ASSIGNMENT-1

Name : Y. Dayakar Reddy

Register no: 19231108S

course code: CSA-0671

Course name: Design of Analysis of Algorithms for Approximation Algorithm.

19>311085
Y. Dayakar Reddy

1. find the efficiency and order of notation for recursive algorithm - factorial of a given no.

General plan:-

1) Integer n
2) multiplication
3) n times
4) $f(n) = f(n-1) + n$

$m(n) = m(n-1) + 1 \rightarrow$ constant k

To compute $f(n-1)$

$n = 0$

$0! = 0$

$m(0) = 0$

5) solving

Pseudo Code:-

```
Algorithm fact (n)
// Problem description : computes fact of n
// Input : Any integer n
// output : factorial of n
    if (n == 0)
     return 1;
Else
    return fact (n-1) * f(n)
```

Substitution methods:-

1) forward substitution 2) Backward substitution

forward substitution:-

$m(n) = m(n-1)+1 \rightarrow (1)$

$m(0) = 0$

$n=1$

$m(1) = m(1-1)+1$

$m(1) = 1$

$n=2$

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

$= 1+1$

$m(2) = 2$

$\vdots \qquad \vdots$

$n=i$

$m(i) = m(n-i)+1$

Backward substitution:-

$m(n) = m(n-1)+1 \longrightarrow (1)$

$m(0) = 0$

$n = n-1$

$m(n-1) = m(n-2)+1 \rightarrow (2)$

sub (2) in (1)

$m(n) = m(n-2)+2 \rightarrow (3)$

$m(n-2) = m(n-3)+1 \rightarrow (4)$

sub (4) in (3)

$m(n) = m(n-3)+3 \rightarrow (5)$

$\vdots \qquad \vdots$

$n = (n-i) = m(n-i-1)+1$

$\boxed{T(n) \in O(n)} \rightarrow$ Time complexity

192211085
Y. Dayakar Reddy

2) Find the efficiency and order of notation for the non recursion Algorithm. find the maximum value in a list.

General plan :-

1) Input
2) Basic operation
3) no. of times.
4) summation Σ
5) solving summation.

Pseudo Code :-

Algorithm max-element (A[0,1,2,...,n-1])
// Problem description
// Input : Given Array
// output : maximum element in the Array

max-value ← A[0]
for i ← 1 to n-1 do
{ if (A[i] > max-value)
max-value ← A[i]
} return max-value.

Iteration :-

{5, 8, 4, 7, 9}
max-value = 5

i = 1

if A[i] > 5

if 8 > 5 satisfies

Iteration - 2 :-

max - value = 8

i = 2

if A[2] > 8

if 4 > 8 not satisfied

return 8

simplarly it compares by Iteration 3, 4 and it find max-value is 9

Time complexity :-

$$c(n) = \sum_{i=1}^{n-1} 1$$

formula :- $\sum_{i=k}^{n} 1 = n-k+1$

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

$$c(n) = n-1$$

$$T(n) \in \Theta(n)$$

3) Explain the steps to solve the Towers of Hanoi problem. And also estimate the order of notation for n disk. using the substitution method for to predict the order of growth.

192311085
Y. Dayakar Reddy

Tower of Honai :-
we have to move the disk from one pole to other by supportive

General plan :-

1) n disk
2) move
3) n time
4) Recurresence relation.
   (i) Reccurence equation
   (ii) Inetial condition.

Pseudo code :-

Algorithm TOH
// Problem description
// input : Any Intern
// output : Tower of Honai n.
If (n==1)
{
write ("Disk mode from A to B")
return
{ // move top n-1 disk from A to B using C
TOH
// move remaining disk.
TOH
}

Recurrence relation :-

If n>1
m(n) = m(n-1) +1+m(n-1).

Initial condition

$n = 1$

$m(1) = 1 \rightarrow$ only one desk contains

Solving:-

forward substitution:-

$m(n) = 2m(n-1) + 1 \rightarrow ①$

$m(1) = 1$

$n = 2 \rightarrow$ sub in equ ①

$m(2) = 2m(1) + 1$

$m(2) = 3$

$n = 3$

$m(3) = 4$

$n = i \quad m(i) = 2m(n-i) + i$

Backward substitution:-

$m(n) = 2m(n-1) + 1 \rightarrow ①$

$m(1) = 1$

$n = n-1$

$m(n-1) = 2m(n-2) + 1 \rightarrow ②$

sub ② in ①

$m(n) = 4m(n-2) + 2 + 1 \rightarrow ③$

$m(n) = 2^i m(n-1) + 2^{i-1} \cdots + 2 + 1$

$$x^{i-1} + x^{i-2} + \cdots + 2 + 1 = \frac{1-x^2}{1-x}$$

$$m(n) = 2^i m(n-i) + \frac{1-2^i}{1-2}$$

$$\hookrightarrow \frac{1-2^i}{-1}$$

$$= 2^i - 1$$

$$m(n) = 2^i m(n-i) + 2^i - 1$$

sub $i = n-1$

$$m(n) = 2^{n-1} m(n-(n-1)) + 2^{n-1} - 1$$

$$m(n) = 2^{n-1} m(1) + 2^{n-1} - 1$$

$$= 2 \cdot 2^{n-1} - 1$$

$$= 2 \cdot 2^n 2^{-1} - 1$$

$$= 2^n - 1$$

$T(n) \in O(2^n)$ → Time complexity.