Algorithm Analysis and Design

Recurrence Equation

(Solving Recurrence using Recursion Tree Methods)

Lecture – 9 and 10

Overview

- A ***recurrence*** is a function is defined in terms of
  - one or more base cases, and
  - itself, with smaller arguments.

Examples:

- $T(n) = \begin{cases} 1 & \text{if } n = 1, \\ T(n-1) + 1 & \text{if } n > 1. \end{cases}$

  Solution: $T(n) = n$.

- $T(n) = \begin{cases} 1 & \text{if } n = 1, \\ 2T(n/2) + n & \text{if } n \geq 1. \end{cases}$

  Solution: $T(n) = n \lg n + n$.

- $T(n) = \begin{cases} 0 & \text{if } n = 2, \\ T(\sqrt{n}) + 1 & \text{if } n > 2. \end{cases}$

  Solution: $T(n) = \lg \lg n$.

- $T(n) = \begin{cases} 1 & \text{if } n = 1, \\ T(n/3) + T(2n/3) + n & \text{if } n > 1. \end{cases}$

  Solution: $T(n) = \Theta(n \lg n)$.

Overview

- Many technical issues:
  - Floors and ceilings

    [Floors and ceilings can easily be removed and don't affect the solution to the recurrence. They are better left to a discrete math course.]
  - Exact vs. asymptotic functions
  - Boundary conditions

Overview

In algorithm analysis, the recurrence and it's solution are expressed by the help of asymptotic notation.

- Example: $T(n) = 2T(n/2) + \Theta(n)$, with solution $T(n) = \Theta(n \lg n)$.
  - The boundary conditions are usually expressed as $T(n) = O(1)$ for sufficiently small n..
  - But when there is a desire of an exact, rather than an asymptotic, solution, the need is to deal with boundary conditions.
  - In practice, just use asymptotics most of the time, and ignore boundary conditions.

Recursive Function

- Example

$A(n)$

$\{$

$If(n > 1)$

$Return\left(A\left(\frac{n}{2}\right)\right)$

$\}$

The relation is called recurrence relation

The Recurrence relation of given function is written as follows.

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

Recursive Function

- To solve the Recurrence relation the following methods are used:

1. Iteration method
2. **Recursion-Tree method**
3. Master Method
4. Substitution Method

Recursion Tree Method

- Recursion Tree is another method for solving recurrence relations. This method work on two steps. These are
  - First, A set of pre level costs are obtained by sum the cost of each level of the tree and the height of the tree.
  - Second, to determine the total cost of all level of recursion, we sum all the pre level cost.
- This method is best used for good guess.
- For generating good guess, we can ignore floors($\lfloor \quad \rfloor$) and ceiling ($\lceil \quad \rceil$) when solving the recurrences. Because, they usually do not affect the final guess.

Recursion Tree Method (Example 1)

Example 1

Solve the recurrence $T(n) = 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \Theta(n^2)$ by using recursion tree method.

Ans :

We start by focusing on finding an upper bound for the solution by using good guess. As we know that floors and ceilings usually do not matter when solving the recurrences, we drop the floor and write the recurrence equation as follows:

$$T(n) = 3T\left(\frac{n}{4}\right) + cn^2 \quad , c > 0$$

The term cn^2, at the root represent the costs incurred by the subproblems of size $\frac{n}{4}$.

Recursion Tree Method (Example 1)

$$T(n) = 3T\left(\frac{n}{4}\right) + cn^2 \quad , c > 0 \; is \; a \; constant$$

$T(n)$

Fig –(a)

Figure (a) shows T(n), which progressively expands in (b) and (d) to form recursion tree.

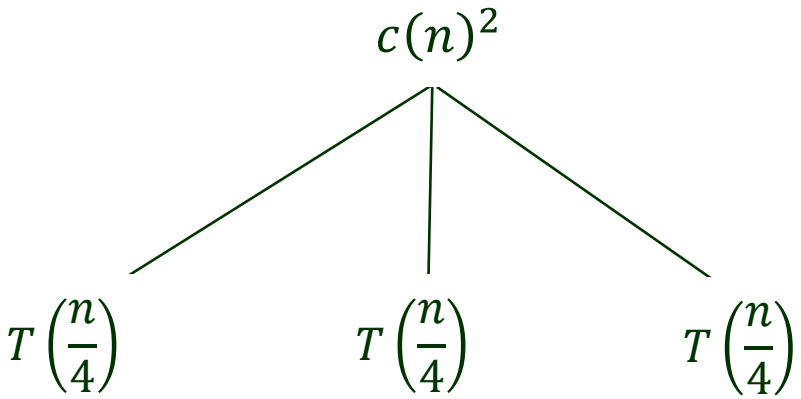

Fig –(b)

Recursion Tree Method (Example 1)

$$T(n) = 3T\left(\frac{n}{4}\right) + cn^2 \quad , c > 0 \; is \; a \; constant$$

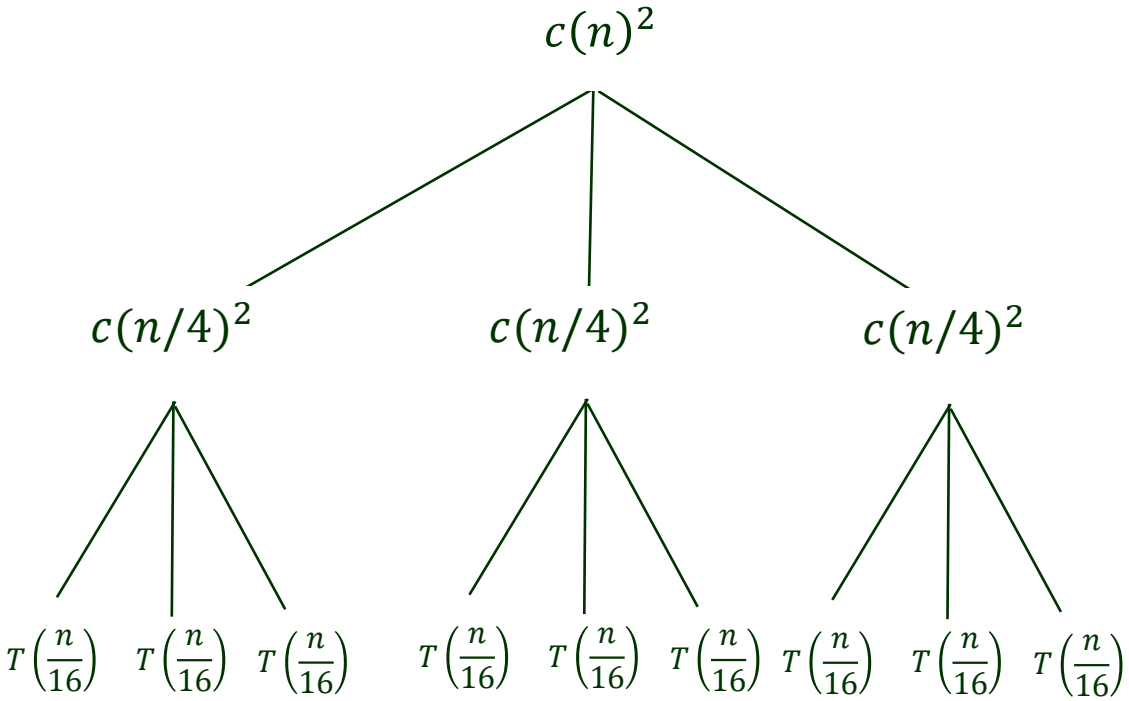

Fig –(c)

Recursion Tree Method (Example 1)

$$T(n) = 3T\left(\frac{n}{4}\right) + cn^2 \quad , c > 0 \text{ is a constant}$$

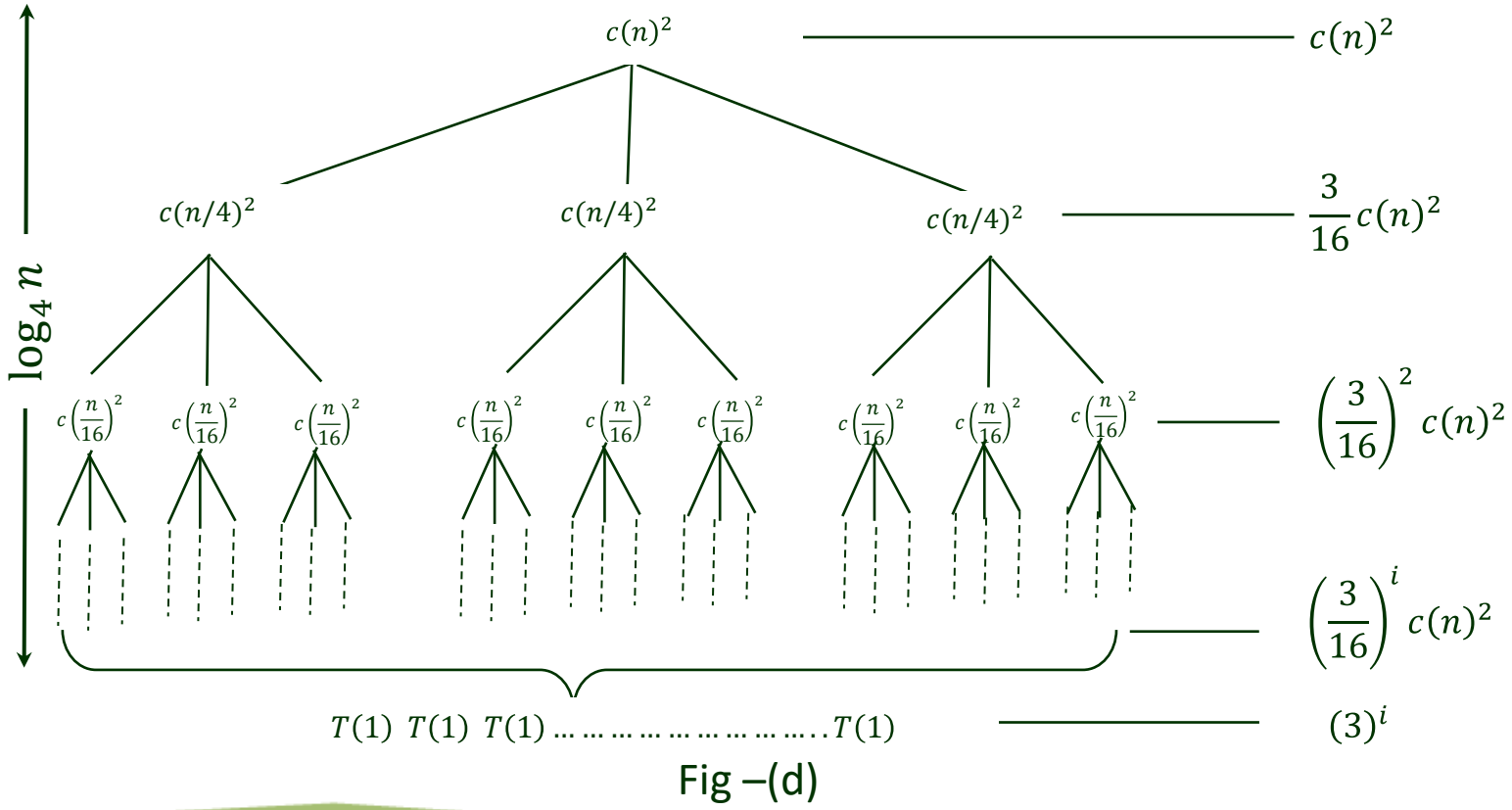

Fig –(d)

Recursion Tree Method (Example 1)

Analysis

First, we find the height of the recursion tree

Observe that a node at depth 'i' reflects a subproblem of size $\frac{n}{(4)^i}$.

i.e. the subproblem size hits $n = 1$, when $\frac{n}{(4)^i} = 1$

So, if $\frac{n}{(4)^i} = 1$

$\Longrightarrow n = (4)^i$ $\quad (Apply\ Log\ both\ side)$

$\Longrightarrow \log n = \log(4)^i$

$\Longrightarrow i = \log_4 n$

So the height of the tree is $\log_4 n$.

Recursion Tree Method (Example 1)

Second, we determine the cost of each level of the tree.
The number of nodes at depth $'i'$ is $(3)^i$.

So, each node at depth $'i'(i.e.\, i = 0,1,2,3,4, \ldots, \log_4 n - 1)$ has cost $c\left(\frac{n}{4^i}\right)^2$.

Hence the total cost at level $'i'$ is $3^i c\left(\frac{n}{4^i}\right)^2$

$$\Rightarrow 3^i . c . \left(\frac{n}{4^i}\right)^2$$

$$\Rightarrow 3^i . c . \frac{n^2}{16^i}$$

$$\Rightarrow \frac{3^i}{16^i} . c . n^2$$

$$\Rightarrow \left(\frac{3}{16}\right)^i . c . n^2$$

Recursion Tree Method (Example 1)

However, the bottom level is special. Each of the bottom node has contribute cost $= T(1)$

Hence the cost of the bottom level is $= 3^i$

$$\Longrightarrow 3^{\log_4 n} \qquad (as\ i = \log_4 n)\ the\ height\ of\ the\ tree$$

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

So, the total cost of entire tree is

$T(n)$

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

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

Recursion Tree Method (Example 1)

The left term is just a sum of Geometric series. So the value of $T(n)$ is as follows.

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

The above equation looks very complicated. So, we use an infinite geometric series as an upper bound. Hence the new form of the equation is given below:

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

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

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

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

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

Recursion Tree Method (Example 2)

Example 2

Solve the recurrence $T(n) = 4T\left(\frac{n}{2}\right) + n$ by using recursion tree method.

$$T(n) = 4T\left(\frac{n}{2}\right) + cn \quad , c > 0$$

The term cn, at the root represent the costs incurred by the subproblems of size $\frac{n}{2}$.

Construction of Recursion tree

$$T(n)$$

Fig –(a)

Figure (a) shows T(n), which progressively expands in (b) to form recursion tree.

Recursion Tree Method (Example 2)

$$T(n) = 4T\left(\frac{n}{2}\right) + cn \quad , c > 0$$

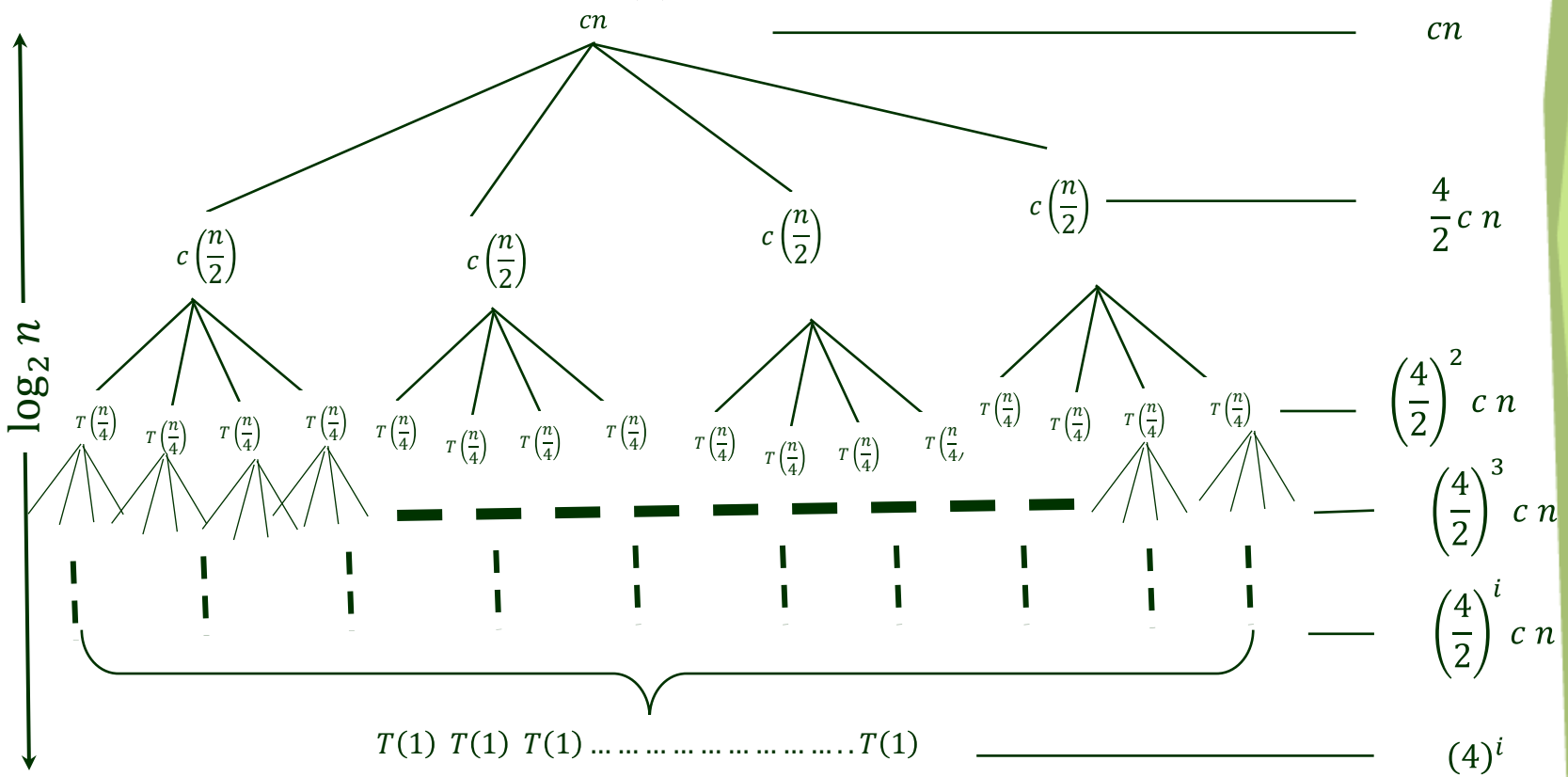

Fig –(b)

Recursion Tree Method (Example 2)

Analysis

First, we find the height of the recursion tree

Observe that a node at depth 'i' reflects a subproblem of size $\frac{n}{(2)^i}$.

i.e. the subproblem size hits $n = 1$, when $\frac{n}{(2)^i} = 1$

So, if $\frac{n}{(2)^i} = 1$

$\Longrightarrow n = (2)^i$ $\quad (Apply\ Log\ both\ side)$

$\Longrightarrow \log n = \log(2)^i$

$\Longrightarrow i = \log_2 n$

So the height of the tree is $\log_2 n$.

Recursion Tree Method (Example 2)

Second, we determine the cost of each level of the tree.
The number of nodes at depth $'i'$ is $(4)^i$.
So, each node at depth $'i'(i.e.\, i = 0,1,2,3,4, \ldots\ldots, \log_2 n - 1)$ has cost $c\frac{n}{2^i}$.
Hence the total cost at level $'i'$ is $4^i c\frac{n}{2^i}$

$$\Rightarrow 4^i . c . \frac{n}{2^i}$$

$$\Rightarrow 4^i . c . \frac{n}{2^i}$$

$$\Rightarrow \frac{4^i}{2^i} . cn$$

$$\Rightarrow \left(\frac{4}{2}\right)^i . cn$$

Recursion Tree Method (Example 2)

However, the bottom level is special. Each of the bottom node has contribute cost $= T(1)$

Hence the cost of the bottom level is $= 4^i$

$\Longrightarrow 4^{\log_2 n}$ $\quad (as\ i = \log_2 n)\ the\ height\ of\ the\ tree$

$\Longrightarrow n^{\log_2 4}$

So, the total cost of entire tree is

$$T(n) = cn + \frac{4}{2}cn + \left(\frac{4}{2}\right)^2 cn + \left(\frac{4}{2}\right)^3 cn + \cdots + \cdots + \left(\frac{4}{2}\right)^i cn + \Theta(n^{\log_2 4})$$

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

Recursion Tree Method (Example 2)

The left term is just a sum of Geometric series. So the value of $T(n)$ is as follows.

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

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

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

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

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

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

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

Recursion Tree Method (Example 3)

Example 3

Solve the recurrence $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$ by using recursion tree method.

$$T(n) = 2T(n/2) + cn \quad , c > 0$$

The term cn, at the root represent the costs incurred by the subproblems of size $\frac{n}{2}$.

Construction of Recursion tree

$$T(n)$$

Fig –(a)

Figure (a) shows T(n), which progressively expands in (b) to form recursion tree.

Recursion Tree Method (Example 3)

$$T(n) = 2T(n/2) + cn \quad , c > 0$$

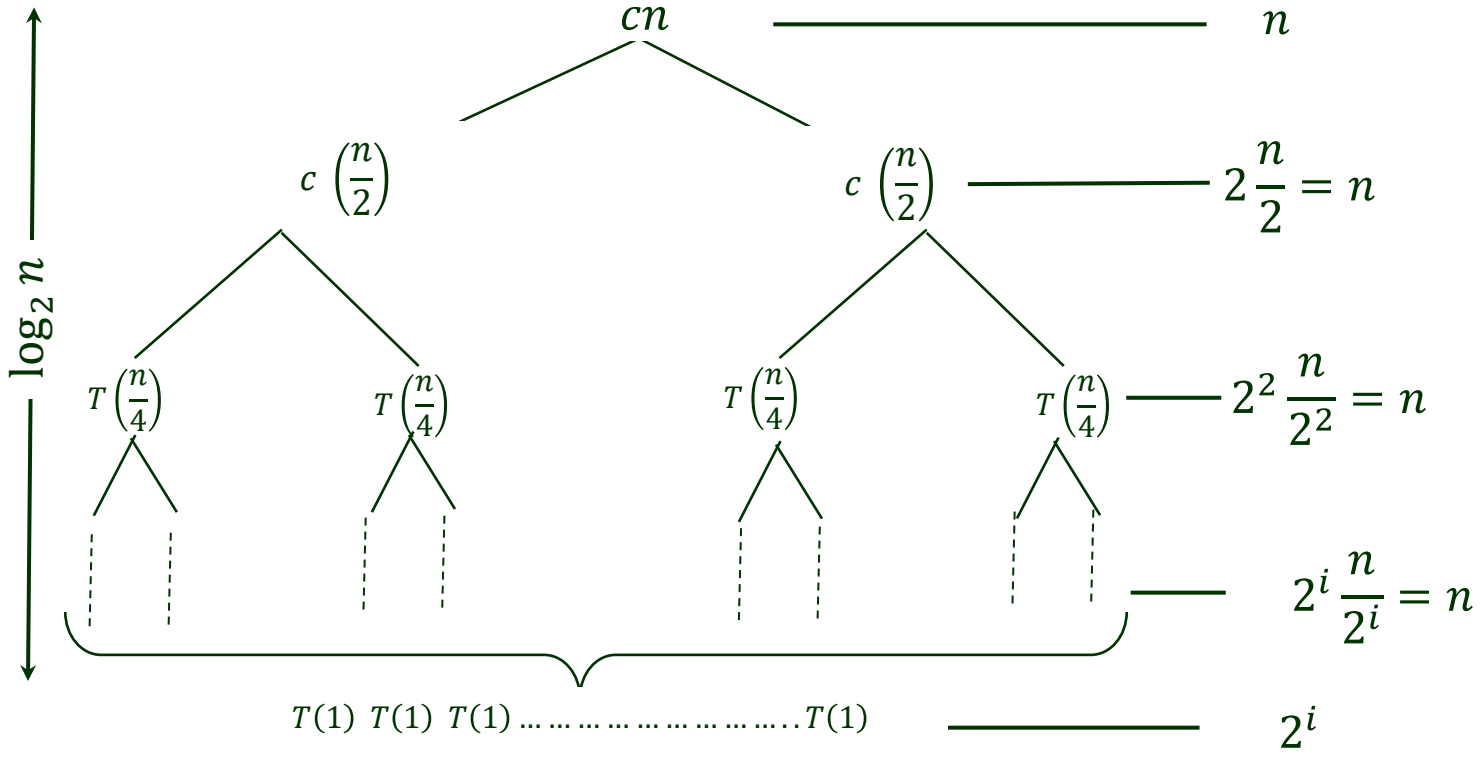

Fig –(b)

Recursion Tree Method (Example 3)

Analysis

First, we find the height of the recursion tree

Observe that a node at depth 'i' reflects a subproblem of size $\frac{n}{(2)^i}$.

i.e. the subproblem size hits $n = 1$, when $\frac{n}{(2)^i} = 1$

So, if $\frac{n}{(2)^i} = 1$

$\Longrightarrow n = (2)^i \qquad (Apply\ Log\ both\ side)$

$\Longrightarrow \log n = \log(2)^i$

$\Longrightarrow i = \log_2 n$

So the height of the tree is $\log_2 n$.

Recursion Tree Method (Example 3)

Second, we determine the cost of each level of the tree.
The number of nodes at depth $'i'$ is $(4)^i$.

So, each node at depth $'i'(i.e.\, i = 0,1,2,3,4, \ldots\ldots, \log_2 n - 1)$ has cost $c\left(\frac{n}{2^i}\right)$

.

Hence the total cost at level $'i'$ is $2^i c\left(\frac{n}{2^i}\right) = cn$

However, the bottom level is special. Each of the bottom node has contribute cost $= T(1)$

Hence the cost of the bottom level is $= 2^i$

$\Longrightarrow 2^{\log_2 n}$ $\qquad (as\ i = \log_2 n)\ the\ height\ of\ the\ tree$

$\Longrightarrow n^{\log_2 2}$

$\Longrightarrow n$

Recursion Tree Method (Example 3)

So, the total cost of entire tree is

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

$$T(n) = cn \sum_{i=0}^{\log_2 n} 1^i .$$

$$T(n) = cn\,(\log_2 n + 1)$$

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

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

Recursion Tree Method (Example 4)

Example 4

Solve the recurrence $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + \Theta(n)$ by using recursion tree method.

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + cn \quad , c > 0$$

The term cn, at the root represent the costs incurred by the subproblems of size $\frac{n}{2}$, $\frac{n}{4}$, and $\frac{n}{8}$.

Construction of Recursion tree

$T(n)$

Fig –(a)

Figure (a) shows T(n), which progressively expands in (b) to form recursion tree.

Recursion Tree Method (Example 4)

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + cn \quad , c > 0$$

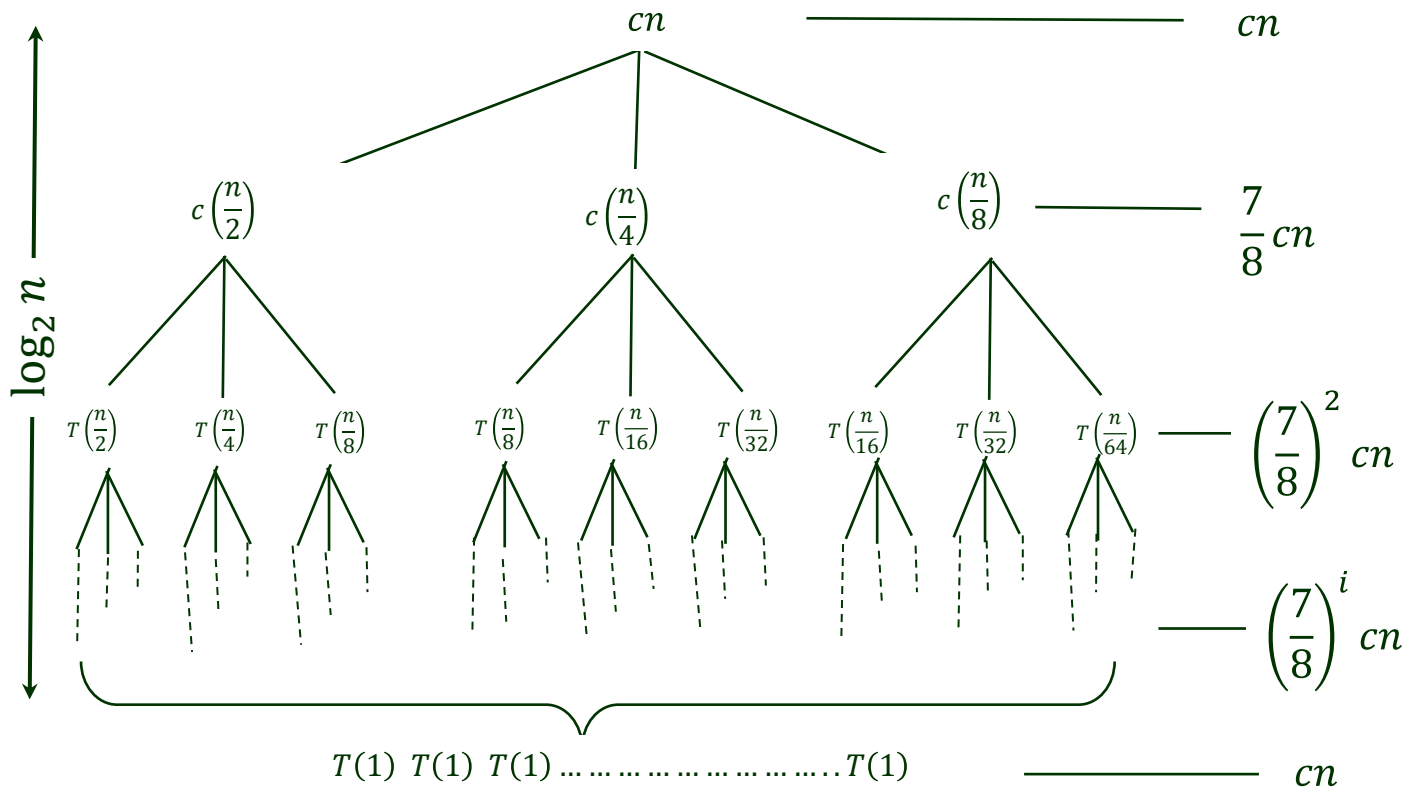

Fig –(b)

Recursion Tree Method (Example 4)

Analysis

First, we find the height of the recursion tree

Hear the problem divide into three subproblem of size $\frac{n}{2^i}$, $\frac{n}{4^i}$, and $\frac{n}{8^i}$.

For calculating the height of the tree, we consider the longest path of the tree. It has been observed that the node on the left-hand side is the longest path of the tree.

Hence the node at depth $'i'$ reflects a subproblem of size $\frac{n}{2^i}$.

i.e. the subproblem size hits $n = 1$, when $\frac{n}{2^i} = 1$

So, if $\frac{n}{2^i} = 1$

$\Longrightarrow n = 2^i$ $\quad (Apply\ Log\ both\ side)$

$\Longrightarrow \log n = \log(2)^i$

$\Longrightarrow i = \log_2 n$

So the height of the tree is $\log_2 n$.

Recursion Tree Method (Example 4)

Second, we determine the cost of the tree in level $'i' = \left(\frac{n}{2^i} + \frac{n}{4^i} + \frac{n}{8^i}\right)$

So, the total cost of the tree is:

$$T(n) = cn + \frac{7}{8}cn + \left(\frac{7}{8}\right)^2 cn + \left(\frac{7}{8}\right)^3 cn + \cdots + \cdots$$

For simplicity we take ∞ Geometric Series

$$T(n) = cn + \frac{7}{8}cn + \left(\frac{7}{8}\right)^2 cn + \left(\frac{7}{8}\right)^3 cn + \cdots + \infty$$

$$T(n) \leq \sum_{i=0}^{\infty} \left(\frac{7}{8}\right)^i cn$$

$$T(n) \leq cn \left[\frac{1}{1 - \frac{7}{8}}\right]$$

$$T(n) \leq cn \left[\frac{8}{1}\right]$$

$$T(n) \leq 8cn$$

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


Thank U