CSA0672 - Design and Analysis of Algorithm

Name :- O. Bhanu prakash Reddy Data : 06/06/24

Reg no :- 192372075 Analytical : 01

1. Solve the following recorance relations:

a) $x(n) = x(n-1)+5$ for $n>1$ $x(1)=0$

$$x(n) = x(n-1)+5 \quad -①$$

$$x(n-1) = x(n-1-1)+5$$

$$= x(n-2)+5 \quad -②$$

$$x(n-2) = x(n-2-1)+5$$

$$= x(n-3)+5 \quad →③$$

sub eq ③ in ②

$$x(n-1) = x(n-3)+5+5$$

$$= x(n-3)+10$$

$$x(n) = x(n-3)+10+5$$

$$= x(n-3)+15$$

$$x(n) = x(n-k)+5k.$$

$$n-k = 1$$

Now

$$x(n) = x(1)+5(n-1)$$

$$x(n) = 0+5n-5$$

$$O(n)//$$

b) $x(n) = 3x(n-1)$ for $n>1$, $x(1)=4$

$x(n) = 3x(n-1) \rightarrow (1)$

$x(n-1) = 3x(n-1-1) = 3x(n-2) \rightarrow (2)$

$x(n-2) = 3x(n-3) \rightarrow (3)$

Sub eq (3) in 2.

$x(n-1) = 3[3x(n-3)]$

$x(n-1) = 9x(n-3) \rightarrow (4)$

Sub eq (4) in (1).

$x(n) = 3[9x(n-3)]$

$x(n) = 27x(n-3)$

At some k

$x(n) = 3^k x(n-k) \rightarrow (5)$

$n-k=1$

$k = n-1$

Eq (5) $\Rightarrow x(n) = 3^{n-1} a(1)$

$= 3^{n-1} \cdot 4$

$= 3^n \cdot 3^{-1} \cdot 4$

$= 3^n$

∴ The time complexity $= O(3^n)$

(c) $x(n) = x(n/2) + n$ For $n>1$ $x(1)=1$

$x(n) = x(n/2) + c \rightarrow (1)$

$x(n/2) = x(n/4) + c \rightarrow (2)$

$x(n/4) = x(n/8) + c \rightarrow (3)$

sub (2) in (1);

$x(n) = x(n/4) + c + c$

$= x(n/4) + 2c \rightarrow (4)$

$= x(n/2^2) + 2c$

sub (3) in (4)

$x(n) = x(n/8) + c + 2c$

$= x(n/2^3) + 3c$

$x(n) = x(n/2^k) + kc.$

$n = 2k. \quad x(1) = 1$

$x(n) = x(n/n) + kc$

$x(n) = 1 + kc$

$x(n) = 1 + \log n \cdot c$

The time complexity = $O(\log n)$

(d) $x(n) = x(n/3) + 1$ for $n > 1$ $x(1) = 1$ (solve for $n = 3k$)

$x(n) = x(n/3) + 1 \rightarrow (1)$

$x(n/3) = x(n/3/3) + 1$

$= x(n/3^2) + 1 \rightarrow (2)$

$x(n/3^2) = x(n/3^3) + 1 \rightarrow (3)$

sub (2) in (1)

$x(n) = x(n/2^2) + 2 \rightarrow (4)$

sub (3) in 4.

$x(n) = x(n/3^3) + 3 \rightarrow (5)$

$x(n) = x(n/3^k) + k$

$= x(n/n) + k$

$= x(1) + k$

$= 1 + k$

$x(n) = \log n$

Time complexity = $O(\log n)$

2) Evaluate following recurrence completely.

(i) $T(n) = T(n/2) + 1$ where $n = 2^k$ for all $k \geq 0$

$T(n) = T(n/2) + 1 \quad n = 2^k$

$n = 2^k$

$T(2^k) = T(2^k/2) + 1 = T(2^{k-1}) + 1$

$n = k-1$

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

$n = k-2$

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

$T(2^1) + T(2^0) + 1$

$n = 2^k \Rightarrow k = \log_2 n$

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

since

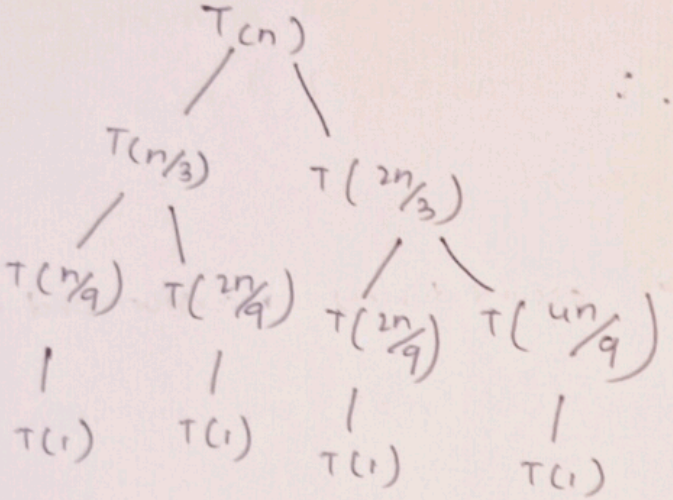
$2^0 = 1$, $T(2^0) = T(1)$

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

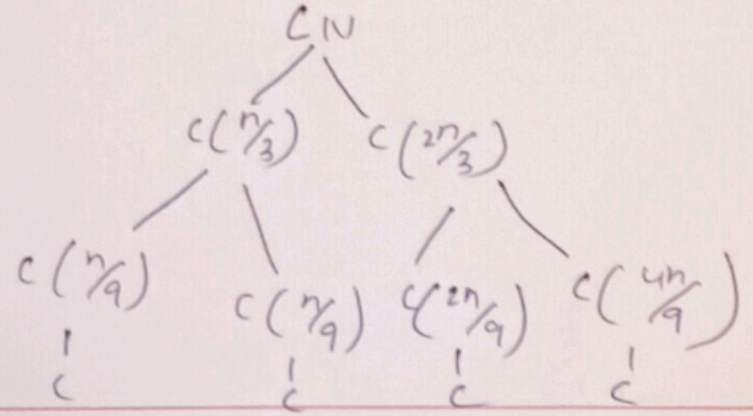
$T(n) = 1 + \log_2 n)$

∴ Time complexity = $O(\log n)$

(ii) $T(n) = T(n/3) . T(2n/3) + cn$



∴ we use Recursion Tree method



(3) Consider following algorithm

```
min1(A[0...n-1])
if n=1 return A[0]-1
Else temp = min1[A(0...n-2),)
if temp ≤ A(n-1) return temp
else.
  return A[n-1] - n-1
```

(a) what does the algorithm compute?

This algorithm compute minimum elements in an array A of size n.

If i<n, A[i] is smaller than all elements then, A[i] j = i+1 to n-1, the it returns A[i] it also returns the left most minimal element.

b) Mainly comparision occurs during recursion and solve it?

So $T(n) = T(n-1) + 1$ ∴ $n > 1$ (one comparison)

$T(1) = 0$ (No compare when $n = 1$)

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

$= 0 + (n-1)$

$= n-1$

Time complexity = $O(n)$

4) Analyze order of growth

1) $f(n) = 2n^2+5$ and $g(n) = 7n$ use then $\Omega(g(n))$ notation,

$f(n) = 2n^2+5$

$g(n) = 7n$

$f(n) \geq c\, g(n)$

$n = 1$

$f(1) = 2(1)^2+5 = 7$

$g(1) = 7.$

$n = 2$

$f(2) = 2(2)^2+5$

$= 8+5 = 13$

$g(2) = 7 \times 2 = 14$

$n = 3$

$f(3) = 2(3)^2+5$

$= 18+5$

$= 23$

$g(3) = 21$

$f(n) \geq g(n) \cdot c$

$f(n)$ is always greater than or equal to $x(g(n))$ when n value is greater or equal to 3

$\therefore \quad f(n) = \Omega(g(n))$

$f(n)$ grows more than $g(n)$ from below asymptotically.