CSA-0672 DAA Analytical Assignment-1

K. Mounika
192372072

6/6/24

1) Solve the following recurrence relations

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

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

$x(1) = 0 \rightarrow$

$n=2$

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

$= x(1)+5$

$= 5 \rightarrow (1)$

$n=3$

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

$= x(2)+5$

$= 10 \rightarrow (2)$

$n=4$

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

$= x(3)+5$

$= 10+5$

$= 15$

The general for 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 recurrence

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

Given

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

$x(1) = 4$

Sub $n=2$

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

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

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

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

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

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

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

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

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

$x(2k) = x\left(\frac{2k}{2}\right) + 2k$

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

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

Sub $k=2$
$x(2 \cdot 2) = x(2) + 2 \cdot 2$
$x(2) = x(1) + 2 = 1+2 = 3$
$x(4) = x(2) + 4 = 3+4 = 7$

Sub $k=3$

$x(2\cdot3)=x(3)+2\cdot3$

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

$\therefore$ The general equation for given equation is

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

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

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

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

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

$x(3k)=xk+1$

Sub $k=1$

$x(3\cdot1)=x(1)+1$

$=1+1$

$x(3)=2$

Sub $k=2$

$x(3\cdot2)=x(2)+1$

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

Sub $k=3$

$x(3\cdot3)=x(3)+1$

$=2+1$

$x(9)=3$

The general equation for given expression is

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

Evaluate the following recurrences completely

(i) $T(n)=T(n/2)+1$, where $n=2k$ for all $k\geq0$

Given $n=2k$, i.e $k=\log n$

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

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

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

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

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

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

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

$$T(n) = aT(n/b) + f(n)$$

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

Master's 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 $T(n) = \Theta(n \log_b^a \log n)$

$f(n) = \Omega(n^2)$ where $c > \log_b^a$, $af(n/b) \leq kf(n)$ for $k < 1$

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

Find $\log_b^a \Rightarrow \log_b^a = \log_3^2$

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

Recorrence relation $\Rightarrow T(n) = \Theta(n)$

Consider the following recursion algorithm

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

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) What does this 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 returns 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.

b) Setup a recurrence relation for the algorithms basic operation count and solve it.

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

if n=1

return A[0]

Else

temp = Min1(A[0 ... n-2]) — n-1

if temp <= A[n-1]

return temp

Else

Return A[n-1]

$T(n)$ = No. of basic operations

if $n=1$ then $T(1)=0$

" $T(n) = T(n-1)+1$ " is the recurrence relation.

$T(1) = 0$

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

$= T(1)+1$

$= 0+1$

$T(2) = 1$

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

$= T(2)+1$

$= 1+1$

$= 2$

basic

$T(4) = T(4-1)+1$
$= T(3)+1$
$= 2+1$
$= 3$

$T(n) = n-1$

$\therefore$ Time Complexity : $O(n)$ where n = size of the array

4) Analyze the order of growth

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

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

$g(n) = 7n$

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

$n=2 \Rightarrow f(n) = 2(2)^2+5 = 13$ $\quad g(n) = 7(2) = 14$

$n=3 \Rightarrow f(n) = 2(3)^2+5 = 23$ $\quad g(n) = 7(3) = 21$

$n=4 \Rightarrow f(n) = 2(4)^2+5 = 2(16)+5 = 37$ $\quad g(n) = 7(4) = 28$

$f(n) \geq g(n) \cdot c$ Condition satisfies at $n = \cancel{1}$ onwards

So the $\Omega(7n)$ is the reccurrence relation.

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