Module 1

Fundamental of Algorithms

Smita Attarde

Introduction:-

Complexity of recursive algorithms →

→ When an algorithm contains a recursive call to itself, its running time is described by a recurrence equation or recurrence.

→ A recurrence describes the overall running time of a problem of size n.

→ Mathematical tools are used to solve these recurrences.

→ These tools are:-

1) Substitution method.
2) Recursion tree method.
3) Master method.

1) Substitution method →

→ The solution is guessed and mathematical induction is used to prove that the guess is correct or incorrect.

→ This method is powerful, but it can be applied only in cases when it is easy to guess the form of the answer.

Smita Attarde

Smita Attarde

eg) Consider a recurrence relation

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

with initial condition $T(0) = 0$.

Soln:

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

If $n = 1$, then $T(1) = T(0) + 1$

$= 0 + 1$

$= 1$

If $n = 2$, then $T(2) = T(1) + 2$

$= 1 + 2$

$= 3$

If $n = 3$, $T(3) = T(2) + 3$

$= 3 + 3$

$= 6$

Thus,

n	T(n)
1	1
2	3
3	6
⋮	⋮

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

This can be denoted in terms of Big-Oh notation as -

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

Smita Attarde

eg) Solve the following recurrence relation using substitution method.

$T(n) = T(n-1) + 1$ where $T(0) = 0$.

soln:-

If $n=1$, $T(1) = T(0) + 1$
$= 0 + 1$
$= 1$

If $n=2$, $T(2) = T(1) + 1$
$= 1 + 1$
$= 2$

If $n=3$, $T(3) = T(2) + 1$
$= 2 + 1$
$= 3$

Thus,

n	T(n)
1	1
2	2
3	3
⋮	

$T(n) = n$

ie. $T(n) = O(n)$.

Smita Attarde

2) Recursion tree method →

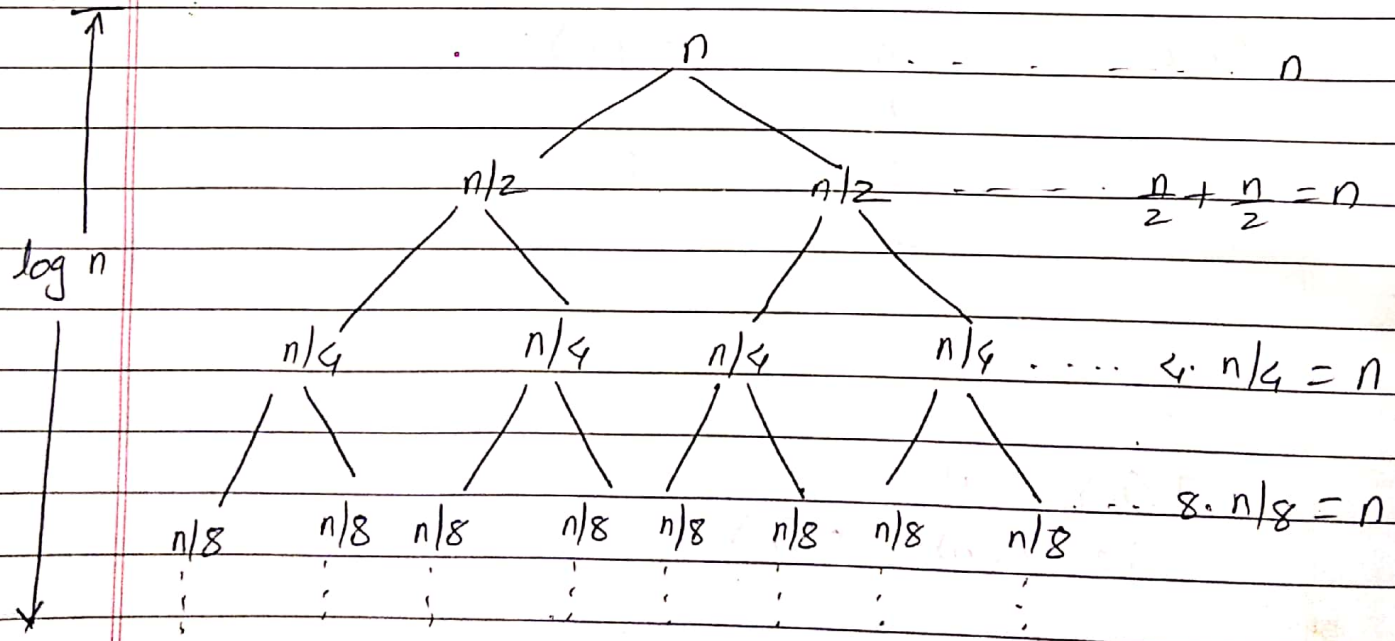
→ In this method, we draw a recurrence tree and calculate the time taken by every level of the tree.

→ Finally, we sum the work done at all the levels.

→ To draw the recurrence tree, we start from the given recurrence relation and keep drawing till we find a pattern among all the levels. This pattern is typically a arithmetic or geometric series.

eg) Solve the recurrence relation using recursion tree method.

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



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

Hence, the total cost $= n \log n \cdot T(1)$

$= n \log n$

Smita Attarde

3) Master Method →

→ It is a direct way to get the solution.

→ Master method works only for the following type of recurrences.

$$T(n) = aT(n/b) + F(n)$$

where, a is constant and $a \geq 1$

$n \geq d$ and d is some constant

[F(n) must be positive]

case 1: If $F(n)$ is n^d where $d \geq 0$, then a) if $a < b^d$, $T(n) = \Theta(n^d)$ b) if $a = b^d$, $T(n) = \Theta(n^d \log n)$ c) if $a > b^d$, $T(n) = \Theta(n^{\log_b a})$

case 2: If $F(n)$ is $(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a})$

case 3: If $F(n)$ is $(n^{\log_b a} \log^k n)$, then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

case 4: If $F(n)$ is $(n^{\log_b a + c})$, then $T(n) = \Theta(F(n))$

eg) Solve using Master Method →

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

Soln: we have, $T(n) = aT(n/b) + F(n)$
Here, $a = 4$, $b = 2$ and $F(n) = n$ [case 1]
ie. n^1, hence $d = 1$

Now, $b^d = 2^1 = 2$
Thus, $a > b^d$ [case 1c]

$$\therefore T(n) = \Theta(n^{\log_b a})$$
$$= \Theta(n^{\log_2 4})$$
$$= \Theta(n^2)$$

Thus, the time complexity is $\Theta(n^2)$

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

Soln: Here, $a = 2$, $b = 2$ and $F(n) = n \log n$ [case 3]

$$\therefore T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$$

Now, $\log_b a = \log_2 2 = 1$

$$\log^{k+1} = \log^{1+1} = \log^2$$

$$\therefore T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$$
$$= \Theta(n^1 \log^2 n)$$
$$= \Theta(n \log^2 n)$$

Smita Attarde

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

Soln:- Here,

$a = 4$, $b = 2$ and $F(n) = n^2$ [case 1]

ie. $d = 2$

Now, $b^d = 2^2 = 4$

Here, $a = b$ (case 1b)

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

$= \Theta(n^2 \log n)$

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

Soln: Here,

$a = 2$, $b = 2$, $F(n) = n^3$ [case 1]

ie. $d = 3$

Now, $b^d = 2^3 = 8$

Here, $a < b^d$ [case 1a]

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

$= \Theta(n^3)$

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

Here, $a = 2^n$, $b = 2$ and $d = n$

a and d should be constant.

Hence, master method is not applicable.

6) $T(n) = 2T(n/4) + n^{0.51}$

Here, $a=2$, $b=4$, $d=0.51$

$b^d = 4^{0.51} = 2.03$

$a < b^d$ (case 1a)

$\therefore T(n) = \Theta(n^{0.51})$

7) $T(n) = 0.5\,T(n/2) + 1/n$

Here, $a = 0.5$, $b=2$, $F(n) = 1/n$

Since, $a < 1$, master method is not applicable

8) $T(n) = 96\,T(n/3) + n^2 \log n$

Here, $a=96$, $b=3$ and $F(n) = n^2 \log n$ [case 3]

$\therefore T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

$\log_b a = \log_3 9 = 2$

$\log^{k+1} = \log^{1+1} = \log^2$

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

9) $T(n) = 64\,T(n/8) - n^2 \log n$

Here, $a=64$, $b=8$ and $F(n) = -n^2 \log n$

Master method not applicable, since $F(n)$ is negative.

Smita Attarde