Assignment - 2

Q1→ What is meaning of searching. Explain following searching methodes:

1. Linear Search
2. Binary Search

Also find out time complexity of each method.

Ans→ Searching is the process of finding a given value position in a list of values. It decides whether a search key is present in the data or not. It is the algorithmic process of finding a particular item in a collection of items. It can be done on internal data structure or on external data structure.

- Linear Search

It is the most basic type of searching algorithm. A Linear search sequentially moves through our collection (or data structure) looking for a matching value... n complexity terms this is an O(n) search, the time taken to search the list at the same rate as the list does.

Time Complexity of Linear Search.

$$f(n) = 1 \cdot \frac{1}{n} + 2 \cdot \frac{1}{n} + \dots + n \cdot \frac{1}{n} \cdot 0 = (1+2+\dots+n) \cdot \frac{1}{n}$$

$$= \frac{n(n+1)}{2} \cdot \frac{1}{n} = \frac{n+1}{2} = O(n)$$

~~$= \frac{n^2+n}{2} \cdot \frac{1}{n}$~~

Binary Search

It is also known as half-interval search, logarithmic search, or binary chop, is a search algorithm that finds the position of a target value within a sorted array. Binary Search compares the target value to the middle element in the array.

Time Complexity of Binary Search

Q2→ Write an Algorithm of Bubble sort method Also prove that complexity of this algorithm is $O(N^2)$, where N is no. of elements in array?

Ans→

BUBBLE (DATA, N)

// Here DATA is an array of N elements. This algorithm sorts the elements in DATA

1. Repeat steps 2 and 3 for K=1 to N-1.
2. set PTR := 1.
3. Repeat while PTR ≤ N-K:
   (a) if DATA [PTR] > DATA [PTR + 1], then
   Interchange DATA [PTR] and DATA [PTR +1].
   [End of if structure]
   (b) set PTR := PTR +1
4. Exit.

Complexity of Bubble Sort Algorithm

$$= f(n) = (n-1) + (n-2) + \ldots + 2 + 1$$

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

$$= O(n^2).$$

Q3 → Write a program to perform selection sort method. Also explain with example?

Ans →

```
#include <stdio.h>
#include <conio.h>

int main()
{
    int array[100], n, c, d, position, swap;
    printf("Enter number of elements\n");
    scanf("%d", &n);

    printf("Enter %d integers\n", n);
    for (c=0; c<n; c++)
    {
        scanf("%d", &array[c]);
    }
    for(c=0; c<(n-1); c++)
    {
        position = c;
        for(d=c+1; d<n; d++)
        {
            if(array[position]>array[d])
                position=d;
        }
        if(position!=c)
        {
            swap=array[c];
            array[c]=array[position];
```

```
            array [position] = swap;
        }
    }
    printf("Sorted list in ascending order:\n");

    for (c=0; c<n; c++)
    {
        printf("%d\n", array[c]);
    }
    return 0;
}
```

Q4→ Write an algorithm to find out sum of elements in given array where size of array is size N, prove that complexity of this algo is O(N).

Ans→

ARRAY-SUM(A, N)
[Here A is an Array of N elements]
Step 1. declare I, SUM. Step 2. Set SUM=0,
Step 3. Repeat step 4 for I=0 to N-1
Step 4. set SUM = SUM + A[I]
Step 5. Write SUM
Step 6. Exit

Complexity

$C_1 + C_2 + (n+1)C_3 + nC_4 + C_5 + C_6$.

→ $1 + 1 + 1(n+1) + 1n + 1 + 1$

→ $1 + 1 + n + 1 + 1n + 1 + 1$

→ $2n + 5 \Rightarrow O(N)$ →

Q.5 ⇒ Write a algorithm to perform insertion sort algorithm. Also prove that time complexity of this algo in Best case O(N) and in worst case $O(n^2)$. where n is a number of elements in array?

Ans →

INSERTION SORT (A, N)

Step 1. declare I, J, KEY
Step 2. Repeat step 3 to 6 for I = 1 to N-1
Step 3. set KEY = A[I]
J = I - 1
Step 4. Repeat step 5 while J >= 0 && A[J] > KEY
Step 5. A[J+1] = A[J]
J = J - 1
Step 6. set A[J+1] = KEY
Step 7. Write A[]

Best Case :-

$$f(n) = c_1 + c_2 n + c_3(n-1) + c_4 \sum_{i=1}^{n-1} t_i + c_5 \sum_{i=1}^{n-1} t_{i-1} + c_6(n-1) + c_7$$

$$\Rightarrow c_1 + c_2 n + c_3(n-1) + c_4(n-1) + c_6(n-1) + c_7$$
$$\Rightarrow 1 + n + 1(n-1) + 1(n-1) + 1(n-1) + 1$$
$$\Rightarrow 4n - 1$$
$$\Rightarrow O(n).$$

Worst Case :-

$$\Rightarrow c_1 + c_2 n + c_3(n-1) + c_4\left[\frac{n(n+1)}{2} - 1\right] +$$

$$c_5\left[\left(\frac{n(n+1)}{2} - 1\right) - 1\right] + c_6(n-1) + c_7$$

$$\Rightarrow 1 + n + n - 1 + \left[\frac{n^2 + n - 2}{2}\right] + \left[\left[\frac{n^2+n-2}{2}\right] - 1\right]$$

$$+ n - 1 + 1$$

$$\Rightarrow 3n + \frac{n^2 + n - 2}{2} + \frac{n^2+n-2}{2} - 1$$

$$\Rightarrow \frac{6n + n^2 + n - 2 + n^2 + n - 2 - 2}{2}$$

$$\Rightarrow \frac{6n + 2n^2 + 2n - 6}{2} = \frac{2n^2 + 8n - 6}{2}$$

$$\Rightarrow O(n^2)$$

Q6→ Find the complexity of the below program -

```
function (int n)
{
    if (n == 1)
    return;
    for (int i = 1 ; i <= n; i++)
    {
        for (int j = 1; j <= n ; j++)
        {
            printf(" * ");
            break;
        }
    }
}
```

Algorithm

FUNCTION(N)
// here function is a function of N argument
Step 1. declare I, J
Step 2. if n == 1
then
return
Step 3. Repeat step 4 and Step 5 for I = 1 to I <= N
Step 4. Repeat step 5 for J = 1 to J <= N
Step 5. write *
Step 6. exit

TIME COMPLEXITY

$\Rightarrow c_1 + c_2 + c_3(n+1) + c_4(n+1) + c_5$

$\Rightarrow 1 + 1 + 1(n+1) + 1(n+1) + 1$

$\Rightarrow 1 + 1 + n + 1 + n + 1 + 1$

$\Rightarrow 2n + 5$ $\Rightarrow O(n)$.