R. Haresh Bharadwaj
192321031

Assignment

Solve the following recurrence relation:

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

at $n = 1$; $x(1) = 0$

$n = 2$; $x(2) = x(2-1) + 5$
$= 0 + 5 = 5$

$n = 3$; $x(3) = x(3-1) + 5$
$= x(2) + 5$
$= 10$

$n = 4$; $x(4) = x(3) + 5$
$= 15$

$x(n)$ is $+5$ for each increment

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

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

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

$n = 1$; $x(1) = 4$ (Given)

$n = 2$; $x(2) = 3x(2-1)$
$= 3x(1)$
$= 3 \times 4 = 12$

$n = 3$; $x(3) = 3x(3-1)$
$= 3x(2)$
$= 3 \times 12$
$x(3) = 36$

($x(n)$ is obtained by $\times 3$)

$n = 4$; $x(4) = 3x(4-1)$
$= 3(36)$
$= 108$

$\therefore x(n) = 4 \times 3^{n-1}$

c] $x(n) = x(n/2) + n \quad n > 1 \quad x(1) = 1$

[Solve $n = 2^k$]

$n = 2^k$

$n = 1;\ x(1) = 1$

$n = 2;\ x(2) = x(2/2) + n$

$= 1 + 2 = 3$

$n = 4;\ x(4/2) + 4$

$= x(2) + 4$

$= 7$

$n = 8;\ x(8/2) + 8$

$= x(4) + 8$

$= 7 + 8 = 15$

$n = 16;\ x(8) + 16$

$= 15 + 16 = 31$

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

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

$2^k = n$

$x(n) = x(2^k) = 2^{(\log_2 n) + 1} - 1$

$= 2 \cdot 2^{\log_2 n} - 1$

$= 2n - 1$

d) $x(n) = x(n/3) + 1 \quad n > 1 \quad x(1) = 1$ (Solve $n = 3^k$)

$x(1) = 1$

$x(3) = x(1) + 1 = 2$

$x(9) = x(3) + 1 = 3$

$x(27) = x(9) + 1 = 4$

$x(n) = 1 + \log_3 n$

~~$x(n) = 1$~~

2] Evaluate the following recurrence completely

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

Assume $n = 2^k$ i.e $k = \log n$

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

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

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

$$T(2^k) = T(2^{k-3}) + 3$$

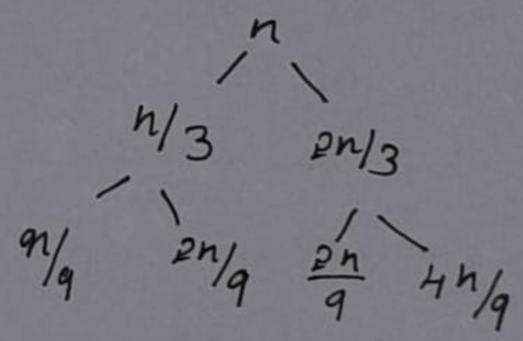
$$T(2^k) = T(2^{k-k}) + k = T(2^0) + k = T(1) + k$$

If $T(1) = 1$

$T(2^k) = 1 + k$ i.e $T(n) = \log n + 1$

Thus, we get $T(n) = O(\log n)$

ii] $T(n) = T(n/3) + T(2n/3) + cn$, where c is constant



$T(n) =$ Sum of all number

length $= \log_3 n$

$T(n) > n \log_{3/2} n$ $\therefore T$ is $\Omega(n \log n)$

depth $= \log_{3/2} n$

$T(n) \leq n \log_{3/2} n$

3] Consider the following recursion algorithm

Min [A[0]... n-]

if n-1 return A[0]

Else temp = min1 A[0 ... n-2]

if temp <= A(n-1] return temp

else

Return A(n-1)

a] Wt does this algorithm Compute?

return min Value in Array A.

1. Best Case (n=1)

if n=1. only one element. It return the A[0] as its min Value in a single element array

2. recursive Case:

•] if n>1, create the temp

•] Call recursivly (A[0 to n-2]) = first n-1 element

•] Comparing temp with last element (A[n-1])

if temp < A(n-1)

return temp

else

return A[n-1]

b] Set up a recurrance relation for the algorithm basic operation Count & Solve it.

Base case: $T(1) = C_1$ [constant (C_1)]

recursive case = $T(n) = T(n-1) + C_2$ [$C_2 \to$ constant]

Final Solutio

$$T(n) = C_2 * n^2 + (C_1 - C_2)$$

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

4] Analyse the order of growth

i] $F(n) = 2n^2+5$ & $g(n) = 7n$. Use the $\Omega g(n)$

As n grown. $2n^2$ grows much faster than $7n$

$F(n) = 2n^2+5 >= C.(7n)$ $\quad f(n) \geq C \cdot g(n)$

if $n=1$; $7 = 7$

$n=2$; $13 \quad 14$

$n=3$; $23 > 21$ $\quad F(n)$ is greater

$n=4$; $37 > 28$ $\quad F(n) \geq g(n)$

$n=5$; $55 > 35$