

**THE SUPERIOR UNIVERSITY LAHORE**

**Lab 2**

**(Fall2023)**

**Faculty of Computer Science and Information Technology Section: BSIT-3A**

**Subject: Data Structures & Algorithms-Lab Total Marks: -**

**Instructor: Sr. Shehryar Aziz**

Instructor: Muhammad Hassan Total Marks: 50

**dd**

**Introduction:**

In this lab, you will gain a comprehensive understanding of array sorting and searching algorithms in C++. You will learn to implement Linear Search, Binary Search, Insertion Sort, and Selection Sort. Additionally, you will independently write code for the Bubble Sort algorithm. This lab aims to enhance your programming skills and improve your ability to choose the appropriate algorithm for different scenarios.

1. **Linear Search:**

**Description:**

Linear search is a fundamental searching algorithm that scans an array sequentially to find a target element. It compares each element with the target until a match is found or the entire array has been searched.

**Algorithm:**

* Start from the first element of the array.
* Compare the current element with the target.
* If they match, return the index of the current element.
* If not, move to the next element.
* Repeat steps 2-4 until the target is found or the end of the array is reached.
* If the target is not found after iterating through the entire array, return -1.

int linearSearch(int arr[], int size, int key) {

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

if (arr[i] == key) {

return i; // Return the index of the element if found

}

}

return -1; // Return -1 if the element is not found

}

1. **Binary Search:**

**Description**:

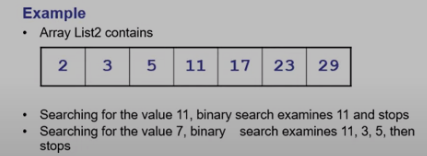
Binary search is Quicker Method of searching for value in the Array .

Binary search is very Quick but it can only Search a sorted Array.

It can not be applied on unsorted array .

**Steps to search A value in array :**

* It location the midel elements of the array and compares with the searching number.
* If the Are equal ,the Search is successful and the index of searching element is returned.
* If the array is not equal ,it reduce to search half of the array .
* If the searching number less then the midle elements
  + It search the first element half of the array .
  + Otherwise it search last element half of the array.
* The process continue until the require number is found or loop complete without successful



int binarySearch(int arr[], int size, int key) {

int left = 0;

int right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == key) {

return mid; // Return the index of the element if found

}

if (arr[mid] < key) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1; // Return -1 if the element is not found

}

Or

#include<iostream>

using namespace std;

int main(){

int A[10]={10,20,30,40,50,60,70,80,90,100};

int n, mid ,start,end,loc;

loc=-1;

start=0;

end =9;

cout<<"Enter the any number to find :";

cin>>n;

while(start<=end){

mid=(start+end)/2;

if(A[mid]==n){

loc=mid;

break;

}

else if(n<A[mid]){

end=mid-1;}

else

start=mid+1;

if(loc==-1){

cout<<n<<"not found!="<<endl;

}

else

cout<<n<<"found at index"<<loc<<endl;

return 0;

}

}

1. **Insertion Sort:**

**Description:**

Insertion sort is a straightforward sorting algorithm that builds the final sorted array one element at a time. It is suitable for small data sets and works efficiently with nearly sorted arrays.

**Algorithm:**

* Start with the second element (index 1) and consider it as the current element.
* Compare the current element with the elements to its left.
* Move each element to the right until you find the correct position for the current element.
* Insert the current element into its correct position in the sorted sub array.
* Repeat steps 1-4 for the remaining elements in the array.

void insertionSort(int arr[], int size) {

for (int i = 1; i < size; i++) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = key;

}

}

1. **Selection Sort:**

**Description:**

Selection sort is a simple sorting algorithm that repeatedly selects the minimum element from the unsorted portion of the array and places it at the beginning.

**Algorithm:**

* Find the minimum element in the unsorted part of the array.
* Swap the minimum element with the first element in the unsorted part.
* Mark the first element as sorted.
* Repeat steps 1-3 for the remaining unsorted portion of the array.

void selectionSort(int arr[], int size) {

for (int i = 0; i < size - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < size; j++) {

if (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

if (minIndex != i) {

std::swap(arr[i], arr[minIndex]);

}

}

}

1. **Bubble Sort:**

**Description:**

Bubble sort is a straightforward sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process continues until no more swaps are required.

Code for Bubble Sort will not be provided here. You will write the code for Bubble Sort as part of the lab task.

**Lab Tasks**

1. Write a C++ program that performs a linear search on an array of integers. The program should ask the user to enter a target value and then use the linear search algorithm to find and display the index of the target value in the array.
2. Write a C++ program that performs a binary search on a sorted array of integers. The program should ask the user to enter a target value and then use the binary search algorithm to find and display the index of the target value in the array. Ensure that the array is sorted before performing the binary search.
3. Write a C++ program that sorts an array of integers using the insertion sort algorithm. The program should ask the user to enter the number of elements in the array, input the elements, and then use the insertion sort algorithm to sort and display the sorted array.
4. Write a C++ program that sorts an array of integers using the selection sort algorithm. The program should ask the user to enter the number of elements in the array, input the elements, and then use the selection sort algorithm to sort and display the sorted array.
5. Write a C++ program that implements the Bubble Sort algorithm to sort an array of integers. The program should ask the user to enter the number of elements in the array, input the elements, and then use the Bubble Sort algorithm to sort and display the sorted array. Remember, you will need to write the code for Bubble Sort from scratch.