ASSIGNMENT-1:

Name: B.Hari Sai

Register no: 192511098

course code: CSA-0671

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

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. $F(n) = F(n-1) + n$

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

   To compute one more multiplication.

   $F(n-1)$ $F(n-1)$ by 1

   $n = 0$

   $0! = 0$

   $M(0) = 0 \Rightarrow$ Initial condition

5) solving

Pseudo code:

Algorithm fact(n)

// problem description: Computes facts of n

// Input : Any integer n

// output : factorial of n

```
if (n==0)
    return 1
else
    return fact(n-1) * n
```

Substitution method:

1) forward substitution 2) Backward substitution

Forward substitution:

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

$M(0) = 0$

$n=1 \Rightarrow$ substitute in eq (1)

$M(1) = M(1-1)+1$

$= M(0)+1$

$M(1) = 0+1$

$M(1) = 1$

$n=2 \Rightarrow$ substitute in eq (2)

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

$M(2) = M(1)+1$

$M(2) = 1+1$

$M(2) = 2$

$n=3 \Rightarrow$ substitute in eq (3)

$M(3) = M(3-1)+1$

$= M(2)+1$

$= 2+1$

$M(3) = 3$

$\vdots \qquad \vdots$

$n = i$

$M(i) = M(i-1)+1 \Rightarrow M(n) = M(n-1)+1.$

$m(n) = m(n-1) + 1 \rightarrow ①$

$m(0) = 0$

$m = n-1$

$m(n-1) = m(n-2) + 1 \rightarrow ②$

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

$m(n) = m(n-3) + 3 \rightarrow ⑤$

$\vdots \qquad \vdots$

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

$T(n) \leq O(n) \Rightarrow$ Time complexity

2. find the efficiency and order of notation for the non-recursive Algorithm. find the maximum value in a list.

Ans: General plant

1. Input
2. Basic operation
3. No. of times
4. summations
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: 1

A(0)	A(1)	A(2)	A(3)	A(4)
{ 5	8	4	7	9 }

max_value=5
i=1
if A[1] > 5
if 8 > 5 satisfies.

Iteration 2:

Max_value = 8
i=2
if A(2) > 8
if 4 > 8 not satisfies.

max - value = 8

return 8

similarly it compares by iteration 3,4 and it find max-value is 9.

Time complexity:

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

⇒ one comparison is made with each iteration.

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.

A: Tower of Hanoi: we have to move the disk from one to other by supportive.

General plan:

1) n disk

2) move

3) n times

4) Reccurenence relation. i) Reccurence equation
ii) Initial condition.

pseudo code

Algorithm TOH (n, A, B, C)

// problem Description

// Input: Any integer n

// output: Tower of Hanai n

if (n = = 1)

{ write ("Disk mode from A to B")

return

{ // move top n-1 disk from A to B using C

TOH (n-1, A, B, C)

// move remaining disk

TOH (n-1, B, C, A)

}

Reccurrence relation:

if n>1

$M(n) = M(n-1)+1 + M(n-1)$ (To move disk from B to C)

=> 2 $M(n-1)+1$

Initial condition $n=1$

$m(1)=1 \Rightarrow$ only one disk contains

Solving

forward substitution

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

$m(1)=1$

$n=2 \Rightarrow$ sub in equ (1)

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

$m(2)=3$

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

$\vdots \qquad \vdots$

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

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)

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

$\vdots \qquad \vdots$

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

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

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

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

$$= 2^i - 1$$

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

sub $i = n-1$

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

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

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

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

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

$$= 2^n - 1$$

$T(n) \in O(2^n) \Rightarrow$ time complexity.