Objectives

- Recursive Tree Method
- Examples of solution of Recurrence relation using Recursive Tree Method
- Master theorem for decreasing function
- Master theorem for dividing function
- Assignment (Page No. 88-98, CORMEN, Introduction to Algorithms)

Recursion Tree Method for solving Recurrence Relation:

(Ref. Page No. 88-93, CORMEN, Introduction to Algorithms)

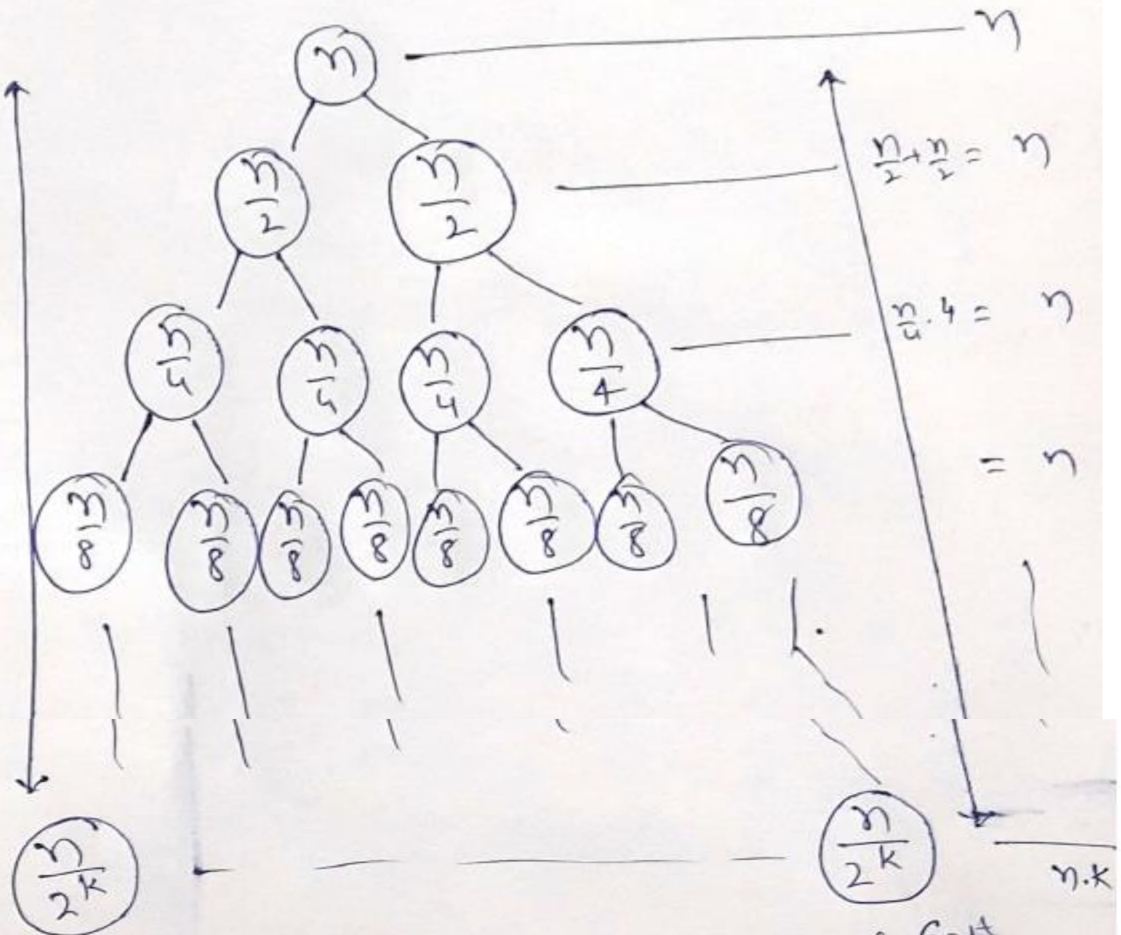
$$T(n) = 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + \theta(n^2)$$

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

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

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

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

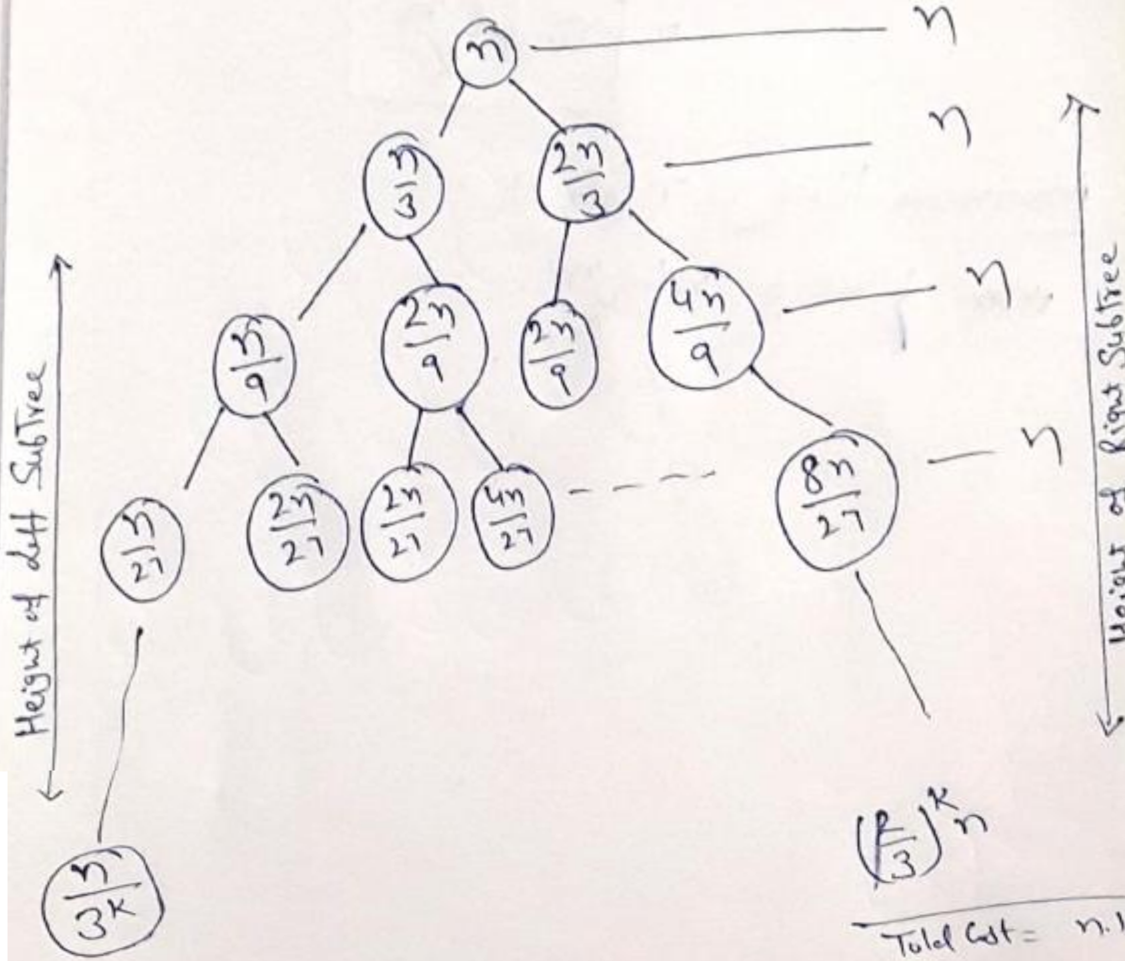


Total Cost

$$\frac{n}{2^k} = 1 \qquad \boxed{k = \log_2 n}$$

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

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n \quad \text{[CORMEN, Page 91]}$$



So. Height of left Subtree $= \frac{n}{3^k} = 1$

$\Rightarrow \boxed{k = \log_3 n}$

Height of Right Subtree $\frac{n}{\left(\frac{3}{2}\right)^k} = 1$

$\Rightarrow \boxed{k = \log_{\frac{3}{2}} n}$

$\log_3 n < \log_{3/2} n$

Total Cost $= n.K$

$= \boxed{n.\log_{3/2} n}$

Minimum time $= n\log_3 n$

Max. time $= n\log_{3/2} n$

Master Theorem for Recurrence Relation:

Master Theorem for decreasing function

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

$$T(n) = T(n-1) + n ----------- O(n^2)$$

$$T(n) = T(n-1) + n^2 ----------- O(n^3)$$

$$T(n) = T(n-1) + \log n -------- O(n \log n)$$

$$T(n) = T(n-2) + 1 \rightarrow \frac{n}{2} \rightarrow O(n)$$

$$T(n) = T(n-50) + n ---------- O(n^2)$$

The general form of recurrence relation for decreasing function is:

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

$$where\ \ a > 0, b > 0\ \ and\ \ f(n) = O(n^k)\ ,\ k \geq 0$$

Case 1: if a=1 $\Rightarrow O(n * f(n))\ \ OR\ \ O(n^{k+1})$

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

$$T(n) = T(n-1) + n ------------ O(n^2)$$

$$T(n) = T(n-1) + n^2 ------------ O(n^3)$$

$$T(n) = T(n-1) + \log n ---------- O(n \log n)$$

Case 2: if a>1 $\Rightarrow O(f(n)a^{\frac{n}{b}})$ OR $O(n^k a^{\frac{n}{b}})$

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

$$a = 2, b = 2, f(n) = 1$$

$$\Rightarrow Since\ a > 1\ So\ O(f(n) * a^{\frac{n}{b}}) = O(2^{\frac{n}{2}})$$

Case 3: if a<1 $\Rightarrow O(f(n))$ OR $O(n^k)$

Master Theorem for dividing function:

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

$$T(n) = 8T(\frac{n}{2}) + n ------------ \theta(n^3)$$

$$T(n) = 2T(\frac{n}{2}) + n ------------ \theta(n \log n)$$

$$T(n) = 4T(\frac{n}{2}) + n^2 \log n -------- O(n^2 \log^2 n)$$

The general form of the recurrence relation is:

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

$$a \geq 1,\ b > 1 \ \ and \ \ f(n) = O(n^k \log^p n)$$

Case 1: $\log_b a > k \ \ then \ \ \theta(n \log_b a)$

Example 1:

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

$$a = 2, b = 2, f(n) = (n^0 \log^0 n) = 1$$

$$k = 0, p = 0$$

$$\log_2 2 = 1 \ and \ \ k = 0$$

$$case\,1\ is\ satisfied \Rightarrow \theta(n)$$

Example 2

$$T(n) = 8T(\frac{n}{2}) + n$$

$$a = 8, b = 2, f(n) = (n^1 \log^0 n) = n$$

$$k = 1, p = 0$$

$$\log_2 8 = 3 \ \ and \ \ k = 1 \Rightarrow \log_b a > k$$

$$case\,1 \ is \ satisfied \Rightarrow \theta(n^3)$$

Case 2: $\log_b a = k$ *then*

$$if\ p > -1 \Rightarrow \theta(n^k \log^{p+1} n)$$

$$if\ p = -1 \Rightarrow \theta(n^k \log\log n)$$

$$if\ p < -1 \Rightarrow \theta(n^k)$$

Example 1:

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

$$a = 2, b = 2, f(n) = (n^1 \log^0 n) = n$$

$$k = 1, p = 0$$

$$\log_2 2 = 1\ and\ \ k = 1\ \ and\ \ p = 0 > -1$$

$$case\ 2\ is\ satisfied \Rightarrow \theta(n \log n)$$

Similarly

$$4T\left(\frac{n}{2}\right)+n^2 \text{-------} \quad O\left(n^2 \log n\right)$$

$$4T\left(\frac{n}{2}\right)+n^2 \log n \text{------} \quad O(n^2 \log^2 n)$$

$$2T\left(\frac{n}{2}\right)+\frac{n}{\log n} \text{----------} \quad O(n \log \log n)$$

Case 3: $\log_b a < k \;\; then$

$$if \;\; p \geq 0 \Rightarrow \theta(n^k \log^p n)$$

$$if \;\; p < 0 \Rightarrow \theta(n^k)$$

$T(n) = 3T(\frac{n}{4}) + n \log n$ (Ref. Page No. 95, CORMEN, Introduction to Algorithms)

THANKS !