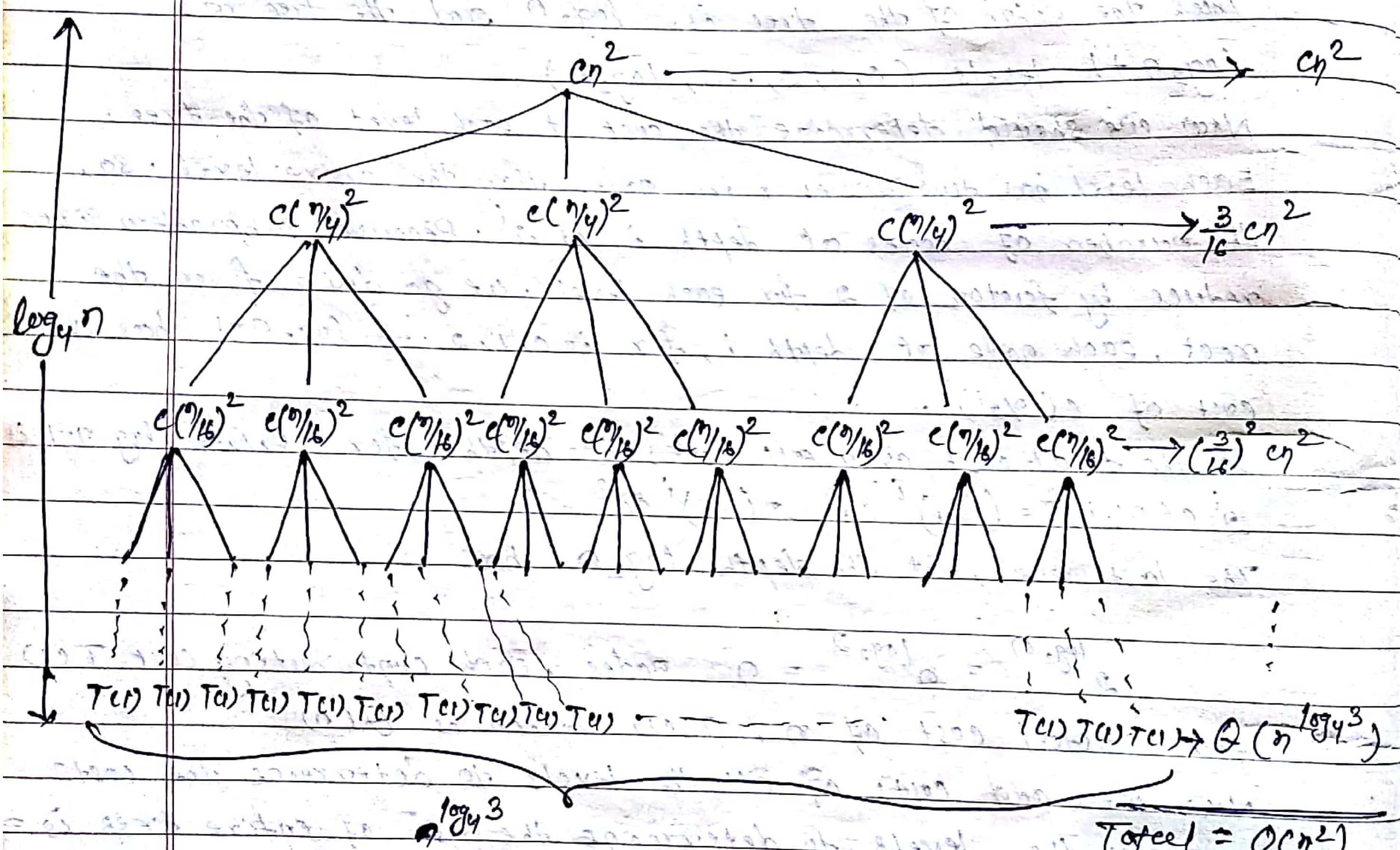
$$= O(n^2)$$

Example 2: The recurrence is $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

we create the recurrence tree for the recurrence

$$T(n) = 3T(n/4) + cn^2$$





The subproblem size for a node at depth i is $n/4^i$.

Thus, the subproblem size hits $n=1$ when $n/4^i = 1$

$\Rightarrow n = 4^i$

$\Rightarrow \log n = \log 4^i$

$\Rightarrow i = \log_4 n$

$\therefore$ The depth of the tree is $\log_4 n$.

The tree has $\log_4 n + 1$ levels $(0, 1, 2, \ldots, \log_4 n)$

Now we should determine the cost at each level of the tree.

Each level has three times more nodes than the above level.

So, the number of nodes at depth i is 3^i.

The cost of each node at depth i is $c(n/4^i)^2$ because the subproblem sizes reduce by a factor of 4 for each level when we go down from root.
The total cost over all the nodes at depth i is

$$\cancel{3^i c(n/4^i)^2 = (3/16)^i cn^2}$$

$$3^i c(n/4^i)^2 = (3/16)^i cn^2$$

The last level, at depth $\log_4 n$ has

$$3^{\log_4 n} = n^{\log_4 3} \text{ nodes.}$$

Each node has cost $T(1)$.
So, Total cost at last level $= n^{\log_4 3} T(1) = \Theta(n^{\log_4 3})$
The cost for entire tree is

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

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

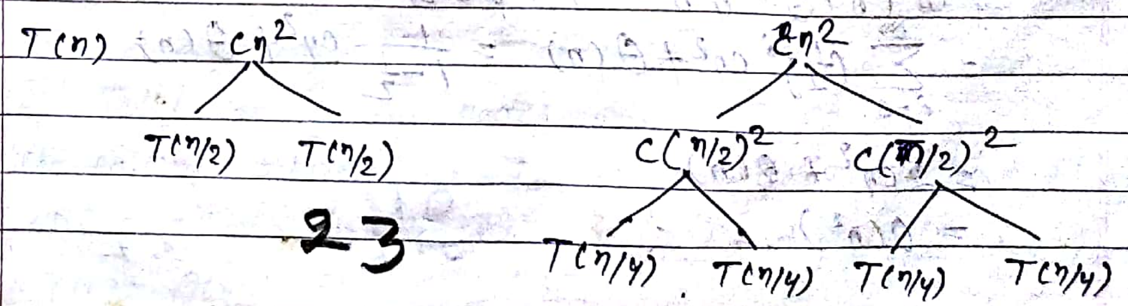
$$< \sum_{i=0}^{\infty} \left(\tfrac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3})$$

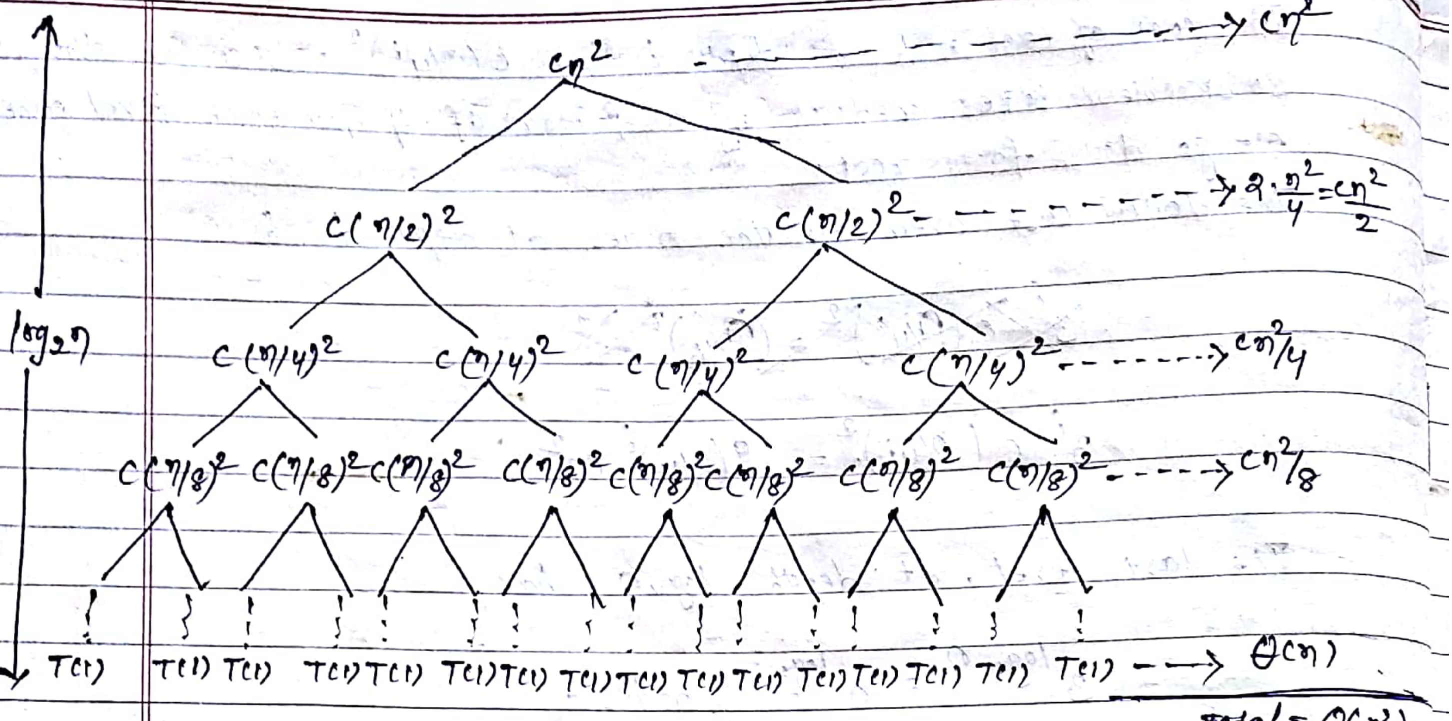
$$= \frac{1}{1 - 3/16} cn^2 + \Theta(n^{\log_4 3})$$

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

$$= \cancel{\Theta(n^2)} = O(n^2)$$

Example 3: Consider $T(n) = 2T\left(\frac{n}{2}\right) + n^2$
we have to obtain the asymptotic bound using recursion tree method.
Solution: The recurrence tree of $T(n) = 2T(n/2) + n^2$ is





The subproblem size for a node at depth i = $n/2^i$

The subproblem size hits $n=1$ when $n/2^i = 1$

$\Rightarrow n = 2^i$

$\Rightarrow \log n = \log 2^i$

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

The depth of the tree is $\log_2 n$.

The tree has $\log_2 n + 1$ levels i.e. $(0, 1, 2, \cdots, \log_2 n)$

The no. of nodes at depth i is 2^i

The cost of each node at depth i = $c(n/2^i)^2$.

The total cost of all the nodes at depth i is

$$2^i \cdot c(n/2^i)^2 = \left(\frac{2}{4}\right)^2 cn^2 = \left(\frac{1}{2}\right)^2 cn^2$$

The no. of nodes at depth $\log_2 n$ is n

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

Each node having cost $T(1)$

So, Total cost of all nodes are $\Theta(n)$

The cost of entire tree is

$$T(n) = cn^2 + \frac{cn^2}{2} + \frac{cn^2}{4} + \frac{cn^2}{8} + \cdots + \Theta(n)$$

$$= \cancel{cn^2\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right) + \Theta(n)}$$

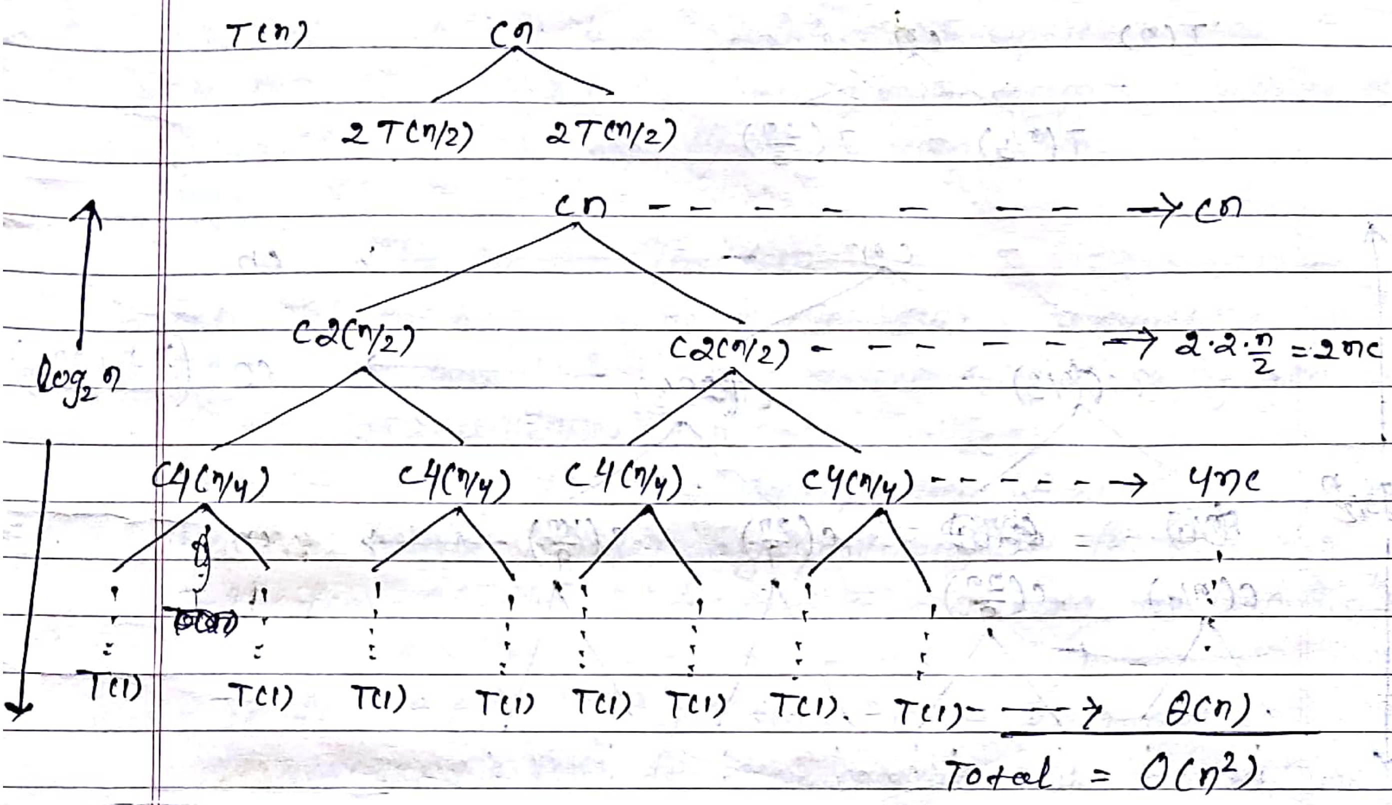
$$= \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i cn^2 + \Theta(n) = \frac{1}{1-\frac{1}{2}} cn^2 + \Theta(n)$$

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

$$= O(n^2)$$

Example 4: Consider the following recurrence $T(n) = 4T(\lfloor \frac{n}{2} \rfloor) + n$ obtain the asymptotic bound using recursion tree method.

Solution: The recursion tree for the above recurrence is



The subproblem size for a node at depth $i = n/2^i$

~~The subproblem~~ $n/2^i = 1 \Rightarrow n = 2^i$

$\Rightarrow \log n = \log 2^i$

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

The depth of the tree is $\log_2 n$.

The tree has $\log_2 n + 1$ levels i.e. $(0, 1, 2, \ldots, \log_2 n)$

The no. of nodes at depth i is 2^i

The cost of each node at depth $i = 2c(n/2^i)^2$

The total cost of all nodes at depth i is

$2^i \cdot 2c(n/2^i)^2 =$

The no. of nodes at depth $\log_2 n$ is

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

Each node having cost $T(1)$

So, Total cost of all nodes are $\theta(n)$

The cost of entire tree is $cn + 2cn + 4cn + \cdots + \log_2 n$ times

$= n(1 + 2 + 4 + \cdots \log_2 n \text{ times}) = n\left(\frac{2^{\log_2 n} - 1}{2 - 1}\right) = \frac{n(n-1)}{1} = O(n^2)$

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

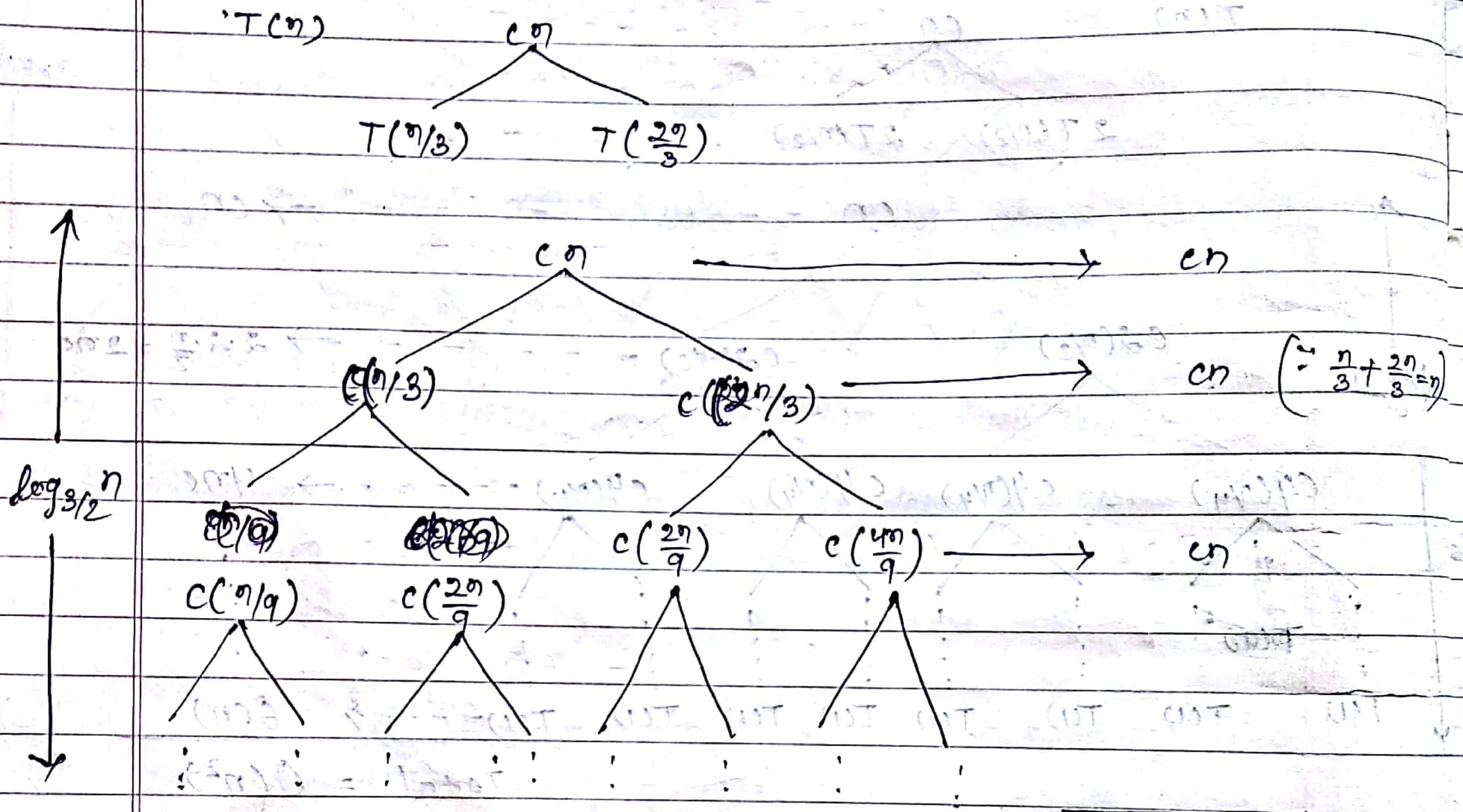
25

Example 5: Consider the following recurrence

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

obtain the asymptotic bound using recurrence tree method.

Solution:



Total = $O(n \log n)$

when we add the values across the levels of the recursion tree we get a value of n for every level. The longest path from the root to a leaf is

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

Since $\left(\frac{2}{3}\right)^i n = 1 \quad \Rightarrow i = \log_{3/2} n$ $\quad \Rightarrow \frac{n}{\cancel{\ }} = \left(\frac{3}{2}\right)^i$

Thus the height of the tree is $\log_{3/2} n$ $\quad \Rightarrow \log n = \log \left(\frac{3}{2}\right)^i$

The cost of entire tree is $\quad \Rightarrow \log_{3/2} n = i \log_{3/2} \frac{3}{2}$

$$T(n) = n + n + n + \cdots + \log_{3/2} n \text{ terms}$$

$$= O(n \log n) \qquad \Rightarrow \boxed{i = \log_{3/2} n}$$