① Kimmi Kumari

Date ____
Page ____

★ Complexity analysis of recursive programs

Time complexity ↙ ↘ space complexity

★ Time Complexity

```
Factorial (n)
{
  if (n == 0)
      return 1
  else
      return n * factorial(n-1)
}
```

assume each simple operation ⇒ cost of 1 unit.

$n *$ → 1 unit; factorial(n-1) → 1 unit.

⇓

$T(n) = T(n-1) + 3$ if $n > 0$

(T(n): Time taken; 3: unit of cost)

$T(0) = 1$

$T(n) = T(n-1) + 3$

$T(n-1) = T(n-2) + 6$

$T(n-2) = T(n-3) + 9$

⋮

$T(n-(k-1)) = T(n-k) + 3k$ → represent with T(0): $n-k=0$, $n=k$

$n-k=0$, $k=n$

$T(n) = T(0) + 3n$

$T(n) = 3n+1$

$T(n) \approx O(3n+1)$

(2)

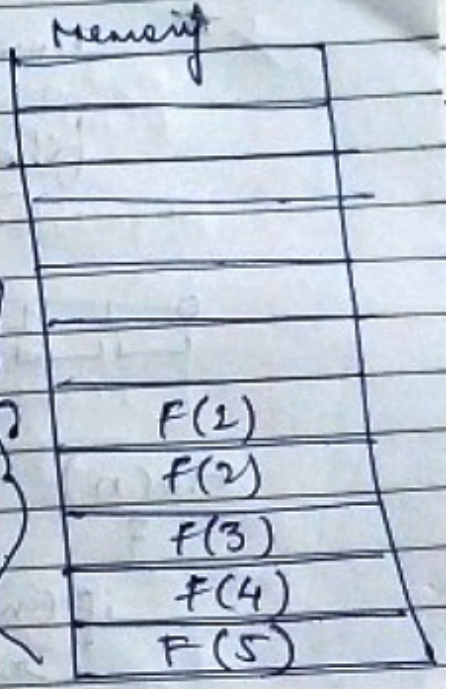
☆ Space complexity

Recursion tree → max depth is n if input is n

F(5) L0
↓
F(4) L1
↓
F(3) L2
↓
F(2) L3
↓
F(1) L4
↓
F(0) L5

max depth = 5

Space O(n)



Once f(0) is executed pop out of stack & same with others

Immi Cumary

☆ Mathematical / Time complexity of Non-Recursive Algorithms ....

Algorithms MaxElement (A[0, n-1])

```
MaxVal ← A[0]
for i ← 1 to n-1 do
    if A[i] > maxval
        MaxVal ← A[i]
return MaxVal .
```

max

0	1	2	3	4
10	20	30	40	50

10 < 20 → max 20
20 < 30 → max 30
30 < 40 → max 40
40 < 50 → Max 50

(i) → n → size of input.

(ii) basic operation ⇔ comparison

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

$C(n) \Leftrightarrow (n-1) - 1 + 1$.

$C(n) = n-1$.

$C(n) \approx O(n)$. Ans

(iii) Since, the above algorithm depends only on input, so no need to go for BC, WC, AC.

(5) Using formulas & rules of establish order of growth.

General plan

(1) Decide on parameter indicating input's size

(2) Identify the basic operation

(3) Check whether no. of times basic operation executed on additional property if so go for WC, BC, AC off

(4) Set up a sum expressing no. of times basic operation is exe

Ex :→ 2

Algorithm Unique Element (A[0 n-1])

```
for i ← 0 to n-2 do
    for j ← 1 to n-1 do
        if ( A[i] == A[j] )
            return false
else
    return true
```

n ⇒ input

basic operation ⇒ comparison

$$C_{worst}(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

$$= \sum_{i=0}^{n-2} n-1-i+1 =$$

$$= \sum_{i=0}^{n-2} n-1 = \frac{(n-2)(n-1)}{2}$$

$$= \frac{1}{2}(n(n)-1)$$

$$= O(n^2) \text{ Ans}$$

0	1	2	3	4
10	20	30	40	50

we will have best-case such as if first two elements are unique else worst-case if last two elements are unique. or no unique elem