A picture containing text

Description automatically generated



Lab Journal: 16

Date: June 5, 2025

Student: Hasan Zahid

Enrollment: 01-131232-028

Data Structures & Algorithms Lab (Spring-2025)

Teacher: Engr. Raheela Ambrin

Department of Software Engineering

Bahria University, Islamabad

Repository Link:

<https://github.com/hasanzafzal/DSA-Lab-16>

**Comments:**

**Signature**

# Lab No: 16 – Implementation of Searching Techniques

## Introduction

## To compare the efficiency of different search techniques we have studied in the course.

## Tools Used

Microsoft Visual Studio 2022

## Task 1:

Implement the following searching techniques and count number of **comparisons** for successful and unsuccessful search.

1. **Sequential Search**
2. **Binary Search**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **Sequential Search** | | **Binary Search** | |
|  | **Successful** | **Unsuccessful** | **Successful** | **Unsuccessful** |
| **10** |  |  |  |  |
| **100** |  |  |  |  |
| **1000** |  |  |  |  |
| **5000** |  |  |  |  |
| **10000** |  |  |  |  |
| **20000** |  |  |  |  |
| **50000** |  |  |  |  |

**Problem Statement**:  
An e-commerce company maintains two types of product datasets:

* Dataset A: Unsorted and small (e.g., daily product updates).
* Dataset B: Sorted and large (e.g., all-time bestsellers).

**Comparison**:

* + Implement sequential search for Dataset A and binary search for Dataset B.
  + Compare the performance of both algorithms on the respective datasets.

### Code

* Search.h:  
  #pragma once

#ifndef SEARCH\_H

#define SEARCH\_H

int SequentialSearch(int arr[], int size, int key, int& comparisons);

int BinarySearch(int arr[], int size, int key, int& comparisons);

#endif

* Search.cpp:  
  #include "Search.h"

int SequentialSearch(int arr[], int size, int key, int& comparisons)

{

comparisons = 0;

for (int i = 0; i < size; i++)

{

comparisons++;

if (arr[i] == key)

{

return i;

}

}

return -1;

}

int BinarySearch(int arr[], int size, int key, int& comparisons)

{

int low = 0, high = size - 1;

comparisons = 0;

while (low <= high) {

comparisons++;

int mid = (low + high) / 2;

if (arr[mid] == key)

return mid;

else if (arr[mid] < key)

low = mid + 1;

else

high = mid - 1;

}

return -1;

}

* Main.cpp:  
  #include <iostream>

#include <cstdlib>

#include <ctime>

#include "Search.h"

using namespace std;

int main()

{

srand(time(0));

int sizes[] = { 10, 100, 1000, 5000, 10000, 20000, 50000 };

cout << "N\tSequential Successful\tSequential Unsuccessful\tBinary Successful\tBinary Unsuccessful\n";

for (int i = 0; i < 7; i++)

{

int n = sizes[i];

int\* arr = new int[n];

for (int j = 0; j < n; j++)

{

arr[j] = j + 1;

}

int comparisons;

cout << n << "\t";

if (n <= 100)

{

int key = arr[rand() % n];

SequentialSearch(arr, n, key, comparisons);

cout << comparisons << "\t\t\t";

key = -1;

SequentialSearch(arr, n, key, comparisons);

cout << comparisons << "\t\t\t";

cout << "-\t\t\t-\n";

}

else

{

cout << "-\t\t\t-\t\t\t";

int key = arr[rand() % n];

BinarySearch(arr, n, key, comparisons);

cout << comparisons << "\t\t\t";

key = -1;

BinarySearch(arr, n, key, comparisons);

cout << comparisons << endl;

}

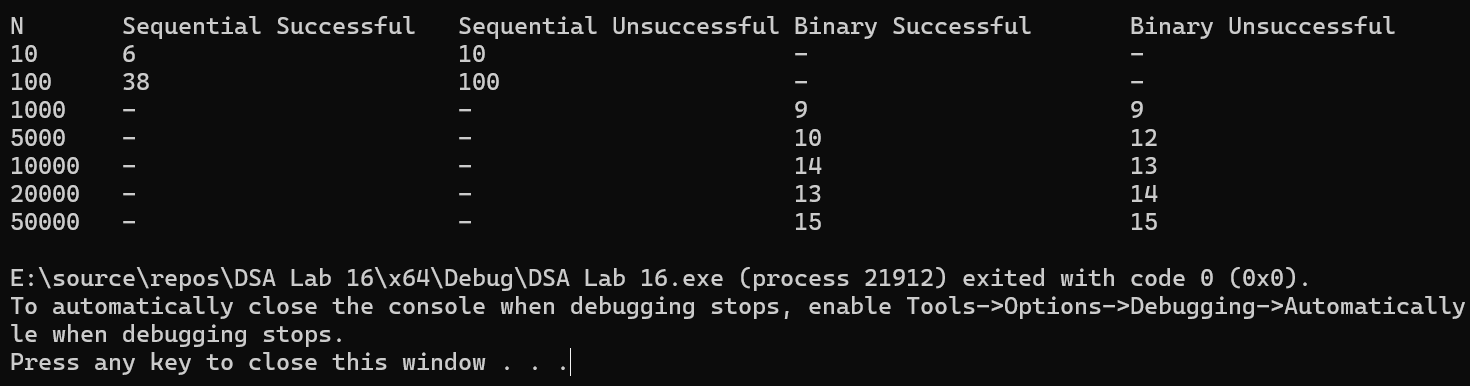
delete[] arr;

}

return 0;

}

### Screenshot



## Conclusion

In this lab searching techniques were implemented in c++ programming.