Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)

Lab Report NO 02

Course Title: Data Structures Lab

Course Code: CSE 206 Section: 232_D10

Student Details

Name		ID
1.	Md.Rabby Khan	213902037

Lab Date : 11-09-2024

Submission Date : 17-09-2024

Course Teacher's Name : Fatema Akter

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

1. TITLE OF THE LAB REPORT EXPERIMENT

Implementation of Linear Search and Binary Search algorithms to find an element in an array of characters

2. OBJECTIVES/AIM

- Implement and compare the linear search and binary search algorithms.
- Understand the differences in performance between these two search methods.
- Analyze and test the algorithms using character arrays.

3. PROCEDURE / ANALYSIS / DESIGN

Algorithm for Linear Search:

1. Start
2. Input the size and elements of the array.
3. Input the search item.
4. Traverse the array using a loop and compare each element with the search item.
5. If found, display the index of the item.
6. If not found after the loop ends, display "Item not found."
7. End.

Algorithm for Binary Search:

1. Start
2. Input the size and elements of the array.
3. Sort the array.
4. Set beg = 0 and end = size - 1.
5. Calculate mid = (beg + end) / 2.
6. If arr[mid] == s_item, display the index.
7. If arr[mid] < s_item, set beg = mid + 1.
8. If arr[mid] > s_item, set end = mid - 1.
9. Repeat steps 5-8 until the item is found or the search range is exhausted.
10. If not found, display "Item not found."
11. End.

4. IMPLEMENTATION

4.1 linear Search

```
// linear Search
#include <iostream>
using namespace std;

int main()
{
   int i, size;
   char s_item;

   cout << "Enter the array size: ";
   cin >> size; // Input array size

   char arr[size]; // Declare array for characters

   cout << "Enter the array values (characters): ";
   for (i = 0; i < size; i++)
      cin >> arr[i]; // Input array elements

   cout << "Array values: ";
   for (i = 0; i < size; i++)
      cout << arr[i] << " "; // Display original array
   cout << endl;

   cout << "\nEnter the search item (character): ";
   cin >> s_item;

   // Perform linear search
   for (i = 0; i < size; i++)
   {
      if (arr[i] == s_item)
      {
         cout << i << " is the location of item '" << s_item << "'" << endl;
         break;
      }
```

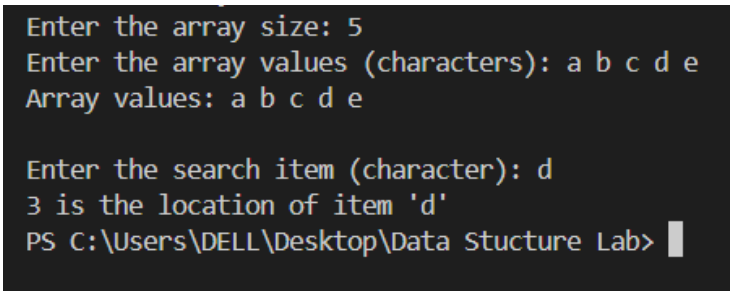
```
    }

    if (i == size)
        cout << "Item not found" << endl;

    return 0;
}
```

5. TEST RESULT / OUTPUT

5.1 Linear Search



4.2 Binary Search

```
//Binary Search
#include <iostream>
#include <algorithm> // For sorting the array
using namespace std;

int main()
{
    int i, size;
    char s_item;

    cout << "Enter the array size: ";
    cin >> size; // Input array size

    char arr[size]; // Declare array for characters

    cout << "Enter the array values (characters): ";
    for (i = 0; i < size; i++)
```

```
        cin >> arr[i]; // Input array elements

    // Sort the array for binary search
    sort(arr, arr + size);

    cout << "Sorted array values: ";
    for (i = 0; i < size; i++)
        cout << arr[i] << " "; // Display sorted array
    cout << endl;

    cout << "\nEnter the search item (character): ";
    cin >> s_item;

    int beg = 0, end = size - 1, mid;

    // Perform binary search
    while (beg <= end)
    {
        mid = (beg + end) / 2;

        if (arr[mid] == s_item)
        {
            cout << mid << " is the location of item '" << s_item << "'" << endl;
            return 0;
        }
        else if (arr[mid] < s_item)
            beg = mid + 1;
        else
            end = mid - 1;
    }

    cout << "Item not found" << endl;

    return 0;
}
```

5. TEST RESULT / OUTPUT

5.2 Binary Search

```
Enter the array size: 5
Enter the array values (characters): a c b d e
Sorted array values: a b c d e

Enter the search item (character): d
3 is the location of item 'd'
```

```
Enter the array size: 5
Enter the array values (characters): c b a d e
Sorted array values: a b c d e

Enter the search item (character): f
Item not found
PS C:\Users\DELL\Desktop\Data Stucture Lab>
```

6. ANALYSIS AND DISCUSSION

The linear search algorithm is straightforward and effective for small or unsorted arrays, but it becomes less efficient as the array size increases due to its O(n) time complexity. Each element must be checked one by one, which can be time-consuming for large datasets. On the other hand, binary search offers superior performance with its O(log n) time complexity, making it highly efficient for large arrays that are already sorted. However, binary search requires that the array be sorted first, which introduces an additional O(n log n) complexity for sorting. This sorting step can be a significant overhead if the array is not already sorted. The main challenges included ensuring that the array was correctly sorted before performing binary search and managing array bounds effectively to prevent errors. Overall, binary search is advantageous for large datasets where sorting is feasible and performance is critical, while linear search is more practical for smaller or unsorted arrays where simplicity and ease of implementation are valued.

7. SUMMARY

This experiment compared linear and binary search algorithms, showing that linear search is effective for small or unsorted arrays but inefficient for large datasets. Binary search, while faster for large sorted arrays with its O(log n) complexity, requires an additional sorting step. The experiment underscored the importance of choosing the right algorithm based on the dataset's characteristics.