Name :- Kishore . R .

Reg No:- 192321146

Course code:- CSA 0666

Course Name :- Design and Analysis of Algorithm

Assignment NO:- 4

Date :- 06. 06. 2024.

Solve the following recurrence relations

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

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

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

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

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

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

$x(2) = 3x(1) = 3 \cdot 4 = 12$

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

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

Thus $x(n) = x(1) \cdot r(n-1)$

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

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

$y(k) = y(k-1) + 2^k$

$y(0) = x(1) = 1$

$y(1) = y(0) + 2^1 = 3 = 1+2$

$y(2) = y(1) + 2^2 = 7 = 1+2+4.$

$y(3) = y(2) + 2^3 = 15 = 1+2+4+8$

$\Rightarrow 2^k - 1.$

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

for $n = 2^k$: $x(n) = 2n - 1$

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

Let denote $x(3^k)$ as $y(k)$

$y(k) = y(k-1) + 1$

$y(0) = x(1) = 1$

$y(1) = y(0) + 1 = 2$

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

$\therefore y(k) = k+1$.

for $n = 3^k$.

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

(2). Evaluate the following recurrences completely (i) $T(n) = T(n/2)+1$, where $n = 2^k$ for all $k \geq 0$.

Substitute $n = 2^k$, we get.

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

$T_k = T_{k-1}+1$ with $T_0 = T(1)$.

$T(1) = T_0 + 1 = T(1)+1$

$T_2 = T_1+1 = T(1)+1+1 = 2$

$T_3 = T_2+1 = T(1)+2+1 = 3$

$\therefore T(k) = T(1)+k$.

$n = 2^k \Rightarrow k = \log_2 n$

we assume $T(1) = 0$, we get.

$T(n) = \log_2 n$.

$T(n) = T(n/3) + T(2n/3) + cn$ where c is constant and the input size.

from master thm we have.

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

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

$\log_b a = \log_3 2 = \dfrac{\log_{10} 2}{\log_{10} 3}$

compare fun with $n^{\log_b a}$.

$f(n) = (n)$,

$n^{\log_b a} = n^{\log_3 2}$

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

③. consider the following recursion algorithm.

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

if n=1 return A [0].

Else temp = min 1 (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 min. value in array A [0 to n-1] .

if n=1, returns only A [0] which is minimum

for n>1, recursively computes the min of first n-1

Thus, algorithm finds minimum element in array.

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

i) Best case:

for n=1, the algorithm Performs constant amount of work say c.

Recursive case:

for $n>1$, the algorithm calls recursively on $A[0.. n-2]$

which has $T(n-1)$ operation.

with base case $T(1)=C$

so, $T(n) = T(n-1)+C$,

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

$T(3) = T(2)+C = 3C$.

$T(n) = nC$

Therefore the algorithm Performs n operation done.

constant c.

so, $T(n) = \Theta(n)$.

---

(4). Analyse the order of growth. use the $\Omega\, g(n)$)

(i) $F(n) = 2n^2+5$ and $g(n) = 7n$.

not ation.

let $f(n) \geq c \cdot g(n)$

$F(n) = 2n^2+5$ $\quad g(n) = 7n$.

for larger value of n, $2n^2$ will dominate

so, $2n^2+5 \geq 7cn$.

for large value of n, 5 can be neglected

$\Rightarrow 2n^2 \geq 7cn$

$\Rightarrow 2n \geq 7c$

$\Rightarrow n \geq 7c/2$

let $c=1$ then $n \geq \frac{7}{2} = n \geq 3.5$.

$n \geq 4$ c taking number. smallest integer 3.5)

$\therefore 2n^2+5 \geq 7n$.

$F(n) = 2n^2+5$ is $\Omega(g(n) = 7n)$ with $c=1$ and $n_0=$)

$F(n) = \Omega(7n)$ large value of n.