DAA

19_milind modi

PAGE:
DATE: 4-3-21

Name :- Milind N. Modi
Roll No : 19
Subject : DAA .Theory

Q1

$$T(n) = \begin{cases} 2 & \text{if } n=2 \\ 2T(n/2)+n & \text{if } n=2^k \text{ for } k>1 \end{cases}$$

~~Step~~ assume

Base case

$n=2$, $T(2) = 2 = 2 \lg 2$.

above holds for the initial step.

Inductive step

Let assume $k > 1$

$\therefore T(2^k) = 2^k \lg 2^k$ —— (1)

so it must holds for the $\therefore k+1$

$$T(2^{k+1}) = 2^{k+1} \lg 2^{k+1}$$

$$T(2^{k+1}) = 2T(2^{k+1}/2) + 2^{k+1} \quad \textcircled{1}$$

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

we can also write like this $(n^{a+b} = n^a \cdot n^b)$

$$= 2T(2^k) + 2 \cdot 2^k$$

$$= 2 \cdot 2^k \lg 2^k + 2 \cdot 2^k \quad \longrightarrow \text{from } (1)$$

$$= 2 \cdot 2^k (\lg 2^k + 1)$$

$$= 2^{k+1} (\lg 2^k + \lg 2)$$

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

∴ By both inductive & base case Step performed, $T(n) = n \lg n$ holds for all n that exact power of 2

Q.2 $f(n) = \frac{n^3}{1000} - 100n^2 + 3$

$O(f) = O(\max(n^3/1000, 100n^2, 3))$

Applying rule to each of these.

$O(\max(n^3, n^2, 1))$

$O(1) < O(n) < O(n^2) < O(n^3)$

$= O(n^3)$

RHS $= c2 \cdot n^3 \geq n^3/1000 - 100n^2 + 3$

$c2 \cdot n^3 > n^3/1000 + 3$

$c2 \geq 1/1000 + 3$

$c2 = 4$

LHS =

$= c1 \times n^3 < n^3/1000 - 100n^2 + 3$

$= c1 * n^3 \leq n^3/1000 - 100n^2$

$c1 \leq 1/1000 - 100/n^2$

$c1 \leq 1/2000$

~~LHS~~

So the answer is $O(n^3)$

Q3

$$T(n) = 2T(n/2) + 17$$

$$T(n) = 2T(n/2) + 17 + n$$

$$T(n) = 2T(n/2) + 17 + n$$

Cls linear form (n)

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

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

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

$$T(n/k) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

Assume $= \frac{n}{2^k} = 1$, $n = 2^k$

$$K = \log n$$

$$\rightarrow T(n) = n\,T(1) + \log(n)$$

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

hence $O(n \log n)$

Q 4) Algorithm :- An algorithm are simply a series of instruction that are followed step by step to do something useful or solve problem

In short algorithm take some value as input and give some valu after processing an input which is output

2) Divide and Conquer

Divide the probalems into a number of Problems that are smaller instances of the problem

Conquer the subproblems by solving recursively.

The solutions to the subproblems into the solution for orignal problem.

3) Recurrence :- is an equation or inequality that describes a function in terms of its value on smaller input.