**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Computer Engineering Department**

Program: B.Tech. Sem V

**Course: Design and Analysis of Algorithms**

**List of Experiments**

w.e.f. 1th Jul 2020

**Faculty:** Abhay Kolhe

PART B

|  |  |
| --- | --- |
| Roll No. B032 | Name: Naman Garg |
| Class : B | Batch : B2 |
| Date of Experiment: 14/07/2020 | Date of Submission: 06/08/2020 |
| Grade : | Time of Submission: |
| Date of Grading: |  |

**B.1 Software Code written by student:**

# naman garg B032

# lab 2 DAA

# AIM: Implementation of Linear Search and Binary Search Technique..

#function of linear search accepts an array and a key

def lsearch(arr, x):

    comp = 0

    #we simplly iterate over the array linearly and while we comapare each element we increment the counter

    for i in range(len(arr)):

        comp += 1

        #once the key is found amongst the array, we exit the loop and return the position

        if arr[i] == x:

            return "the element found at index "+str(i) + " no of comps taken = "+str(comp)

    #this return statement is executed if we dont find the element in the array

    return 'not found' + " no of comps taken = "+str(comp)

#function for binary search, accepts an array and a key

def bsearch(arr, x):

    #we initialize the low, high and mid values of indexes of the array

    low = 0

    high = len(arr) - 1

    mid = 0

    comp = 0

    #if high exceeds low, we have searched the entire array and not found our element therefore we finish the loop

    while low <= high:

        #getting mid index of array/subarray

        mid = (high + low) // 2

        #key is too big, therefore we cut the array in half and use the upper half

        if arr[mid] < x:

            low = mid + 1

            comp += 1

        #key is too smaller, therefore we cut the array in half and use the lower half

        elif arr[mid] > x:

            high = mid - 1

            comp += 1

        else:

            #this statement is executed if the key is found at arr[mid]

            return "element found at position "+str(arr.index(x)) + " no of comparisions reqd =" + str(comp+1)

    #if the element is not found in the array at all

    return "element not found" + " no of comparisions reqd =" + str(comp)

#driver code

if \_\_name\_\_ == '\_\_main\_\_':

    print("enter the elements of the array that is to be linear searched")

    #taking list and splitting it up.

    l = list(map(int, input().split()))

    #accepting a key

    key = int(input("enter the element you want to linear search for\n"))

    print(lsearch(l, key))

    print("enter the elements of the array that is to be binary searched (enter sorted integer data or else we'll sort it for you)")

    l = list(map(int, input().split()))

    #i sort the elements regardless of the input just for the sake of user friendliness

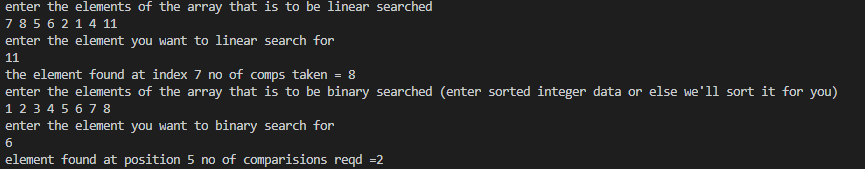
    l.sort()

    key = int(input("enter the element you want to binary search for\n"))

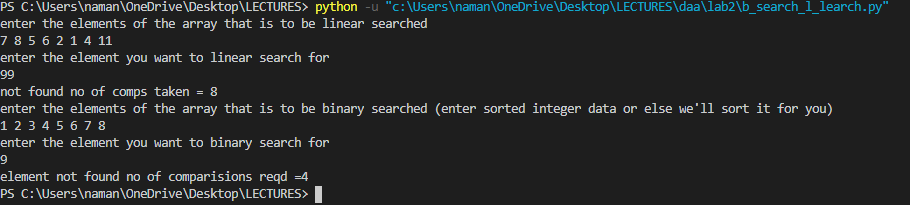
    print(bsearch(l, key))

**B.2 Input and Output:**

**THE ELEMENT IS FOUND::**

****

**THE ELEMENT IS NOT FOUND::**

****

**B.3 Observations and learning:**

I observed that the main difference between Binary in linear searches that linear search works with more versatility of data whereas binary search needs to have specific sorted data therefore if we have a reliable stream of sorted numbers using binary search would be advisable whereas if we are not sure of the sortedness of the data we should use linear search.

**B.4 Conclusion:**

In this lab we implemented two searching techniques linear and binary search both had their pros and cons therefore both have their specific use cases. in production both cannot really be compared however disregarding the requirement of sorted data binary search is faster however linear search is more versatile

**B.5 Question of Curiosity**

Q.1 Identify the applications of Linear Search Technique.

Linear search is useful wherever we have uncertainty based on the shortness of the data if there is unordered data that that is present and we need to find a certain element we use linear search

for example if there is Roll numbers of students in unordered fashion in a certain Data and we need to find some specific roll number we would use linear search

Q.2 Identify the applications of Binary Search Technique.

binary search is useful wherever we have pre sorted or ordered data

therefore binary search is useful in preprocess data that is a data is from more reliable stream or from a source which is reliable in nature

Q.3 Comment on the performance of Linear Search and Binary Search Technique, after

filling up the following table.

|  |  |
| --- | --- |
| **Array Size(n)** | **Data** |
| 4 | 1, 3, 5, 6 |
| 8 | 3, 4, 5, 6, 1, 2, 7, 8 |
| 16 | 3, 4, 5, 6, 1, 2, 7, 8, 1, 3, 5, 6, 6, 1, 2, 7 |
| 32 | 69, 58, 68, 5, 20, 97, 25, 12, 8, 55, 74, 78, 72, 49, 43, 2, 7, 39, 90, 59, 9, 98, 33, 36, 80, 60, 19, 32, 51, 50, 23, 94 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Array Size**  **(n)** | **Binary Search** | | | | | | | **Linear Search** | | | | | | |
| **Best Case** | | **Avg. Case** | | **Worst Case** | | | **Best Case** | | **Avg. Case** | | **Worst Case** | | |
| **Key** | **NoC+** | **Key** | **Key** | **NoC+** | | **NoC+** | **Key** | **NoC+** | **Key** | **Key** | **NoC+** | **NoC+** | |
| 4 | 3 | 1 | 4 | 3 | 1 | 4 | | 3 | 1 | 4 | 3 | 1 | | 4 |
| 8 | 4 | 1 | 8 | 4 | 1 | 8 | | 4 | 1 | 8 | 4 | 1 | | 8 |
| 16 | 4 | 1 | 16 | 4 | 1 | 16 | | 4 | 1 | 16 | 4 | 1 | | 16 |
| 32 | 49 | 1 | 32 | 49 | 1 | 32 | | 49 | 1 | 32 | 49 | 1 | | 32 |

NoC+  =Number of Comparisons

comment :

binary search is faster since we have presorted data and linear search is slower

binary search has time complexity OlogN and linear search has time complexity oN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*