Class: SE_Computer Subject: AOA RollNo_Name: 55_AdnanShaikh

EXPERIMENT NO- 3

AIM: Implementation and analysis of Binary search.

PROBLEM STATEMENT :

WAP to search given number using Binary search algorithm.

Resource Required: Pentium IV, Turbo C, Printer, Printout Stationary

THEORY:

A binary search algorithm is a technique for finding a particular value in a linear array, by ruling out half of the data at each step, widely but not exclusively used in computer science.

A binary search finds the median, makes a comparison to determine whether the desired value comes before or after it, and then searches the remaining half in the same manner. Another explanation would be: Search a sorted array by repeatedly dividing the search interval in half Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half Otherwise, narrow it to the upper half.

1. **Algorithm :**

```
Sort an array a[j] in increasing order.
  1. Read 'n' elements in the array , say 'a '.
  2. Read the element to be searched, say 'KEY '.
  3. low = 0 , high = n-1
  4. while(high>=low) perform
        following steps. Find the
        middle element.
            mid = (low + high )/ 2
  5. if ( KEY= a[mid] )
        print (' element found at mid position)
  6. if KEY < a[mid] then search for target in
        a[low] to a[mid-l] else search for target in
        a[mid+l] to a[high].
7.  if (KEY < a[Mid_position])
        high = mid_position – 1;
    else
        low = mid_position + 1;
8.  Stop.
```

Input:

35 23 86 97 54

Key=97

Output:

Element found at 3 position.

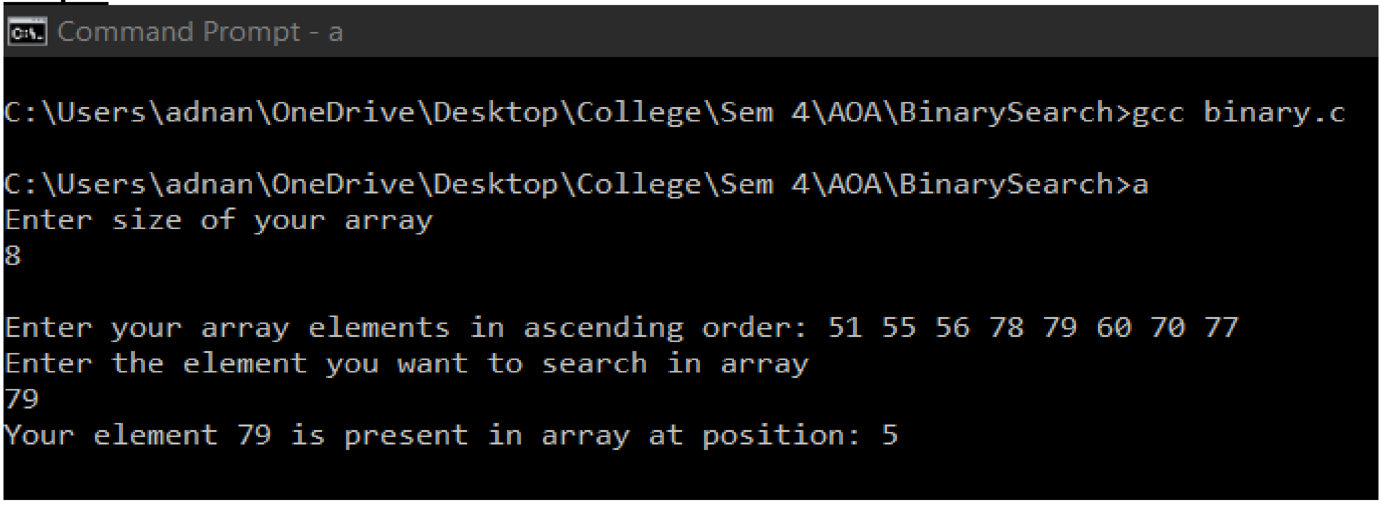
CONCLUSION: The method is called is binary search because at each step, we reduce the length of the table to be searched by half and the table is divided into two equal parts. Binary Search can be accomplished in logarithmic time in the worst case, i.e., $T(n) = \theta(\log n)$. This version of the binary search takes logarithmic time in the best case.

Code:

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

int binarysearch(int arr[],int low,int high,int search);


int main()
{
   int z,search,position;

   printf("Enter size of your array \n");
   scanf("%d",&z);

   int arr[z];

   printf("\nEnter your array elements in ascending order: ");

   for (int i = 0;i<z;i++)
   {
      scanf("%d",&arr[i]);
   }
   printf("Enter the element you want to search in array \n");
```

```
    scanf("%d",&search);

    position = binarysearch(arr,0,z-1,search);

    if(position == -1)
    {
        printf("Your element is not in array");
    }
    else
    {
        printf("Your element %d is present in array at position:
%d",search,(position+1));
    }

    getch();
    return 0;
}


int binarysearch(int arr[],int low,int high,int search)
{
    if(low<=high)
    {
        int mid = (low+high)/2;

        if(arr[mid] == search)
            return mid;
        else if(arr[mid] > search)
            return binarysearch(arr,low,mid-1,search);
        else
            return binarysearch(arr,mid+1,high,search);
    }
    else
        return -1;
}
```

Output:



```
C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\BinarySearch>gcc binary.c

C:\Users\adnan\OneDrive\Desktop\College\Sem 4\AOA\BinarySearch>a
Enter size of your array
8

Enter your array elements in ascending order: 51 55 56 78 79 60 70 77
Enter the element you want to search in array
79
Your element 79 is present in array at position: 5
```