Lab Cycle: 8 **CSL 201 Data structures Lab**

Date of Submission: 25-2-2022

Faculty In charge: Dr Binu V P

Objective: Learn Searching and Sorting Techniques

Objective: Learn Different Searching and Sorting Techniques.

Do the Time and Space complexity analysis of the various algorithms and compare them.

---

1. Implement Linear Search

Search an array or list by checking items one at a time.

Time complexity ……O(n)

---

2. Implement Binary Search.

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. Repeatedly check until the value is found or the interval is empty.

Time complexity ……O(logn)

---

3. Implement Bubble Sort

Sort by comparing each adjacent pair of items in a list in turn, swapping the items if necessary, and repeating the pass through the list until no swaps are done.

Time complexity: $O(n^2)$

---

4. Implement Exchange Sort

Compare the first element with remaining elements in the list and when ever they are out of order, order them .Repeat the procedure for n-1 elements in the list

Time complexity: $O(n^2)$

---

5. Implement Selection Sort

Find the smallest element in the list and place it in the first position. Repeatedly looks through remaining items to find the least one and moves it to its final location.

Time complexity: $O(n^2)$

---

6. Implement Insertion Sort

Sort by repeatedly taking the next item and inserting it into the final data set in its proper order with respect to items already inserted (sorted).

Time complexity: $O(n^2)$

---

7. Merge two sorted Arrays.

Given two sorted arrays A:n and B:m. the program should create a sorted array C:m+n by merging them.

8.Implement Merge sort

It is a divide and conquers algorithm. Divide the list recursively and merge the sorted list in each step to obtain the final sorted list.

The time complexity is O(n.log(n))

---

9. Implement Quick sort

It is also called partition exchange sort. Partition the list recursively and sort the sub lists to obtain the final sorted list.

The time complexity is O(n.log(n))

---

10. Implement Heap sort

The heap sort works as its name suggests - it begins by building a heap out of the data set, and then removing the largest item and placing it at the end of the sorted array. After removing the largest item, it reconstructs the heap and removes the largest remaining item and places it in the next open position from the end of the sorted array. This is repeated until all elements are sorted.

The time complexity is (n log(n))

11.Find the second largest element from a list. There may be more than one largest element.

Hint: Sort the list in the descending order and find the element which is not the largest

---

12. Find the duplicate element from a list. Also list the number of occurrence of each element.

Hint: Sort the list and count the occurrence of each element

---

13. Find the mean, median, mode, and range of the list of values:

(if the list is not sorted sort the list first)

Example

8, 9, 10, 10, 10, 11, 11, 11, 12, 13

The mean is the usual average

(8 + 9 + 10 + 10 + 10 + 11 + 11 + 11 + 12 + 13) ÷ 10 = 105 ÷ 10 = 10.5

The median is the middle value. In a list of ten values, that will be the (10 + 1) ÷ 2 = 5.5th value; that is, I'll need to average the fifth and sixth numbers to find the median:

(10 + 11) ÷ 2 = 21 ÷ 2 = 10.5

The mode is the number repeated most often. This list has two values that are repeated three times.10 and 11

The largest value is 13 and the smallest is 8, so the range is 13 – 8 = 5.

mean: 10.5
median: 10.5
modes: 10 and 11
range: 5