Recurrences

Recursive Function - function calling itself.

Recurrence - is an equation or inequality that describes a function in terms of its value on smaller inputs.

Example to understand how recurrence is created from algorithm?

```
Algorithm ABC(n)
{
    if (n>1)
        return (ABC(n-1))
}
```

$$T(n) = 1 + T(n-1) \quad \text{if } n > 1$$
$$= O(1) \quad n = 1$$

There are three methods to solve recurrences :-

1) The Substitution Method
2) The Recursion Tree Method
3) The Master Method.

Scanned by CamScanner

[I] The Substitution Method

There are two steps
(i) Guess the form of the solution
(ii) Use mathematical induction to find constant & show that solution works.

Examples

1] $T(n) = 2T(n/2) + n$ for $n > 1$ and $T(1) = 1$
Determine upper bound.

Soln → Step 1: Guess the Solution
$T(n) = O(n \log n)$

Step 2: Mathematical Induction

(a) To prove: $T(n) \leq cn\log n$ ... (I) for some constant c

(b) Finding constant c:

Let us assume that bound holds for $T(n/2)$.

$\therefore T(n/2) \leq c(n/2)\log(n/2)$ ..... (II)

Substituting equation II in recurrence relation.

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

Scanned by CamScanner

$$T(n) = 2\left[c\, n/2 \log(n/2)\right] + n$$

$$= cn \log(n/2) + n$$

$$= cn\log n - cn\log 2 + n$$

$$T(n) = cn\log n - cn + n \quad \ldots \text{III}$$

We need to prove,

$$\text{e.g. } T(n) \leq cn\log n$$

$$cn\log n - cn + n \leq cn\log n$$

$$-cn + n \leq 0$$

Dividing by n,

$$-c + 1 \leq 0$$

$$\boxed{c \geq 1}$$

(c) Proof the base condition:

Suppose, put $n = 1$ in eqn I.

$$T(1) \leq c(1)\log(1) \leq 0$$

But $T(1) = 1$. Hence Not satisfied.

(d) Revise the induction

Need to find lowest value of n_0 for which relation is satisfied.

Consider, $n_0 = 2$

Scanned by CamScanner

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

$= 2T(1) + 2$

$\boxed{T(2) = 4}$

Now, for relation.

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

$T(2) \leq c(2\log 2)$

$T(2) \leq 2c$

For sufficient large value of c, $c \geq 2$ the condition holds the relation.

$\therefore T(n) = O(n\log n)$ where $n_0 = 2$

$c = 2$

For, $n_0 = 3$,

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

$= 2T(1) + 3$

$= 5$

for relation,

$T(n) \leq cn\log n.$

$T(3) \leq 2 \times 3 \log 3.$

$5 \leq 2 \times 3 (1.5)$

$5 \leq 9$

$\therefore$ satisfied, $\therefore T(n) = O(n\log n)$ $n_0 = 2$

$c = 2$

Scanned by CamScanner

2] $T(1) = 1$ and $T(n) = T(n/2) + 1$ for $n > 1$

determine the upper bound

Solution: Step 1: Guess the solution.

$T(n) = O(\log n)$

Step 2: Mathematical Induction

(a) To prove: $T(n) \leq c \log n$ for some constant c ——— (I)

(b) Finding constant c:

Let us assume that bound holds for $T(n/2)$

$\therefore T(n/2) \leq c \log(n/2)$ ..... (II)

using equation (II) in recurrence equation,

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

$= c \log(n/2) + 1$

$= c \log n - c + 1$ ——— (III).

We need to prove,

$T(n) \leq c \log n$

$c \log n - c + 1 \leq c \log n.$

$-c + 1 \leq 0$

$\boxed{c \geq 1}$

Scanned by CamScanner

(c) ~~Proof the~~ Prove the base condition :

Suppose, put $n=1$ on equation I

$$T(1) \leq c \log 1 \leq 0$$

But $T(1) = 1$, Hence Not satisfied.

(d) Revise the Induction

Need to find the lowest value of n_0 for which relation is satisfied.

Consider $n_0 = 2$

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

$$= 2$$

Now, for relation,

$$T(n) \leq c \log n$$

$$T(2) \leq c \log 2$$

$$T(2) \leq c$$

Consider $n_0 = 3$

$$T(3) = 2T\left(\frac{3}{2}\right) + 1$$

$$= T(1) + 1$$

$$= 2$$

Now for relation

$$T(n) \leq c \log 3$$

$$T(3) \leq 1.5\, c$$

For sufficient larger value of c, $c \geq 2$ the condition holds the relation.

$\therefore$ $T(n) = O(\log n)$ for $n_0 = 2$

$c = 2$

Scanned by CamScanner

3] Show that solution for $T(n) = n(n-1) + n$ is $O(n^2)$

Soln → Step 1: Guess the solution & find exact form

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

Exact form :

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

$$T(1) = 1$$

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

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

$$T(4) = 1+2+3+4$$

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

Step 2 : Mathematical Induction

(a) To prove : $T(n) \leq c\left[\frac{n(n+1)}{2}\right]$ — (I)

for some constant c

(b) Finding Constant c :

Let us assume that bound holds for $T(n-1)$.

$$\therefore T(n-1) \leq c\left[\frac{(n-1)(n)}{2}\right] \quad \cdots \text{(II)}$$

using equation II, in recurrence equation,

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

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

$$= \frac{c}{2}n^2 - \frac{c}{2}n + n$$

Scanned by CamScanner

87

We need to prove,

$$T(n) \leq c\left[\frac{n(n+1)}{2}\right]$$

$$\frac{c}{2}n^2 - \frac{c}{2}n + n \leq \frac{c}{2}n^2 + \frac{c}{2}n$$

$$n \leq cn$$

Dividing by n,

$$\boxed{c \geq 1}$$

(c) Prove the base condition;

Suppose, put $n=1$ in equation I.

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

$$\leq c\left(1\left(\frac{1+1}{2}\right)\right)$$

$$\leq c$$

$$\leq 1 \qquad [c = 1]$$

which is satisfied.

$\therefore$ $T(n) = O(n^2)$ for $c = 1$

$n_0 = 1$

Scanned by CamScanner

4] Show that solution for $T(n) = 2T(\lceil n/2 \rceil + 17) + n$ is $O(n \log n)$.

Solution:

Step 1: Guess the solution
$\therefore T(n) = O(n \log n)$

Step 2: Mathematical Induction

(a) To prove: $T(n) \leq cn \log n$ ... (I)

(b) Finding constant c:

Let us assume that the bound holds true for $T(n/2 + 17)$

$$\therefore T\left(\frac{n}{2} + 17\right) = c\left(\frac{n}{2} + 17\right) \log\left(\frac{n}{2} + 17\right) \quad \text{---- (II)}$$

using equation (II) in recurrence equation.

$$\therefore T(n) = 2\left[c\left(\frac{n}{2} + 17\right) \log\left(\frac{n}{2} + 17\right)\right] + n.$$

$$= cn \log\left(\frac{n}{2} + 17\right) + 34c \log\left(\frac{n}{2} + 17\right) + n$$

$$= cn \log\left(\frac{n+34}{2}\right) + 34c \log\left(\frac{n+34}{2}\right) + n$$

$$= cn \log(n+34) - cn + 34c \log(n+34) - 34c + n$$

$$= cn \log\left(n\left(1 + \frac{34}{n}\right)\right) + 34c \log\left(n\left(1 + \frac{34}{n}\right)\right) - c(n+34) + n$$

$$= cn \log n + cn \log\left(1 + \frac{34}{n}\right) + 34c \log n + 34c \log\left(1 + \frac{34}{n}\right) - c(n+34) + n$$

Scanned by CamScanner

b)

$$= c(n+34)\log n + c(n+34)\log\left(1+\frac{34}{n}\right) - c(n+34) + n$$

When n is very large, then $34/n$ tends to zero and $n+34 \cong n$.

$$= cn\log n + cn\log(1) - nc + n$$

$$T(n) = cn\log n - cn + n$$

We need to prove,

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

$$cn\log n - cn + n \leq cn\log n$$

Dividing by n.

$$-c + 1 \leq 0$$

$$\therefore \boxed{c \geq 1}$$

(c) Prove the base condition

Suppose put $n=1$ in eqn (I)

$$T(1) \leq c(1)\log(1) \leq 0$$

But $T(1) = 1$. Hence not satisfied.

Scanned by CamScanner

(d) Revise the Induction

For larger values of n, $n/2 + 17 \cong n/2$.

$\therefore \quad T(n) = 2T(n/2 + 17) + n$

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

We need to find lowest value of n_0 for which relation is satisfied.

consider,

$n_0 = 2$

$T(2) = 2T(1) + 2$

$= 4$

For Relation,

$T(2) \leq c\,2\log 2$

$\leq 2c$

$n_0 = 3$

$T(3) = 2T\left(\left\lceil \frac{2}{3} \right\rceil\right) + 3$

$= 2T(2) + 3$

$= 8 + 3$

$= 11$

for relation

$T(3) \leq c \cdot 3 \log 3$

$\leq 4.5\,c$

For sufficient larger value of c, $c \geq 3$ then condition holds the relation.

$\therefore \quad T(n) = \Theta(n \log n)$ for $n_0 = 2$, $c = 3$

Scanned by CamScanner

5] Solve the recurrence $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log n$ by making change of variable

Solution : 1] Changing the Variable

Let $m = \log n$ — (1)

$\therefore n = 2^m$ — (2)

$\therefore \sqrt{n} = 2^{m/2}$ — (3)

$\therefore$ substituting (1), (2) (3) in recurrence equation,

$$T(2^m) = 2T(2^{m/2}) + m$$

Now Rename, $S(m) = T(2^m)$

$$\therefore S(m) = 2S(m/2) + m \quad \text{— (4)}$$

2] Prove :

(a) Guess the Solution :

$$S(m) = O(m \log m) \quad \text{— (5)}$$

(b) Mathematical Induction

To prove : $S(m) \leq cm \log m$ — (6)

Finding c :

Let us assume bound holds for $S(m/2)$

$$\therefore S(m/2) \leq c\,m/2 \log m/2 \quad \text{— (7)}$$

Scanned by CamScanner

Substituting eqn (7) in (4)

$$S(m) = 2\left[c\frac{m}{2}\log m/_2\right] + m$$

$$= cm\log m - cm + m$$

We need to prove,

$$S(m) \leq cm\log m$$

$$cm\log m - cm + m \leq cm\log m$$

diving by m,

$$-c + 1 \leq 0$$

$$c \geq 1$$

Prove the base condition :

put $m = 1$, in equation (6)

$$S(1) \leq c(1)\log(1) = 0$$

But $S(1) = 1$, Hence not satisfied.

Revise the Induction

consider, $m_0 = 2$

$$S(2) = 2S(1) + 2$$

$$= 4$$

For Relation,

$$S(2) \leq c \cdot 2\log 2$$

$$\leq 2 \cdot c$$

Scanned by CamScanner

for sufficient larger value of c,
$c \geq 2$, relation holds.

$\therefore \quad S(m) = O(m \log m)$ for $c = 2$, $m_0 = 2$

3] Changing Back from $S(m)$ to $T(n)$

$$T(n) = T(2^m)$$
$$= S(m)$$
$$= O(m \log m)$$

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

$$n_0 = 2^{m_0}$$
$$= 2^2$$
$$\boxed{n_0 = 4}$$

$\therefore \quad T(n) = O(\log n \log \log n)$ for $c = 2$, $n_0 = 4$

Scanned by CamScanner

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

Scanned by CamScanner

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

Scanned by CamScanner

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,

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

Scanned by CamScanner

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

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

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

$$n = 2^i$$

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

$$= i$$

Scanned by CamScanner

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,

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

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

Scanned by CamScanner

$$i = \log_2 n$$

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

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

Scanned by CamScanner

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

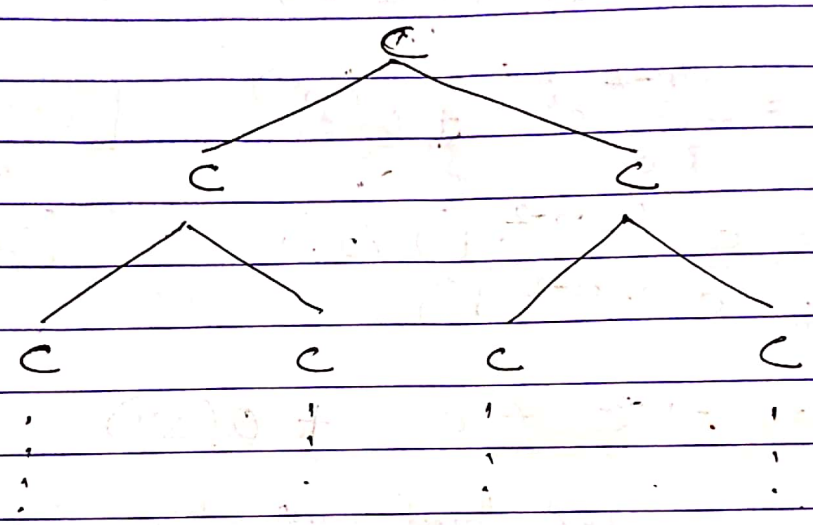
$$\therefore \quad T(n) = \Theta(n^2)$$

Scanned by CamScanner

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

$n - i = 1$

$\boxed{i = n-1}$

Scanned by CamScanner

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

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

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

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

Scanned by CamScanner

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

Scanned by CamScanner

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,

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

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

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

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

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

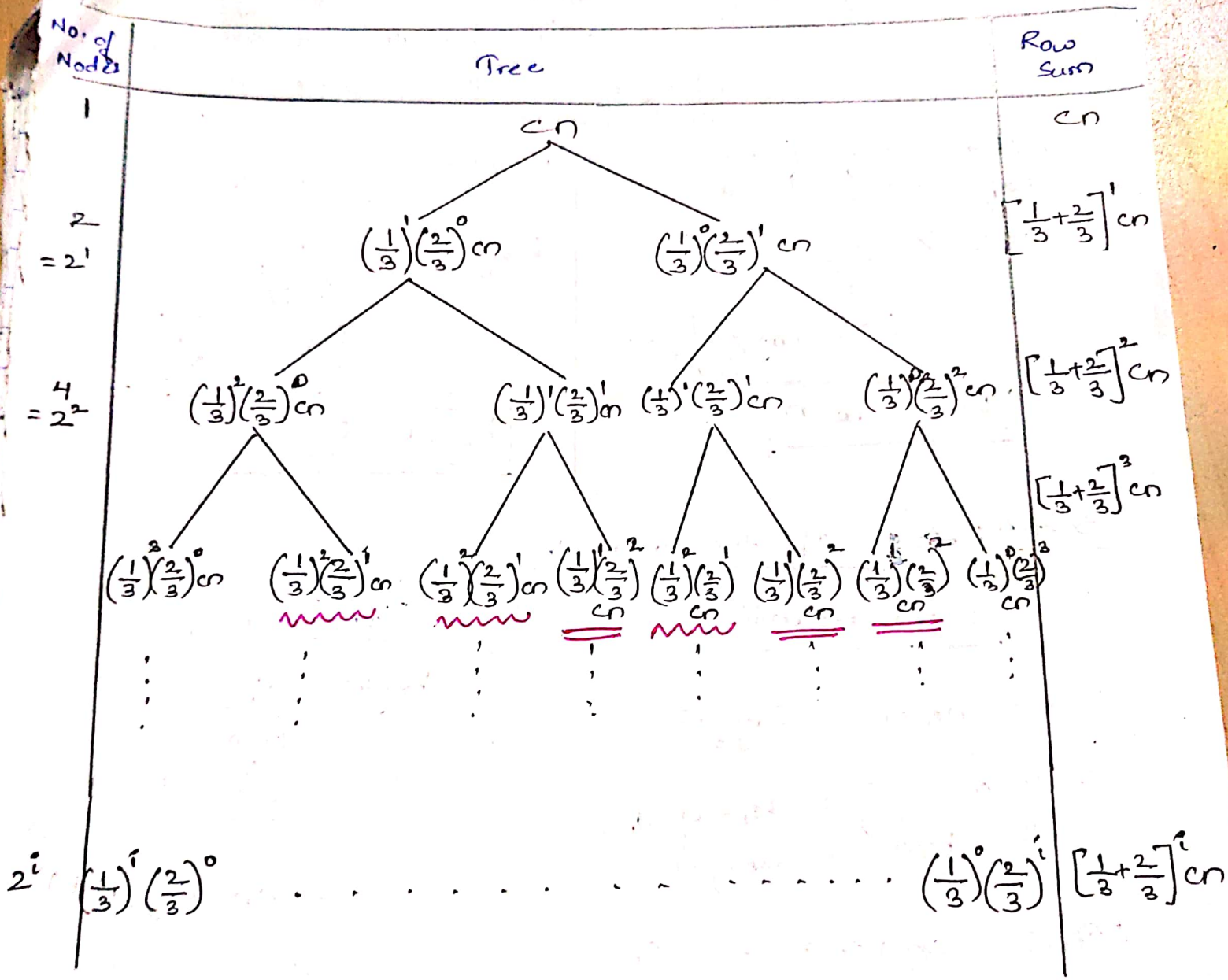
First term is higher than second term,

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

Scanned by CamScanner

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

Step 1 : Recursion Tree



Step 2 : Cost at each level = cn

Scanned by CamScanner

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

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

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

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

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

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

Scanned by CamScanner

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

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

Scanned by CamScanner

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.

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

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

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

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

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

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

Scanned by CamScanner

THE MASTER METHOD

EXPERIMENT : No. (FOR SOLVING RECURRENCES)

PAGE No.
DATE

- The master method is used for solving recurrences of the form :

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \quad \text{— (I)}$$

where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

- The recurrence in equation (1) describes the running time of an algorithm that divides a problem of size n into a subproblems, each of size n/b, where a and b are positive constants.

- function $f(n)$ includes cost of dividing the problem and combining the results of subproblem.

MASTER THEOREM

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be ~~defined~~ a recurrence defined as,

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

then $T(n)$ has the following asymptotic bounds

Sundaram

Teacher's Sign.: _____

Scanned by CamScanner

Case 1:

If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$

i.e $f(n) < n^{\log_b a}$

then,

$$T(n) = \Theta(n^{\log_b a})$$

Case 2:

If $f(n) = \Theta(n^{\log_b a})$

i.e $f(n) = n^{\log_b a}$

then,

$$T(n) = \Theta(n^{\log_b a} \log n)$$

Case 3:

i) If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$,

i.e $f(n) > n^{\log_b a}$

and

ii) if $a f(n/b) \leq c f(n)$ for some constant $c < 1$

then,

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

Scanned by CamScanner

EXPERIMENT : No.

PAGE No.
DATE

EXAMPLES

1. $T(n) = 9T\left(\frac{n}{3}\right) + n$

Step 1 : Comparing with standard form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$\therefore a = 9 \quad b = 3 \quad f(n) = n$

Step 2 : Calculate $n^{\log_b a}$

$$n^{\log_b a} = n^{\log_3 9} = n^2$$

Step 3 : Identifying the case

$f(n) = n \qquad n^{\log_b a} = n^2$

$\therefore \boxed{f(n) < n^{\log_b a}}$

i.e $f(n) = O(n^{2-\epsilon})$ where $\epsilon = 1$

By case 1 of master theorem,

$$T(n) = \Theta(n^{\log_b a})$$

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

Sundaram

Teacher's Sign.: _____

Scanned by CamScanner

2. $T(n) = T\left(\frac{2n}{3}\right) + 1$

Step 1 : Comparing with standard form

i.e $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a = 1 \qquad b = \frac{3}{2} \qquad f(n) = 1$

Step 2 : Calculate $n^{\log_b a}$

$n^{\log_b a} = n^{\log_{3/2} 1} = n^0$

$$\boxed{n^{\log_b a} = 1}$$

Step 3 : Identifying the case

$f(n) = 1 \qquad n^{\log_b a} = 1$

$$\therefore \boxed{f(n) = n^{\log_b a}}$$

i.e $f(n) = \Theta(n^{\log_b a})$

By case 2 of master theorem,

$T(n) = \Theta(n^{\log_b a} \log n)$

$$\boxed{T(n) = \Theta(\log n)}$$

Scanned by CamScanner

EXPERIMENT : No.

PAGE No.
DATE

3. $T(n) = 3T\left(\frac{n}{4}\right) + n \log n$

Step 1: Comparing with standard form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$\therefore \quad a = 3 \qquad b = 4 \qquad f(n) = n \log n$

Step 2: Calculate $n^{\log_b a}$

$$n^{\log_b a} = n^{\log_4 3} = n^{0.793}$$

Step 3: Identifying the case

(i) $f(n) = n \log n \qquad n^{\log_b a} = n^{0.793}$

$$\therefore \boxed{f(n) > n^{\log_b a}}$$

i.e $f(n) = \Omega(n)$

$= \Omega(n^{0.793 + \epsilon})$ where $\epsilon \cong 0.2$

(ii) $af(n/b) = 3\left[\frac{n}{4} \log \frac{n}{4}\right]$

$= \frac{3}{4} n \log n - \frac{3}{4} \log 4$

$$\boxed{af(n/b) \leq cn \log n} \quad \text{where } c = \frac{3}{4}$$

By case 3 of Master theorem,

$\therefore \quad T(n) = \Theta(f(n)) = \Theta(n \log n)$

Teacher's Sign.: _____

Scanned by CamScanner

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

Step 1: Comparing with standard form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$\therefore \quad a = 4 \qquad b = 2 \qquad f(n) = n$

Step 2: Calculate $n^{\log_b a}$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

Step 3: Identifying the case

$f(n) = n \qquad n^{\log_b a} = n^2$

$\therefore \boxed{f(n) \ll n^{\log_b a}}$

i.e $f(n) = O(n^{2-\epsilon})$ where $\epsilon = 1$

By case 1 of master theorem,

$$T(n) = \Theta(n^{\log_b a})$$

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

Scanned by CamScanner

5. $T(n) = 2T(n/2) + n \log n$

Step 1: Comparing with standard form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$\therefore a = 2 \quad b = 2 \quad f(n) = n \log n$

Step 2: Calculate $n^{\log_b a}$

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

Step 3: Identifying the case

$f(n) = n \log n \qquad n^{\log_b a} = n$

$\therefore f(n)$ is asymptotically larger than $n^{\log_b a}$, it is not polynomially larger.

$\therefore$ The ratio $f(n) / n^{\log_b a} = \log n$ and $\log n$ is asymptotically less than n^{ϵ} for any positive constant ϵ.

$\therefore$ Hence, Not solvable using master method because recurrence falls in the gap between case 2 & case 3.

Scanned by CamScanner

6. $T(n) = 2T(n/2) + \Theta(n)$

$f(n) = n \quad n^{\log_b a} = n$

– case 2

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

7. $T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2)$

$f(n) = n^2 \quad n^{\log_b a} = n^{\log_2 8} = n^3$

– case 1

$\therefore T(n) = \Theta(n^3)$

8. $T(n) = 7T(n/2) + \Theta(n^2)$

$f(n) = n^2 \quad n^{\log_b a} = n^{\log_2 7} = n^{2.80}$

– case 1

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

9. $T(n) = 2T(n/4) + 1$

$f(n) = 1 \quad n^{\log_b a} = n^{\log_4 2} = n^{0.5} = n^{1/2} = \sqrt{n}$

– case 1

$T(n) = \Theta(n^{\log_4 2}) = \Theta(n^{0.5})$

10. $T(n) = 2T(n/4) + \sqrt{n}$

$f(n) = \sqrt{n} \quad n^{\log_b a} = \sqrt{n}$

– case 2

$T(n) = \Theta(\sqrt{n} \log n)$

Scanned by CamScanner

EXPERIMENT : No.

PAGE No.
DATE

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

$f(n) = n$

$n^{\log_b a} = n^{\log_4 2} = \sqrt{n}$

$f(n) = \Omega(n^{\log_b a + \epsilon})$ — case 3

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

12. $T(n) = 2T(n/4) + n^2$

$f(n) = n^2$

$n^{\log_b a} = \sqrt{n}$

$f(n) > n^{\log_b a}$

$af(n/b) = 2\left[\frac{n^2}{16}\right] = \frac{2}{16} n^2 \leq cn^2$

$c = \frac{3}{16} < 1$

case 3

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

Scanned by CamScanner

12. $T(n) = 4T(n/2) + n^2 \log n$

Step 1: Comparing with standard form

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$a = 4 \qquad b = 2 \qquad f(n) = n^2 \log n$

Step 2: Calculate $n^{\log_b a}$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

Step 3: Identifying the case

$f(n) = n^2 \log n \qquad n^{\log_b a} = n^2$

$\therefore f(n)$ is asymptotically larger than $n^{\log_b a}$, but it is not polynomially larger

$\therefore$ The ratio $f(n) / n^{\log_b a} = \log n$ and $\log n$ is asymptotically less than n^{ϵ} for any positive constant ϵ.

$\therefore$ Hence, **Not Solvable** using master method because recurrence falls in the gap between case 2 & case 3.

Scanned by CamScanner

THE MASTER METHOD

EXPERIMENT : No. (FOR SUBTRACT & CONQUER RECURRENCE) PAGE No. DATE

OR

(FOR DECREASING FUNCTION)

MASTER THEOREM

Let $a > 0$, $b > 0$ and $k >= 0$ be constants and $f(n)$ be a function such that $f(n) = O(n^k)$

Let $T(n)$ be a recurrence defined as,

$$T(n) = aT(n-b) + f(n) \qquad n > 1$$

$$= c \qquad n \leq 1$$

where 'c' is constant.

then $T(n)$ has following asymptotic bounds:

Case 1:

If $a < 1$ then

$T(n) = O(n^k)$ i.e $O(f(n))$

Case 2:

If $a = 1$ then

$T(n) = O(n^{k+1})$ i.e $O(n * f(n))$

Case 3:

If $a > 1$ then

$T(n) = O(n^k a^{n/b})$ i.e $O(f(n) \times a^{n/b})$

Teacher's Sign.: _____

Scanned by CamScanner

Ex ① $T(n) = 3T(n-2) + n$

Step 1: comparing with standard form

$T(n) = aT(n-b) + f(n)$

$\therefore a = 3 \quad b = 2 \quad f(n) = n \quad k = 1$

Step 2: Identifying case.

$\because a = 3 > 1$

By case 3 of master theorem,

$T(n) = O(f(n)\, a^{n/b})$ OR $O(n^k a^{n/b})$

$$\boxed{T(n) = O(n \times 3^{n/2})}$$

② $T(n) = T(n-1) + n$

Step 1: comparing with standard form

$T(n) = aT(n-b) + f(n)$

$\therefore a = 1 \quad b = 1 \quad f(n) = n \quad k = 1$

Step 2: Identifying case.

$\because a = 1$

By case 2 of master Theorem,

$T(n) = O(n \times f(n))$ OR $O(n \times n^k)$

$= O(n^2)$

Scanned by CamScanner

EXPERIMENT : No.

PAGE No.
DATE

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

Step 1: Comparing with standard form

$T(n) = aT(n-b) + f(n)$

$\therefore a = 2 \quad b = 2 \quad f(n) = n^2 \quad k = 2$

Step 2: Identifying case.

$\because a = 2 > 1$

By case 3 of master theorem,

$T(n) = O(f(n)\, a^{n/b})$ OR $O(n^k a^{n/b})$

$= O(n^2 2^{n/2})$

Scanned by CamScanner