**Design and Analysis of Algorithms**

**Frequency Count Method**

Ex - 1 :- Algorithm sum(A , n)

Input : A – array of n elements

1 . s <- 0 ----------- 1 time

2 . for(i=0 ; i < n ; i++) -------- (i=0) -> 1 time , (i<n) -> n + 1 times and (i++) -> n times , But we will consider only (n+1)times

3 . s <- s + A[i] ---------- n times

4 .end for

5 .return s ------ 1 time

Analyzation :-

**TIME COMPLEXITY**

-Each statement will take 1 unit of time .

Hence , total unit of time taken = 1 + (n+1) + n + 1 = (2n + 3) unit of time , OR

Time-function , f(n) = 2n + 3 , since the degree of the polynomial is 1

hence , ***time-complexity = O(n)***

**SPACE COMPLEXITY**

-Each variable will take 1 word of memory

Hence , A takes n words , and

n , s , i will take 1 word each

hence , total space , S(n) = n+3 , since its degree is also 1

***space-complexity = O(n)***

Ex – 2 : - Analyze the time and space complexity of an algorithm of sum of two n\*n matrices

Algorithm add(A,B,n)

Input : A , B – Two square matrices with n\*n dimensions

1 . for(i=0;i<n;i++) ------------ (n+1) times

2 . for(j=0;j<n;j++) ---------- n \* (n+1) times

3 . C[i,j] = A[i,j] + B[i,j] ------ n \* n times

4 . end for

5 . end for

**TIME COMPLEXITY**

Hence , Time function f(n) = 2n2 + 2n + 1 , since degree = 2

***Time-complexity = O(n2)***

**SPACE COMPLEXITY**

A , B , C - n\*n words each ; i , j , n – 1 word each

Hence , Space function S(n) = 3 n2+ 3 , degree = 2

***Space – complexity = O(n2)***

Ex – 3 :- Analyze the algorithm of multiplication of two square matrices of n\*n dimension .

Algorithm product(A,B,n)

1 . for(i=0;i<n;i++) --------- (n+1) times

2 . for(j=0;j<n;j++) -------(n+1)\*n times

3 . C[i,j] = 0 ; --------n \* n times

4 . for(k=0;k<n;k++) -------n\*n\*(n+1) times

5 . C[i,j] = C[i,j] + A[i,k]\*B[k,j] -----------n\*n\*n times

6 . end for

7 . end for

8 . end for

Time-function f(n) = 2 n3 + 3 n2 + 2n + 1 , degree = 3

***Time-complexity = O(n3)***

Space-function S(n) = 3 n2 + 4 , degree = 2

***Space – complexity = O(n2)***