How to Prepare a recurrence Relation :⇒

Example Code.

```
void funct (int n) ──────────→ T(n)
{
    if (n>0)
    {
        printf ("%d", n); ──→ 1
        funct (n-1); ──────→ T(n-1)
    }
}
```

Recursive calls

$$:\Rightarrow T(n) = T(n-1) + 1 .$$

cost per call.

Recurrence Relation for above code :⇓

$$T(n) = \begin{cases} 1 & n=0 \\ T(n-1)+1 & n>0 \end{cases}$$

Solution to Recurrence Relation

i) Substitution Method

ii) Recurssion Tree Method.

iii) Master Method.

i) Solving Recurrence Relation using Substitution Method :->

Ex.

$$T(n) = \begin{cases} 1 & n = 0 \\ T(n-1)+1 & n > 0 \end{cases}$$ ⇐ (*) Recurrence Relation (Example).

$T(n) = T(n-1)+1$ — Eq. (1)

$\because T(n) = T(n-1)+1$

$T(n-1) = T([n-1]-1)+1$

$\Rightarrow T(n-2)+1.$

Substitute $T(n-1)$ in Eq. (1)

$T(n) = [T(n-2)+1]+1$

$= T(n-2)+2$ — Eq (2)

$T(n-2) = T([n-2]-1)+1$

$\Rightarrow T(n-3)+1.$

Substitute $T(n-2)$ in Eq. (2)

$T(n) = [T(n-3)+1]+2.$

$= T(n-3)+3$

⋮

Continue the process k times, we get

$T(n) = T(n-k)+k$ — Eq (x)

Assume $n-k = 0$ $\therefore n = k$ → putting in Eq. (x)

$\because$ at the end $n=0$ $\therefore$ from Eq (x) $n-k=0$.

$\therefore T(n) = T(n-n)+n.$

$= T(0)+n.$

From Recurrence relation (*) $T(0) = 1$

$= 1+n$

$\Rightarrow O(n)$

Recursion Tree-

- Recursion Tree is another method for solving the recurrence relations.
- A recursion tree is a tree where each node represents the cost of a certain recursive sub-problem.
- We sum up the values in each node to get the cost of the entire algorithm.

Steps to Solve Recurrence Relations Using Recursion Tree Method-

Step-01:

Draw a recursion tree based on the given recurrence relation.

Step-02:

Determine-

- Cost of each level
- Total number of levels in the recursion tree
- Number of nodes in the last level
- Cost of the last level

Step-03:

Add cost of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation.

Following problems clearly illustrates how to apply these steps.

PRACTICE PROBLEMS BASED ON RECURSION TREE-

Problem-01:

Solve the following recurrence relation using recursion tree method-

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

Solution-

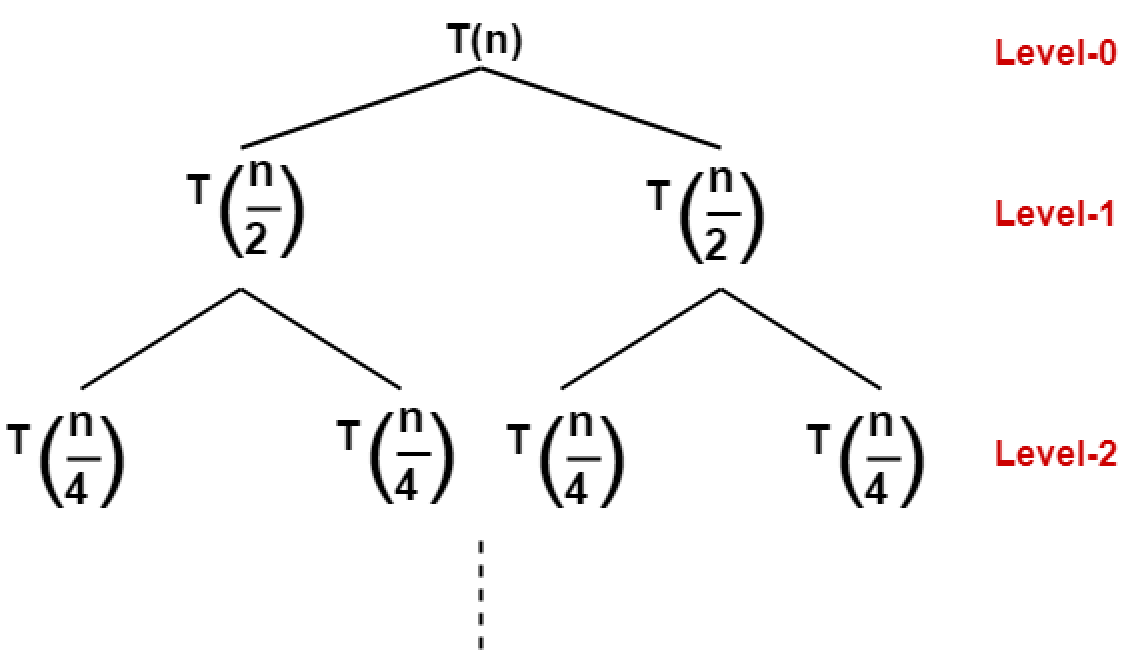
Step-01:

Draw a recursion tree based on the given recurrence relation.

The given recurrence relation shows-

- A problem of size n will get divided into 2 sub-problems of size n/2.
- Then, each sub-problem of size n/2 will get divided into 2 sub-problems of size n/4 and so on.
- At the bottom most layer, the size of sub-problems will reduce to 1.

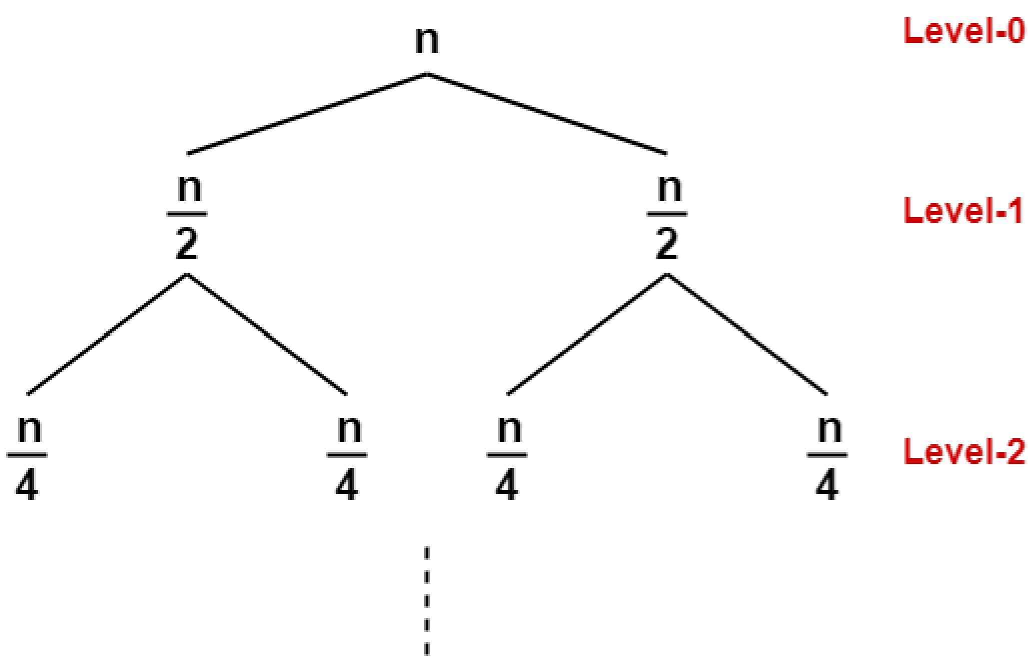
This is illustrated through following recursion tree-



The given recurrence relation shows-

- The cost of dividing a problem of size n into its 2 sub-problems and then combining its solution is n.
- The cost of dividing a problem of size n/2 into its 2 sub-problems and then combining its solution is n/2 and so on.

This is illustrated through following recursion tree where each node represents the cost of the corresponding sub-problem-



Step-02:

Determine cost of each level-

- Cost of level-0 = n
- Cost of level-1 = n/2 + n/2 = n
- Cost of level-2 = n/4 + n/4 + n/4 + n/4 = n and so on.

Step-03:

Determine total number of levels in the recursion tree-

- Size of sub-problem at level-0 = $n/2^0$
- Size of sub-problem at level-1 = $n/2^1$
- Size of sub-problem at level-2 = $n/2^2$

Continuing in similar manner, we have-

$$\text{Size of sub-problem at level-i} = n/2^i$$

Suppose at level-x (last level), size of sub-problem becomes 1. Then-

$$n / 2^x = 1$$

$$2^x = n$$

Taking log on both sides, we get-

$$x\log 2 = \log n$$

$$x = \log_2 n$$

∴ Total number of levels in the recursion tree = $\log_2 n + 1$

Step-04:

Determine number of nodes in the last level-

- Level-0 has 2^0 nodes i.e. 1 node
- Level-1 has 2^1 nodes i.e. 2 nodes
- Level-2 has 2^2 nodes i.e. 4 nodes

Continuing in similar manner, we have-

$$\text{Level-}\log_2 n \text{ has } 2^{\log_2 n} \text{ nodes i.e. } n \text{ nodes}$$

Step-05:

Determine cost of last level-

$$\text{Cost of last level} = n \times T(1) = \theta(n)$$

Step-06:

Add costs of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation-

$$T(n) = \underbrace{\{\, n + n + n + ... \,\}}_{\text{For } \log_2 n \text{ levels}} + \theta\ (n)$$

$= n \times \log_2 n + \theta\ (n)$

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

= **$\theta\ (n\log_2 n)$**

Problem-02:

Solve the following recurrence relation using recursion tree method-

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

Solution-

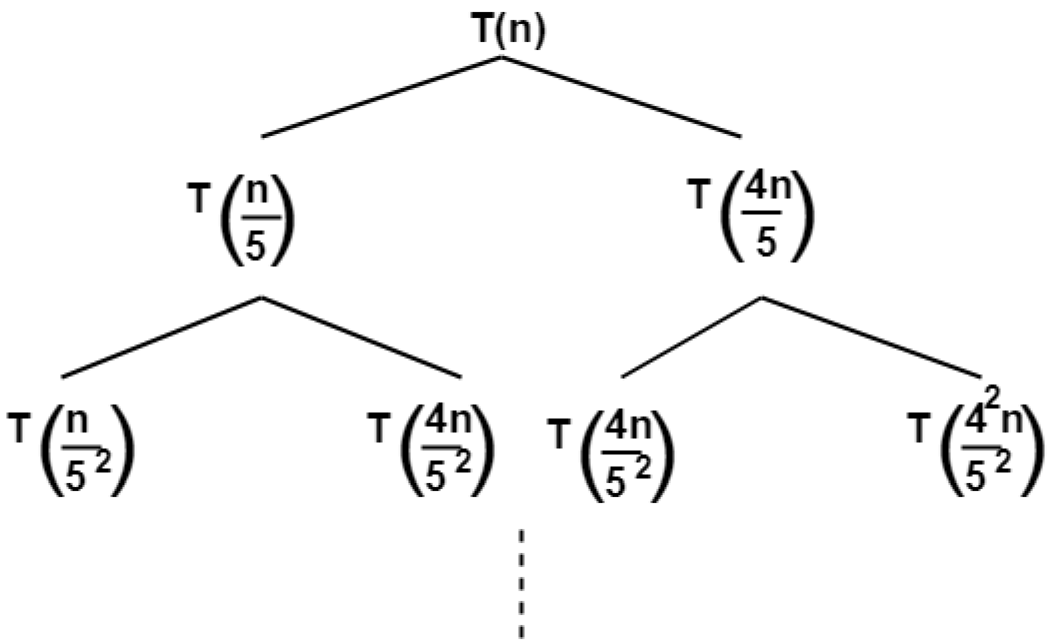
Step-01:

Draw a recursion tree based on the given recurrence relation.

The given recurrence relation shows-

- A problem of size n will get divided into 2 sub-problems- one of size n/5 and another of size 4n/5.
- Then, sub-problem of size n/5 will get divided into 2 sub-problems- one of size $n/5^2$ and another of size $4n/5^2$.
- On the other side, sub-problem of size 4n/5 will get divided into 2 sub-problems- one of size $4n/5^2$ and another of size $4^2n/5^2$ and so on.
- At the bottom most layer, the size of sub-problems will reduce to 1.

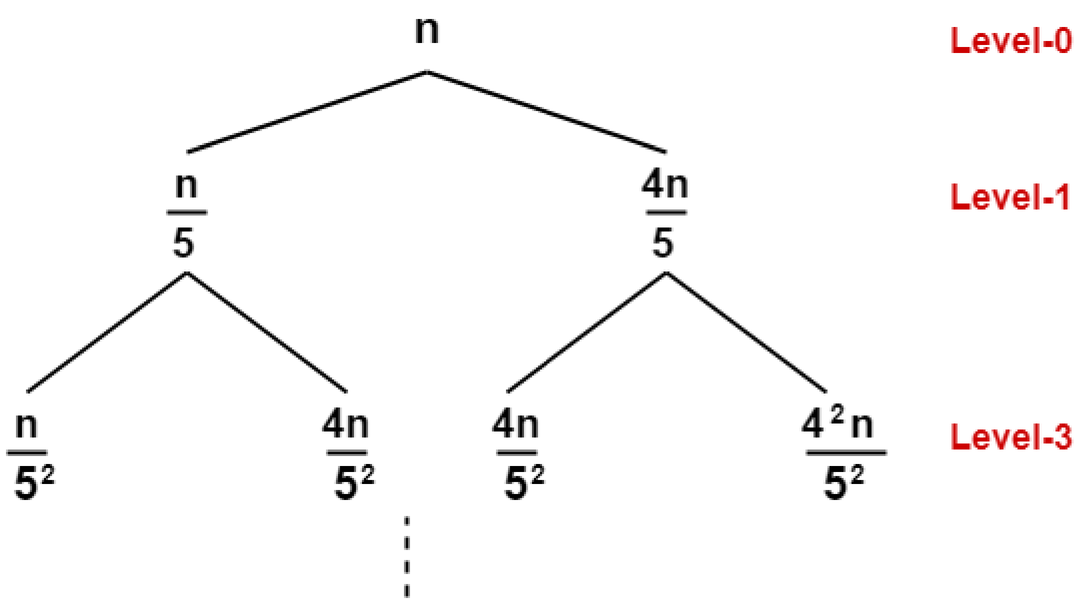
This is illustrated through following recursion tree-



The given recurrence relation shows-

- The cost of dividing a problem of size n into its 2 sub-problems and then combining its solution is n.
- The cost of dividing a problem of size n/5 into its 2 sub-problems and then combining its solution is n/5.
- The cost of dividing a problem of size 4n/5 into its 2 sub-problems and then combining its solution is 4n/5 and so on.

This is illustrated through following recursion tree where each node represents the cost of the corresponding sub-problem-



Step-02:

Determine cost of each level-

- Cost of level-0 = n
- Cost of level-1 = n/5 + 4n/5 = n
- Cost of level-2 = $n/5^2 + 4n/5^2 + 4n/5^2 + 4^2n/5^2 = n$

Step-03:

Determine total number of levels in the recursion tree. We will consider the rightmost sub tree as it goes down to the deepest level-

- Size of sub-problem at level-0 = $(4/5)^0n$
- Size of sub-problem at level-1 =$(4/5)^1n$
- Size of sub-problem at level-2 =$(4/5)^2n$

Continuing in similar manner, we have-

$$\text{Size of sub-problem at level-i} = (4/5)^i n$$

Suppose at level-x (last level), size of sub-problem becomes 1. Then-

$$(4/5)^x n = 1$$

$$(4/5)^x = 1/n$$

Taking log on both sides, we get-

$$x\log(4/5) = \log(1/n)$$

$$x = \log_{5/4}n$$

∴ Total number of levels in the recursion tree = $\log_{5/4}n + 1$

Step-04:

Determine number of nodes in the last level-

- Level-0 has 2^0 nodes i.e. 1 node
- Level-1 has 2^1 nodes i.e. 2 nodes
- Level-2 has 2^2 nodes i.e. 4 nodes

Continuing in similar manner, we have-

$$\text{Level-}\log_{5/4}n \text{ has } 2^{\log_{5/4}n} \text{ nodes}$$

Step-05:

Determine cost of last level-

$$\text{Cost of last level} = 2^{\log_{5/4}n} \times T(1) = \theta(2^{\log_{5/4}n}) = \theta(n^{\log_{5/4}2})$$

Step-06:

Add costs of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation-

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

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

$\mathbf{= \theta(n\log_{5/4}n)}$

Problem-03:

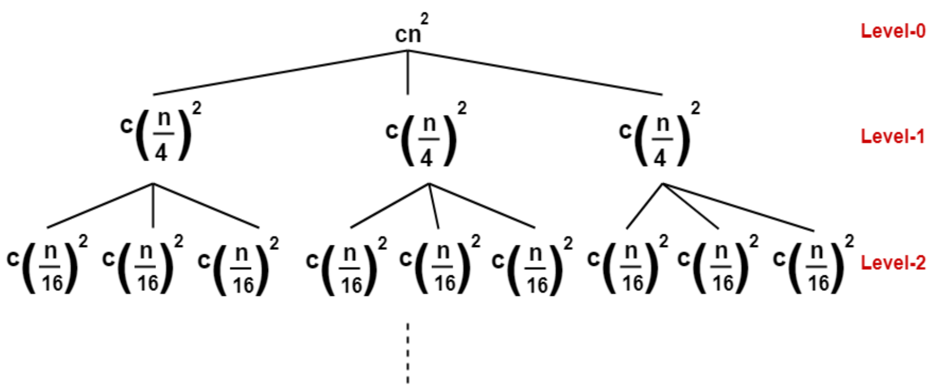
Solve the following recurrence relation using recursion tree method-

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

Solution-

Step-01:

Draw a recursion tree based on the given recurrence relation-



(Here, we have directly drawn a recursion tree representing the cost of sub problems)

Step-02:

Determine cost of each level-

- Cost of level-0 = cn^2
- Cost of level-1 = $c(n/4)^2 + c(n/4)^2 + c(n/4)^2 = (3/16)cn^2$
- Cost of level-2 = $c(n/16)^2 \times 9 = (9/16^2)cn^2$

Step-03:

Determine total number of levels in the recursion tree-

- Size of sub-problem at level-0 = $n/4^0$
- Size of sub-problem at level-1 = $n/4^1$
- Size of sub-problem at level-2 = $n/4^2$

Continuing in similar manner, we have-

$$\text{Size of sub-problem at level-i} = n/4^i$$

Suppose at level-x (last level), size of sub-problem becomes 1. Then-

$$n/4^x = 1$$

$$4^x = n$$

Taking log on both sides, we get-

$$x\log 4 = \log n$$

$$x = \log_4 n$$

∴ Total number of levels in the recursion tree = $\log_4 n + 1$

Step-04:

Determine number of nodes in the last level-

- Level-0 has 3^0 nodes i.e. 1 node
- Level-1 has 3^1 nodes i.e. 3 nodes
- Level-2 has 3^2 nodes i.e. 9 nodes

Continuing in similar manner, we have-

$$\text{Level-}\log_4 n \text{ has } 3^{\log_4 n} \text{ nodes i.e. } n^{\log_4 3} \text{ nodes}$$

Step-05:

Determine cost of last level-

$$\text{Cost of last level} = n^{\log_4 3} \times T(1) = \theta(n^{\log_4 3})$$

Step-06:

Add costs of all the levels of the recursion tree and simplify the expression so obtained in terms of asymptotic notation-

$$T(n) = \underbrace{\left\{ cn^2 + \frac{3\ cn^2}{16} + \frac{9\ cn^2}{(16)^2} + \cdots\cdots\cdots \right\}}_{\text{For } \log_4 n \text{ levels}} + \theta\left(n^{\log_4 3}\right)$$

$= cn^2 \{ 1 + (3/16) + (3/16)^2 + \ldots\ldots\ldots \} + \theta(n^{\log_4 3})$

Now, $\{ 1 + (3/16) + (3/16)^2 + \ldots\ldots\ldots \}$ forms an infinite Geometric progression.

On solving, we get-

$= (16/13)cn^2 \{ 1 - (3/16)^{\log_4 n} \} + \theta(n^{\log_4 3})$

$= (16/13)cn^2 - (16/13)cn^2 (3/16)^{\log_4 n} + \theta(n^{\log_4 3})$

= $\mathbf{O(n^2)}$