ASSIGNMENT-1

Name: N. Srinadh

Register no: 192311086

Course code: CSA - 0671

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

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

General Plan:

① Integer N

② Multiplication

③ n times

④ $f(n) = f(n-1) * n$

$m(n) = m(n-1) + 1$ ⟶ 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

① Forward substitution:-

$m(n) = m(n-1) + 1 \quad -①$

$m(0) = 0$

$n=1$

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

$m(1) = 1$

$n=2$

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

$= 1+1$

$m(2) = 2$

⋮ ⋮

$n=i$

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

Backward substitution:-

$m(n) = m(n-1) + 1 \longrightarrow ①$

$m(0) = 0$

$n = n-1$

$m(n-1) = m(n-2) + 1 \quad -②$

Sub ② in ①

$m(n) = m(n-2) + 2 \quad -③$

$m(n-2) = m(n-3) + 1 \quad -④$

Sub ④ in ③

$m(n) = m(n-3) + 3 \quad -⑤$

⋮ ⋮

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

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

(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
2       if (A[i] > max-value)
            max-value ← A[i]
3   return max-value.
```

Iteration :-

{5, 8, 4, 7, 9}

max-value = 5

i = 1

if A(i) > 5

if 8 < 5 satisfied

Iteration - 2 :-

max-value = 8

i = 2

if A(2) > 8

if 4 > 8 not satisfied

return 4

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

Time complexity :-

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

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

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

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

$$c(n) = \in O(n)$$

Explain the steps to solve the Towers of Honai Problem. And also estimate the order of notation for n disk. using the substitution method for to Predict the order of growth.

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 Recurrence relation.
   (i) Recurrence equation
   (ii) Initial condition

Pseudo code :-

```
Algorithm Toh
// Problem description
// input : Any Enter n
// output : Towes of Honai n.
   If (n == 1)
   {
```

Write ("Disk mode 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 disk contains

Solving:-

forword substitution:-

$m(n) = 2m(n-1) + 1 \rightarrow ①$

$m(1) = 1$

$n = 2 \rightarrow$ sub in eqn ①

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

$m(2) = 3$

$n = 3$ $m(3) = 4$

:

n: i $m(i) = 2m(n-i) + i$

$m(n) = 2m(n-1)+1 \longrightarrow ①$

$m(1) = 1$

$n = n-1$

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

Sub ② in ①

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

$\vdots$

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

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

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

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

$= 2^i-1$

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

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.