Date:- 06/06/24

CSA0672 – DAA

B.Lavanya
192311131

1. Solve the following recurrence relations

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

Given

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

$n = 2$

$$x(2) = x(2-1) + 5$$
$$= x(1) + 5$$
$$= 0 + 5 \quad \text{--- (1)}$$

$n = 3$

$$x(3) = x(3-1) + 5$$
$$= x(2) + 5$$
$$= 5 + 5$$
$$x(3) = 10 \quad \text{--- (2)}$$

$n = 4$

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

The general for the given equation is $x(n) = x(1) + (n-1)d$

In the given equation $d = 5$ and $x(1) = 0$

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

$\boxed{x(n) = 5(n-1)}$ is the recurrence relation.

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 $n=3$

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

$= 3x(2)$

$= \cancel{x}\, 3(12)$

$= 36$

Sub $n=4$

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

$= 3x(3)$

$= 3(36)$

$= 108$

The general farm of the given eq^n is $x(n) = 3^{n-1} \cdot x(1)$

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

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

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

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

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

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

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

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 $k=3$

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

$x(3) = x(1\cdot5)+3$

$\therefore$ The general equation for given expression is

$$\boxed{x(2k) = x(k)+2k}$$

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

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

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

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

$x(3k) = xk+1$

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

$$\boxed{x(3k) = 1+\log_3(k)}$$

② Evaluate the following recurrences completely

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

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

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

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

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

$T(2 \cdot K) = T((K-1)/2) + 2$ (if K is odd)

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

$\Rightarrow$ Recurrences $\Rightarrow \boxed{T(n) = \theta(\log n)}$

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

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

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

Master theorem states:-

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

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

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

for $K < 1$

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

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

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

Recurrence relation $\Rightarrow \boxed{T(n) = \theta(n)}$.

③ Consider the following recursion algorithm

Min 1(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?

b) setup a recurrence relation for the algorithm basic operation count and solve it.

a) ⇒ 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) 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.

$$T(1) = 0$$

$$\begin{aligned} T(2) &= T(2-1)+1 \\ &= T(1)+1 \\ &= 0+1 \end{aligned}$$

$$T(2) = 1$$

$$\begin{aligned} T(3) &= T(3-1)+1 \\ &= T(2)+1 \\ &= 1+1 \\ &= 2 \end{aligned}$$

$$\begin{aligned} T(4) &= T(4-1)+1 \\ &= T(3)+1 \\ &= 2+1 \\ &= 3 \end{aligned}$$

$T(n) = n-1$

$\therefore$ 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 $n=1$ onwards

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

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