Analytical - 1 CSA0669 6/6/24
P. Gunasri

Q) solve the following recurrence relations

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

Sol:- Given,

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

$x(1) = 0$

sub $n=2$

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

sub $n=3$

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

sub $n=4$

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

The general form for the given equation is

$x(n) = x(1) + (n-1)\cdot d$

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 $n>1$ $x(1)=4$

Sol:- 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 $n=3$

$x(3) = 3x(3-1)$
$= 3x(2)$
$= \cancel{6x}$
$= 3(12)$
$= 36$

Sub $n=4$

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

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

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

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

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

Sol:- 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 $k=2$

$x(2\cdot 2) = x(2) + 2\cdot 2$
To find $x(2)$
$x(2) = x(1) + 2 = 1+2 = 3$
$x(4) = x(2) + 4 = 3+4 = 7$

Sub $k=3$

$x(2\cdot 3) = x(3) + 2\cdot 3$
To find $x(3)$
$x(3) = x(1.5) + 3$

$\therefore$ The general equation for given expression is

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

$x(n) = x(n/3)+1$ for $n > 1$, $x(1) = 1$ (Solve for $n = 3k$)

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

$n = 3k$

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

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

Sub. $k = 1$

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

$= 1+1$

$= 2$

Sub $k = 2$

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

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

$\therefore$ The general equation for $\boxed{x(3k) = 1 + \log_3(k)}$

2) Evaluate the following recurrences completely

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

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

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

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

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

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

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

$\Rightarrow$ Recurrency $\Rightarrow$ $\boxed{T(n) = \Theta(\log n)}$

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

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

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

Master theorem states:-

$f(n) = O(n^c)$ 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 $\log_b a \Rightarrow \log_b a = \log_3 2$

$f(n) = \cancel{3n}\ cn = n^{\log_b a}$

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

---

3) Consider the following recursive 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?

Base case:- if $n=1$, return $A[0]$.

This means that if the array has only one element, that element is minimum.

Recursive case:-

Compute 'temp = Min1 (A[0 .... n-2])' which finds the minimum element in the subarray 'A[0...n-2]'.

Compare this minimum element ('temp') with the last element in the array ('A[n-1]'):

* If 'temp' is less than or equal to 'A[n-1]', return 'temp'.
* Otherwise, return 'A[n-1]'.

6) Setup a recurrence relation for the algorithm basic operation and solve it.

sol:- Algorithm for the recurrence relation:-

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

If n=1 return A[0]

Else

temp = Min1 (A[0 ..... n-2])

If tmp <= A[n-1] return temp

Else return A[n-1]

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

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

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

$\vdots$

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

$T(n) = T(1) + (n-1) \cdot 1$

$T(1) = 0$

$T(n) = 0 + (n-1) \cdot 1$

$T(n) = n-1$

$\Rightarrow$ Recurrence relation: $\boxed{T(n) = n-1}$

---

4) Analyze the order of growth.

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

Sol:- $f(n) = 2n^2+5$, $g(n) = 7(n)$

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

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

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

$f(n) \geq g(n) \cdot c$ condition satisfies at $n=1$ onwards so the $\Omega(7 \cdot n)$ is the recurrence relation.

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