Lecture Notes

on

Analysis of an Algorithm

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

July 2020

(Be safe and stay at home)

Recurrence

- When an algorithm contains a recursive call to itself, it's running time can be described by a recurrence or recurrence equation.
- T(n)= $\begin{Bmatrix} O(1); if n < 1 \\ 2T(\frac{n}{2}) + n); otherwise \end{Bmatrix}$

Solving Recurrence Relations

1. Substitution Method
2. Master's Method
3. Recurence-tree Method

Substitution Method: Problem 01

T(n)= $\begin{cases} O(1); if n <= 1 \\ T(n-1)+n); otherwise \end{cases}$ T(n)=T(n-1)+n

=T(n-2)+T(n-1)+n
=T(n-3)T(n-2)+T(n-1)+n
:
:
=1+2+3+..........+n
= $\frac{n(n+1)}{2}$
=(n^2)

Problem 02:

T(n)= $\left\{ \begin{array}{c} O(1); if n <= 1 \\ n * T(n-1); otherwise \end{array} \right\}$ T(n)=n*T(n-1)

=n*[(n-1)*T(n-2)]
=n*[(n-1)*(n-2)*T(n-3)]
:
: Repeat (n-1) times
=n*(n-1)*(n-2)*......*T(n-(n-1))
=n*(n-1)*(n-2)*......*2*1
=$O(n^n)$

Problem 03:

T(n)= $\begin{Bmatrix} O(1); if n <= 1 \\ T(\frac{n}{2}) + c; otherwise \end{Bmatrix}$

assume $n = 2^k$

$log_2 n = k$

$= [T(\frac{n}{2^2}) + c] + c$

$= [T(\frac{n}{2^3}) + c] + 2c$

$= T(\frac{n}{2^4}) + 4c$

:

$: Repeat'k'times$

$= T(\frac{n}{2^k}) + k.c$

$= T(\frac{n}{n}) + log_2 n.c$

$= 1 + c.log_2 n$

$= O(log_2 n)$

Practice Questions:

1. T(n)= $\begin{Bmatrix} O(1); if n <= 1 \\ 2T(\frac{n}{2}) + n; otherwise \end{Bmatrix}$
2. T(n)= $\begin{Bmatrix} O(1); if n <= 1 \\ 8T(\frac{n}{2}) + n^2; otherwise \end{Bmatrix}$
3. T(n)= $\begin{Bmatrix} O(1); if n <= 1 \\ 7T(\frac{n}{2}) + n^2; otherwise \end{Bmatrix}$
4. T(n)= $\begin{Bmatrix} O(1); if n <= 1 \\ 3T(\frac{n}{2}) + n^2; otherwise \end{Bmatrix}$

Q & A?

Queries are welcome on slack channel for discussion