**Subject: Data Structures Class/ Sem: S. Y B.Tech/ Sem-IV A.Y: 2024-25 (Even)**

**EXPERIMENT NO. 6**

**SEARCHING AND SORTING**

**AIM**: Implementation of Linear Search, Binary Search and Bubble Sort.

**THEORY:**

**Linear Search**

Linear Search is the process of finding some particular element in the list. If the element is present in the list, then the process is called successful, and the process returns the location of that element; otherwise, the search is called unsuccessful. Linear search is also called as sequential search algorithm. In Linear search, we simply traverse the list completely and match each element of the list with the item whose location is to be found. If the match is found, then the location of the item is returned; otherwise, the algorithm returns NULL. It is widely used to search an element from the unordered list. The worst- case time complexity of linear search is O(n).

**Binary Search**

A binary search algorithm is a technique for finding a particular value in a linear array, by ruling out half of the data at each step, widely but not exclusively used in computer science. A binary search finds the median, makes a comparison to determine whether the desired value comes before or after it, and then searches the remaining half in the same manner. Another explanation would be: Search a sorted array by repeatedly dividing the search interval in half Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half Otherwise, narrow it to the upper half.

**Bubble Sort**

Bubble sort is a sorting algorithm that compares two adjacent elements and swaps them until they are in the intended order. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of O(n2) where n is the number of items.

**ALGORITHM:**

**Linear Search**

// 'a' is the given array, 'n' is the size of given array, 'val' is the value to search

1. set pos = -1
2. set i = 1
3. repeat step 4 while i <= n
4. if a[i] == val

set pos = i print pos go to step 6

set ii = i + 1

1. if pos = -1

print "value is not present in the array "

1. exit

**Binary Search**

1. Sort an array a[j] in increasing order.
2. Read ‘n’ elements in the array , say ‘a ’.
3. Read the element to be searched, say ‘KEY ‘.
4. low = 0 , high = n-1
5. while(high>=low) perform following steps. Find the middle element.
6. mid = (low + high )/ 2
7. if ( KEY= a[mid] )

print (‘ element found at mid position)

1. if KEY < a[mid] then search for target in a[low] to a[mid-l] else search for target in a[mid+l] to a[high].
2. if (KEY < a[Mid\_position]) high = mid\_position - 1;

else

low = mid\_position + 1;

1. Stop.

**Bubble Sort**

// A is an array

for i ← 1 to length [A] do

            for j ← length [A] down-to i +1 do if A[j] < A[j-1] then

          Exchange A[j] ⟷ A[j-1]

**CODE**:

Linear Search

#include <stdio.h>

int linearSearch(int arr[], int n, int x) {

  for (int i = 0; i < n; i++) {

    if (arr[i] == x) {

      return i;

    }

  }

  return -1;

}

int main() {

  int arr[] = {2, 5, 1, 9, 4, 7};

  int n = sizeof(arr) / sizeof(arr[0]);

  int x = 4;

  int index = linearSearch(arr, n, x);

  if (index == -1) {

    printf("Element is not found!\n");

  } else {

    printf("Element is found at index: %d\n", index);

  }

  return 0;

}

Binary Search

// Binary Search in C

#include <stdio.h>

int binarySearch(int array[], int x, int low, int high) {

  // Repeat until the pointers low and high meet each other

  while (low <= high) {

    int mid = low + (high - low) / 2;

    if (x == array[mid])

      return mid;

    if (x > array[mid])

      low = mid + 1;

    else

      high = mid - 1;

  }

  return -1;

}

int main() {

  int array[] = {3, 4, 5, 6, 7, 8, 9};

  int n = sizeof(array) / sizeof(array[0]);

  int x = 4;

  int result = binarySearch(array, x, 0, n - 1);

  if (result == -1)

    printf("Not found");

  else

    printf("Element is found at index %d", result);

  return 0;

}

Bubble Sort

// Bubble sort in C

#include <stdio.h>

// perform the bubble sort

void bubbleSort(int array[], int size) {

  // loop to access each array element

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

    // loop to compare array elements

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

      // compare two adjacent elements

      // change > to < to sort in descending order

      if (array[i] > array[i + 1]) {

        // swapping occurs if elements

        // are not in the intended order

        int temp = array[i];

        array[i] = array[i + 1];

        array[i + 1] = temp;

      }

    }

  }

}

// print array

void printArray(int array[], int size) {

  for (int i = 0; i < size; ++i) {

    printf("%d  ", array[i]);

  }

  printf("\n");

}

int main() {

  int data[] = {-2, 45, 0, 11, -9};

  // find the array's length

  int size = sizeof(data) / sizeof(data[0]);

  bubbleSort(data, size);

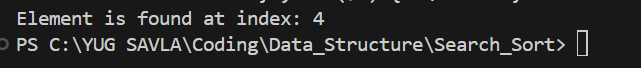
  printf("Sorted Array in Ascending Order:\n");

  printArray(data, size);

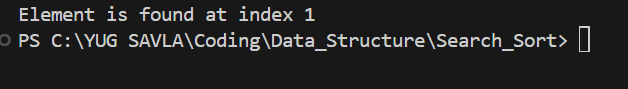
}

**OUTPUT:**

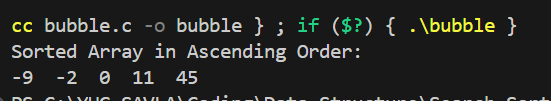
Linear Search:



 Binary Search



Bubble Sort



**ANALYSIS OF TIME COMPLEXITY:**

**Linear Search:** O(n) **Binary Search:** O( log n ) **Bubble Sorting:** O(n2)

**CONCLUSION**: Thus we have successfully implemented linear and binary searching techniques and bubble sorting.