CSA0672 DAA Analytical Assignment - 1

Name: R. Lavanya

Reg NO: 192321022

Course code: CSA0672

1) Solve the following recurrence relations

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

Given:

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

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

$n=2$

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

$= x(1)+5$

$= 5 \rightarrow ①$

$n=3$

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

$= x(2)+5$

$= 10 \rightarrow ②$

$n=4$

$x(4) = x(4-1)+5$

$= x(3)+5$

$= 15$

The given equation is $x(n) = x(1)+(n-1)d$ in the given equation $d=5$ and $x(1)=0$

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

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

$x(n) = 5(n-1)$ is the recurrance.

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

Given

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

$x(1) = 4$

sub $n=2$

sub $n = 3$

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

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

sub $n=4$

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

The general form of equation is $x(n) = 3^{n-1} \cdot x(1)$

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

$x(n) = 3^{n-1} \cdot 4$ is the recurrance relation.

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

soln

$x(n) = x(n/2) + n$
$x(1) = 1$; $n = 2k$
$x(2k) = x\left(\frac{2k}{2}\right) + 2k$
$x(2k) = xk + 2k$

sub $k=1$

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

sub $k=2$

$x(2 \cdot 2) = x(2) + 2 \cdot 2$

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

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

Assignment - 1

Name: D Lavanya

sub $k=3$

$x(2.3) = x(3) + 2.3$

$x(3) : x(1.5) + 3$

The General equation for given equation is

$x(2k) = x(k) + 2k$.

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

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

$x(1) = 1$; $n = 3k$

$x(3k) = x\left(\frac{3k}{3}\right) + 1$

$x(3k) = xk + 1$

sub $k = 1$

$x(3.1) = x(1) + 1$

$= 1 + 1$

$x(3) = 2$

sub $k = 2$

$x(3.2) = x(2) + 1$

$x(6) = x(2/3) + 1$

sub $k = 3$

$x(3.3) = x(6) + 1$

$= 2 + 1$

$x(9) = 3$

The general equation for given expression is

$x(3k) = 1 + \log_3(k)$

Evalute the following recurrence completely

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

soln

$n = 2k$; $k = \log n$

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

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

sub $k=3$

$x(2.3) = x(3) + 2.3$

$x(3) = x(1.5) + 3$

The General equation for given equation is

$x(2k) = x(k) + 2k$.

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

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

$x(1) = 1$; $n = 3k$

$x(3k) = x\left(\frac{3k}{3}\right) + 1$

$x(3k) = xk + 1$

sub $k = 1$

$x(3.1) = x(1) + 1$

$= 1 + 1$

$x(3) = 2$

sub $k = 2$

$x(3.2) = x(2) + 1$

$x(6) = x(2/3) + 1$

sub $k = 3$

$x(3.3) = x(6) + 1$

$= 2 + 1$

$x(9) = 3$

The general equation for given expression is

$x(3k) = 1 + \log_3(k)$

Evalate the following reccurrence complabls

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

soln $n = 2k$; $k = \log n$

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

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

$T(2 \cdot k) = T(k/2) + 2$ (if k is even)

$T(2 \cdot k) = T\left(\frac{k-1}{2}\right) + 2$ (if k is odd)

$T(2 \cdot k) = T(1) + k$

Recurrences $\Rightarrow T(n) = \Theta(\log n)$

(ii) $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$ where 'c' is a constant and 'n' is the input size.

Soln

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$a = 2, b = 3 \quad f(n) = cn$

master theorem :-

$f(n) = \Theta(n^c)$ where $c < \log_b^a$, then $T(n) = \Theta(n^{\log_b^a})$

$f(n) = \Theta(n^{\log_b^a})$ then $T(n) = \Theta(n \log_b^a \log x)$

$f(n) = \Omega(n^2)$ where $c > \log_b^a$, af $(a/b) \leq kf(n)$

for $k < 1$

$T(n) = \Theta(f(x))$

find $\log_b^a = \log_b^a = \log_3^2$

$f(n) = cn = n\log_b^a$

Recurrence Relation $\Rightarrow T(n) = \Theta(n)$

Consider the following recursion algorithm.

Min1[A[0 ... n-1]] - 1

if n=1 return A[0] - 1

else

temp = Min[A[0, ...n 2]]

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

else

Return A[n-1]

≥ 7

What does the algorithm compute?

=> This algorithm computes the minimum element in an array A of size n using a recursive approach.

=> Base Case:-

If the array has only one element (n==1), It return that single element as the minimum.

=> Recursive Case:-

* If the array has more than one element (n>1) the function makes a recursive call to find the min element in Subarray consisting of the first n-1 elements

* The result of this recursive call ("temp") is then compared to the last element of the current array segment ("A[n-1]")

* The function returns the smaller of these two values.

=> Setup a recursive relation for the algorithm basic operation cont and solve it.

Min1[A[0 ... n-1])

if n=1

return A[0]

else

temp: Min 1 [A[0... n2]) - n-1

if temp <= A[n-1]

return temp

Else

Return A[n-1]

T(n) = No. of basic Operation

if n = 1 then T(1) = 0

T(n) = T(n-1)+1 is the recurrence relations

T(1) = 0

T(2) = T(2-1) + 1

= T(1) + 1

= 0 + 1

T(2) = 1

T(3) = T(3-1) + 1

= T(2) + 1

= 1 + 1

= 2.

T(4) = T(4-1) + 1

= T(3) + 1

= 2 + 1

= 3

T(n) = n-1

Time Complexity = O(n), where n = size of the array

4) Analyze the order of growth.

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

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

$g(n) = 7n$

if $n=1 \Rightarrow F(n) = 2(1)^2+5$ $g(n) = 7(1)$

$= 7$ $= 7$

$n=2 \Rightarrow F(n) = 2(2)^2+5$ $\qquad$ $g(n) = 7(2)$

$= 13$ $\qquad$ $= 14$

$n=3 \Rightarrow F(n) = 2(3)^2+5$ $\qquad$ $g(n) = 7(3)$

$= 23$ $\qquad$ $= 21$

$n=4 \Rightarrow F(n) = 2(4)^2+5$

$= 2(16)+5$

$= 37$

$F(n) \geq g(n).c$ condition satisfies at $n=1$ onwards

So the $\Omega(7n)$ is the occurrence relations.

time complexity is $\Omega(n)$.