CSA0669

06/06/2024

# ASSINGNMENT

AJMAL RAHMAN A
192324094
DESIGN AND ANALYSIS OF ALGORITHMS

1) Solve the following Recurrence relation

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

$x(n) = x(n-1) + 5$ — ①

$x(n-1) = x(n-2) + 5$ — ②

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

$x(n-2) = x(n-3) + 5$ — ③

$n = 2$

$x(2) = x(1) + 5 = 0 + 5$
$= 5$

$n = 3$

$x(3) = x(2) + 5 = 10$

$n = 4$

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

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

sub ② in ①

$x(n) = (x(n-2) + 5) + 10$
$= \boxed{x(n-2) + 10}$ — ④

sub. ③ in ④

$x(n) = x(n-3) + 5 + 10$
$= x(n-3) + 15$

$x(n) = x(n-1) + 5 \times (n-2) + 10 \ldots (n-(k-1)) + 5k$

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

$n - k = 1$

$n - 1 = k$

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

∴ Time complexity is $O(n)$

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

$x(n) = 3x(n-1)$ — (1)

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

$x(n-1) = 3x(n-2)$ — (2)

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

$x(n-2) = 3x(n-3)$ — (3)

Sub (3) in (2)

$x(n) = 3[3x(n-2)]$

$x(n) = 9x(n-2)$ — (4)

Sub. (4) in (3)

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

$= 27x(n-3)$

$n=2$

$x(2) = 3x(1) = 12$

$n=3$

$x(3) = 3x(2) = 36$

$n=4$

$x(4) = 3x(3) = 108$

$n=5$

$x(5) = 3x(4) = 324$

now,

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

$= 3^{n-1} \cdot x(1)$

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

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

$x(n) = 3(n-1) + 9(n-2) + 27(n-3) + \ldots 3^k($

$x(n) = 3^k x(n-k)$

$n-k=1$ $\quad = 3^{n+1} x(1)$

$k = n-1$ $\quad = O(3^n)$

∴ The Time complexity is $O(3^n)$

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

[solve $n = 2^k$]

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

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

$= 1 + 2 = 3$

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

$= n(2) + 4$

$= 7$

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

$= n(4) + 8$

$= 7 + 8$

$= 15$

$n(16) ; n(8) + 16$

$= 15 + 16 = 31$

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

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

$2^k = n$

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

$= 2 \cdot 2 \log_2 n - 1$

$= 2n - 1$

$\therefore$ Time complexity is $O(n)$

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

$x(1) = 1$

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

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

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

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

$\therefore$ Time complexity is $O(\log n)$

2) Evaluate following recurrence Complexity

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

assume $n = 2^k$ i.e $K = \log n$

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

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

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

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

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

$$\therefore T(n) = \log n + 1$$

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

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

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

$n$

$n/3 \quad 2n/3$

$n/9 \quad 2n/9 \quad 2n/9 \quad 4n/9$

$T(n) =$ sum of all number

length $= \log_3 n$

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

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

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

3) Consider the following recursion algorithm

Min [A [0] ... n]

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) 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 it's min value in a single element array.

2. Recursive case:

• If n > 1, create temp

• call recursively (A[0 + 0 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 algorithm basic operation count & solve it

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

recursive case = $T(n) = T(n-1) + C_2$ [$C_2 \rightarrow$ const]

Find solution

$T(n) = (2n^2 + (c_1 - c_2))$

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

4) Analyze the order of growth

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

As n grows, $2n^2$ grows much faster than $T(n)$

$f(n) = 2n^2+5 \geq c \cdot (7n)$ $f(n) \geq c \cdot g(n)$

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

$n = 2$ ; $13 > 14$

$n = 3$ ; $23 > 21$

$n = 4$ ; $37 > 28$

$n = 5$ ; $55 > 35$

f(n) is greater

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