06/06/2024

B.V. Jayani
192371005
CSA 0669

1) Solve the following recurrence relation

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

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

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

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

at $n = 4$: $x(4) = x(4-1) + 5$
$= x(3) + 5$
$= 10 + 5$
$x(4) = 15$

$\because$ $x(n)$ increases by 5 for each increment, diff $(d) = 5$

$x(n) = x(1) + (n-1).d$ { formula for n^{th} term to find general form of $x(n)$.

Here, $x(1) = 0$, $d = 5$

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

$\boxed{x(n) = 5(n-1)}$

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

$n=1$, $x(1) = 4$ (gn.)

$n=2$: $x(2) = 3x(2-1)$

$= 3x(1)$

$= 3(4) = 12$

$x(2) = 12$

$x(3) = 3x(3-1)$

$= 3x(2)$

$= 3(12) = 36$

$x(3) = 36$.

$x(4) = 3x(4-1)$

$= 3x(3)$

$= 3(36) = 108$

$x(4) = 108$

$\because$ $x(n)$ obtained by multiplying the previous term by 3.

Ratio $= 3$

$x(n) = x(1) \cdot r^{n-1}$

Here $x(1) = 4$, $r = 3$

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

c) $x(n) = x(n/2) + n$ for $n > 1$ $x(1) = 1$
(solve for $n = 2k$)

$n = 1,$ $x(1) = 1$ $(n = 2k)$

$x(2) = x(2/2) + 2$

$= x(1) + 2$

$= 1 + 2 = 3$

$x(2) = 3$

$x(4) = x(4/2) + 4$

$= x(2) + 4$

$= 3 + 4 = 7$

$x(4) = 7$

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

$= x(4) + 8$

$= 7 + 8 = 15$

$x(8) = 15$

$x(16) = x(16/2) + 16$

$= x(8) + 16$

$= 15 + 16 = 31$

$x(16) = 31$

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

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

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

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

$\boxed{x(n) = 2n - 1}$

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

(solve for $n = 3^k$)

$n = 1$, $x(1) = 1$

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

$= x(1) + 1$

$= 1 + 1 = 2$

$x(3) = 2$

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

$= x(3) + 1$

$= 2 + 1 = 3$

$x(9) = 3$

$x(27) = x(27/3) + 1$

$= x(9) + 1$

$= 3 + 1 = 4$

$x(27) = 4$

$x(n) = 1 + \log_3 n$ hold true for $n = 3^k$

$$\boxed{x(n) = 1 + \log_3^n}$$

2) Evaluate the flwng. recurrences completely

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

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

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

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

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

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

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

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

$$= T(2^0) + k$$

$$= T(1) + k$$

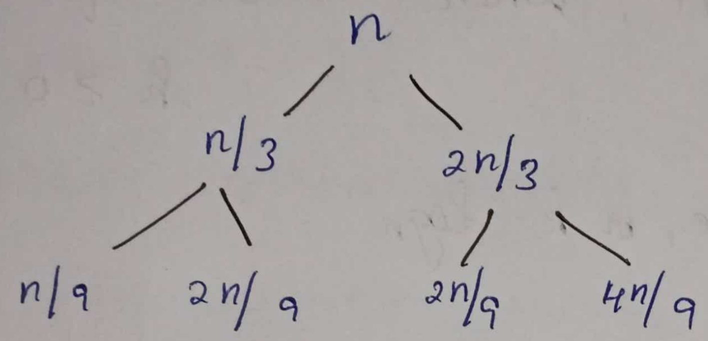
If $T(1) = 1$, we get

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

i.e, $T(n) = \log n + 1$

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

ii) $T(n) = T(n/3) + T(2n/3) + cn$ where 'c' is a constant and 'n' is the input size.



$T(n) =$ " cn " $=$ sum of all nos. in this tree

length $= \log_3 n$

$T(n) \geq n \log_3 n$ $\quad (\because T$ is $\Omega(n \log n))$

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

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

T is $\Theta(n \log n)$

3) Consider the following recursion algorithm

Min [A [0 ...... n-1])

if n-1 return A [0]

else temp = min (A [0 ...... n-2])

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

else

return A [n-1]

a) What does this algorithm compute?

This algorithm computes minimum value in an array A.

i) Best case (n=1):

if {n=1, only 1 element. It returns the A[0] as its the min. value in a single element array.

ii) Recursive case (n >1):

=> if n>1, creates the temporary variable (temp)

=> Call recursively (A [0] to n-2) = first n-1 element

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

if temp $\leq$ A [n-1]

return temp

else:

return A[n-1]

b) Setup a recurrance relation for the algorithmic basic operation count and solve it.

Base case = $T(1) = C_1$ [C_1 is constant → return single element]

Recursive case = $T(n) = T(n-1) + C_2$ [C_2 → Constant representing the basic operations for comparsion and assignment.

final solution:

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

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

4) Analyze the order of growth

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

As n grows, $2n^2$ grows much faster than the 7n.

$F(n) = 2n^2 + 5 >= C \times 7n.$

if $n=1$, $7=7$

$n=2$, $13=14$

$n=3$, $23, 14$

$n=4$, $37, 28$

$n=5$, $55=35$

$n \geq 4$, $F(n) = 2n^2 > 7n$.

F (n) is always greater than or equal to $C * g(n)$

$F(n) = \Omega(g(n))$

$\because$ F(n) is atleast as fast as the order of growth of g(n). F(n) grows at least as fast as 7n as n approaches positive infinity.