Symbiosis Institute of Technology

Faculty of Engineering

CSE- Academic Year 2024-25

Data Structures – Lab Batch 2023-27

Lab Assignment No:- 1

Name of Student	Faheemuddin Sayyed
PRN No.	23070122196
Batch	23-27
Class	CSE C-1
Academic Year & Semester	SY 24-25
Date of Performance	18/07/24
Title of Assignment:	A. Implement following searching algorithm: Linear search with multiple occurrences B. Implement following searching algorithms in menu: 1. Binary search with iteration 2. Binary search with recursion
Theory Questions:	1. Prepare table for following 5 different searching algorithms for their best case, average case and worst case time complexities. 2. Apply binary search on the input data set. Show all steps. 3. Compare linear search and binary search 4. Why is the complexity of Binary search O(log n)? Answer 1:

Algorithm	Best Case	Average Case	Worst Case
Linear Search	O(1)	O(n)	O(n)
Binary Search	O(1)	O(log n)	O(log n)
Jump Search	O(1)	O(√n)	O(√n)

Interpolation Search	O(1)	O(log log n)	O(n)
Exponential Search	O(1)	O(log n)	O(log n)

Answer 2:

1. **Initial Array:** [3, 6, 8, 12, 14, 17, 25, 29, 31, 35]
   - Low = 0, High = 9, Mid = (0 + 9) / 2 = 4
   - Mid value = 14
   - 17 > 14, search in right half.
2. **New Array:** [17, 25, 29, 31, 35]
   - Low = 5, High = 9, Mid = (5 + 9) / 2 = 7
   - Mid value = 25
   - 17 < 25, search in left half.
3. **New Array:** [17]
   - Low = 5, High = 6, Mid = (5 + 6) / 2 = 5
   - Mid value = 17
   - 17 == 17, target found at index 5.

Answer 3:

Linear Search:

- Scans each element in the array sequentially.
- Time Complexity: O(n) in the worst case.
- Does not require sorted data.
- Simple implementation.

Binary Search:

- Divides the array into halves to search for the target.
- Time Complexity: O(log n) in the worst case.
- Requires sorted data.
- More efficient for large datasets.

Answer 4:

- Binary Search works by repeatedly dividing the search interval in half.
- Each comparison reduces the search space by half.
- After k iterations, the search space reduces to $n / 2^k$.
- Solving for k gives $k = \log_2(n)$.
- Therefore, the time complexity is O(log n) because it takes logarithmic steps to reach the target element.

Source Code/Algorithm/Flow Chart:

```c
#include <stdio.h>
#include <stdlib.h>

int linearSearch(int a[], int len, int key){
    int count = 0;

    for(int i = 0; i < len; i++){
        if(a[i] == key)
            count++;
    }

    return count;
}

int binarySearch(int a[], int len, int key){
    int l = 0, mid;
    int h = len - 1;

    while(l <= h){
        mid = (l+h)/2;
        if(key == a[mid])
            return mid;
        else if(key > a[mid])
            l = mid + 1;
        else
            h = mid - 1;
    }

    return -1;
}

int recursiveBinarySearch(int a[], int l, int h, int key){
    int mid;

    if(l <= h){
        mid = (l+h)/2;
        if(key == a[mid])
            return mid;
        else if(key < a[mid])
            return recursiveBinarySearch(a, l, mid - 1, key);
        else
            return recursiveBinarySearch(a, mid + 1, h, key);
    }

    return -1;
}
```

```c
int main(){
    int *a;
    int n, key;

    printf("\nEnter size of array: ");
    scanf("%d", &n);
    a = (int *)malloc(n * sizeof(int));

    printf("\nEnter elements of the array:\n");
    for(int i = 0; i < n; i++)
        scanf("%d", &a[i]);

    printf("\nEnter key to search: ");
    scanf("%d", &key);

    linearSearch(a, n, key) ? printf("\nElement found with %d occurances (LS).\n", linearSearch(a, n, key)) : printf("\nElement not found (LS).\n");
    (binarySearch(a, n, key) == -1) ? printf("\nElement not found (BS).\n") : printf("\nElement found at index %d (BS).\n", binarySearch(a, n, key));
    (recursiveBinarySearch(a, 0, n - 1, key) == -1) ? printf("\nElement not found (RBS).\n") : printf("\nElement found at index %d (RBS).\n", recursiveBinarySearch(a, 0, n - 1, key));

    return 0;
}
```

Output Screenshots

```
Enter size of array: 5

Enter elements of the array:
1 2 3 4 5

Enter key to search: 3

Element found with 1 occurances (LS).

Element found at index 2 (BS).

Element found at index 2 (RBS).
fahee@Faheems-MacBook-Pro Data Structures %
```

Practice questions

1. What is Fibonacci Search explain in detail
2. Write algorithm for Fibonacci search
3. Implement Fibonacci Search
4. o/p screenshot

Conclusion

Thus we have studied different sorting algorithms and their time complexities.