

DATA STRUCTURES LAB

Experiment 1: Searching Algorithms

Project By:

Mohammed Rabeeh

Roll No: 35

TVE18CS036

Contents

1	Aim	2
2	Problem Description	2
3	Algorithm	2
3.1	Linear Search	2
3.2	Binary Search	3
4	Program Code	3
5	Output	6
6	Result	6

1 Aim

To implement **Linear Search** and **Binary Search**(Iterative and Recursive) in a C Program.

2 Problem Description

Searching algorithm is any algorithm which is used to retrieve information stored within some data structure. Search algorithms can be classified into two.

Linear search algorithms check every element in the data structure in a linear fashion to find the search value(key).

Binary Search or half interval searches, repeatedly target the center of the search structure and divide the search space in half . The only prerequisite for binary search is that the search array must be a sorted one.

In this experiment, we will be using array data structure to test both Linear Search and Binary Search.

3 Algorithm

3.1 Linear Search

1. Take input array and the element to search x.
2. Start from the left most item of the array and compare each to x using a loop.
3. If element matches with x, return the index.
4. If no element matched with x, return -1

3.2 Binary Search

1. Take input array and the element to search x.
2. Compare x with the middle element.
3. If x matches with middle element, we return the mid index.
4. Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.
5. Else (x is smaller) recur for the left half.

4 Program Code

```
#include<stdio.h>
#define MAX_SIZE 1000

void sort(int arr[], int n) {
    int i, j, temp;
    for(i = 0; i < n-1; i++) {
        for(j = 0; j < n-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

int lsearch(int search, int arr[], int n) {
    int i;
    for(i = 0; i < n; i++) {
        if(arr[i] == search) {
            return i;
        }
    }
    return -1;
}
```

```

int bsearchi(int search, int arr[], int left, int right) {
    int middle;
    while(left<=right) {
        middle = (left + right)/2;
        if(arr[middle] == search) {
            return middle;
        } else if(arr[middle] <= search) {
            left = middle + 1;
        } else {
            right = middle - 1;
        }
    }
    return -1;
}

int bsearchr(int search, int arr[], int left, int right) {
    int middle;
    middle = (left + right)/2;
    if(left > right) {
        return -1;
    }
    if(arr[middle] == search) {
        return middle;
    } else if(arr[middle] <= search) {
        return bsearchr(search, arr, middle + 1, right);
    } else {
        return bsearchi(search, arr, left, middle - 1);
    }
}

void readElems(int arr[], int n) {
    int i;
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
}

void main() {

```

```

int arr[MAX_SIZE], n, search, option, result;
printf("\n1. Linear search\n2. Binary Search(Iterative)\n3.
    Binary Search(Recursive)\nChoose an option: ");
scanf("%d", &option);
switch(option) {
    case 1:
        printf("Enter number of elements: ");
        scanf("%d", &n);
        printf("Enter the numbers: ");
        readElems(arr, n);
        printf("Enter the number to be searched: ");
        scanf("%d", &search);
        result = lsearch(search, arr, n);
        if(result == -1) {
            printf("The number %d was not found.\n", search);
        } else {
            printf("The number %d was found at index %d.\n", search,
                result + 1);
        }
        break;
    case 2:
        printf("Enter number of elements: ");
        scanf("%d", &n);
        printf("Enter the numbers: ");
        readElems(arr, n);
        printf("Enter the number to be searched: ");
        scanf("%d", &search);
        sort(arr, n);
        result = bsearchi(search, arr, 0, n-1);
        if(result == -1) {
            printf("The number %d was not found.\n", search);
        } else {
            printf("The number %d was found at index %d.\n",
                search, result + 1);
        }
        break;
    case 3:
        printf("Enter number of elements: ");
        scanf("%d", &n);
        printf("Enter the numbers: ");

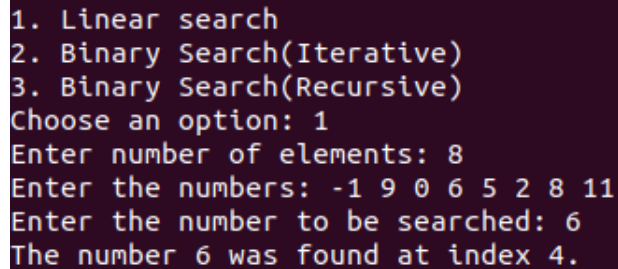
```

```

        readElems(arr, n);
        printf("Enter the number to be searched: ");
        scanf("%d", &search);
    sort(arr,n);
    result = bsearchr(search, arr, 0, n-1);
    if(result == -1) {
        printf("The number %d was not found.\n", search);
    } else {
        printf("The number %d was found at index %d.\n",
            search, result + 1);
    }
    break;
default:
    printf("Invalid Option!\n");
    break;
}
}

```

5 Output



```

1. Linear search
2. Binary Search(Iterative)
3. Binary Search(Recursive)
Choose an option: 1
Enter number of elements: 8
Enter the numbers: -1 9 0 6 5 2 8 11
Enter the number to be searched: 6
The number 6 was found at index 4.

```

6 Result

The searching algorithms **Linear Search** and **Binary Search** were implemented in C and the output was verified.