Design and Analysis of Algorithms for Approximation Algorithm

CSA-0671

Name : Jeevitha. S

Reg no : 192324020

Dept : B-tech (AI&DS)

Date-Day : 05/06/2024 Wednesday

ASSIGNMENT-01

(1) Find the efficiency and order of notation for recursive algorithm - Factorial of a given number.

General plan

(i) Input : Any integer n.

(ii) Basic operation : Multiplication

(iii) n times (repeation of basic operation)

(iv) $F(n) = F(n-1) * n$

Recurrence relation $\Rightarrow M(n) = M(n-1) + 1$ $\quad M(0) = 0 \Rightarrow$ Initial condition.

(v) Pseudo code of factorial

```
Algorithm fact (n)
// To find factorial of given number.
// input : Any integer n.
// output : Factorial of n.
    if (n=0) then return 1
    else return fact(n-1)*n
```

(vi) Solve the recurrence relation. (Two types).

$\Rightarrow$ Forward substitution

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

$n=0,\ M(0) = 0$

$n=1,\ M(1) = M(0) + 1 = 0+1 = 1$

$n=2,\ M(2) = M(1) + 1 = 1+1 = 2$

$n=3,\ M(3) = M(2) + 1 = 2+1 = 3$ ..

$n=i,\ M(i) = M(i-1) + 1$

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

⇒ Backward Substitution

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

$M(0) = 0$

$n = n-1$ in (1)

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

sub (2) in (1)

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

$n = n-2$ in (1)

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

sub (4) in (3)

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

i^{Th} rec call

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

$n = i$

$M(i) = M(i-i) + i$

$M(i) = M(0) + i$

$\boxed{M(i) = i}$

$\boxed{M(n) = n}$

Efficiency Analysis

Time complexity: $O(n)$

space Complexity: $O(n)$

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 predicate the order of growth.

General plan

(i) Input: n disk

(ii) Basic operation: moving.

(iii) n times

(iv) Recurrence relation

→ Rec. equation

→ Initial condition.

(v) solving the rec. con

pseudo code

```
Algorithm TOH (n, A, B, C)
//problem discription (To move disk to axillary pole).
//Input: n, A, B, C.
//Output: n times.
if (n==1) then
{ write ("Disk moved from A to B")
  return
}
{ // move top n-1 disk from A to B axillary C
  TOH (n-1, A, B, C)
  // move remaining disk
  TOH (n-1, B, C, A)
}
```

Recurrence relation.

If $n > 1$

$$M(n) = \underbrace{M(n-1)}_{\text{To move } (n-1) \text{ disk from A to B}} + \underbrace{1}_{\text{To move largest disk from A to C}} + \underbrace{M(n-1)}_{\text{To move disk from B to C.}}$$

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

Algorithm

```
def find_max (list):
    max_value = list [0]
    for value in list [1:]:
        if value > max_value:
            max_value = value
    return max_value.
```

Efficiency Analysis

Time Complexity:

* The algorithm iterates through each element in the list exactly once.
* Let n be the number of elements in the list.
* Each Comparison operation (checking if the current value is greater than 'max_value') takes constant time, $O(1)$.
* Therefore, the total time complexity is $O(n)$.

Space Complexity:

* The algorithm uses a constant amount of extra space for the 'max_value' variable.
* No additional space is used that grows with the input size.
* $\therefore$, the space complexity is $O(1)$

Time Complexity : $O(n)$

Space Complexity : $O(1)$

tial condition, n=1
$M(1) = M(1-1) + 1 + M(1-1)$
$M(1) = 1 \rightarrow$ 1movement (A to B).

Solving

Substitution method

(i) Forward

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

$\boxed{M(1) = 1} \rightarrow$ Initial condition

n=2 sub in 1

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

$\boxed{M(2) = 3}$

n=3 Sub in 1

$\boxed{M(3) = 7}$

(ii) Backward

$\boxed{M(n) = 2M(n-1) + 1} \rightarrow 1$

$\boxed{M(1) = 1}$

n = n-1 in 1

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

$\boxed{M(n-1) = 2M(n-2) + 1} \rightarrow 2$

Sub 2 in 1

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

$M(n) = 4M(n-2) + 2 + 1$

$\boxed{M(n) = 4M(n-2) + 3} \rightarrow 3$

n=n-2 in 1

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

$\boxed{M(n-2) = 2M(n-3) + 1} \rightarrow 4$

sub 4 in 3

$M(n) = 4(2M(n-3)+1)+3$

$M(n) = 8M(n-3)+4+3$

$\boxed{M(n) = 8M(n-3)+7} \rightarrow 5$

$M(n) = 2^3 M(n-3) + 2^2 + 2 + 1$

$\vdots$

$M(n) = 2^i M(n-1) + \underbrace{2^{i-1} + \dots + 2 + 1}$.

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

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

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

$i = n-1$

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

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

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

$= 2^{n-1} \cancel{M(1)} + 2^{n-1} - 1 \quad (\because M(1) = 1)$

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

$= 2 . 2^{n-1} - 1$

$= 2 . 2^n . 2^{-1} - 1$

$\boxed{M(n) = 2^n - 1.} \rightarrow$ order of Notation.

Time Complexity: $O(2^n)$.

Space Complexity: $O(n)$.