**DEPARTMENT OF INFORMATION TECHNOLOGY**

**Course Name and Code:** Data Structures Lab **(**ITL302)

**Semester:** III (Second Year)

**Academic Year:** 2023-24 (Odd Semester)

**Experiment No. 08**

**Aim:** Implementation of any one Sorting Technique considering a real-world application.

**Objectives:**

1. To impart knowledge of sorting and searching algorithms.

**Theory:**

1. **Introduction to sorting:**

Sorting refers to rearrangement of a given array or list of elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of elements in the respective data structure. Sorting means reordering of all the elements either in ascending or in descending order.

**Characteristics of Sorting Algorithms:**

**Time Complexity:** Time complexity, a measure of how long it takes to run an algorithm, is used to categorize sorting algorithms. The worst-case, average-case, and best-case performance of a sorting algorithm can be used to quantify the time complexity of the process.

**Auxiliary Space:** This is the amount of extra space (apart from input array) needed to sort. For example, Merge Sort requires O(n) and Insertion Sort O(1) auxiliary space

**Stability:** A sorting algorithm is said to be stable if the relative order of equal elements is preserved after sorting. This is important in certain applications where the original order of equal elements must be maintained.

1. **Types of sorting:**

Some types of sorting are:

* Selection sort
* Bubble sort
* Insertion sort
* Merge sort
* Quick sort
* Heap sort
* Counting sort
* Radix sort
* Bucket sort

1. **Introduction to selection sort:**

Selection Sort is a comparison-based sorting algorithm. It sorts an array by repeatedly selecting the smallest (or largest) element from the unsorted portion and swapping it with the first unsorted element. This process continues until the entire array is sorted.

1. **Algorithm:**

**Accepts an array arr of size n.**

Step 1:Set i = 0

Step 2: while i < n - 1 repeat steps 3 & 4

Step 3: Set j = i + 1

Step 4: while j < n repeat steps 5 & 6

Step 5: If arr[i] > arr[j]

then Swap(arr[i], arr[j])

Step 6: j++

Step 7: i++

Step 8: Stop

1. **Example:**

Assume an array of 4 elements [5, 1, 8, 0] on applying selection sort the following will be the result:

Pass 1: a) [1, 5, 8, 0]

b) [1, 5, 8, 0]

c) [0, 5, 8, 1]

Pass 2: a) [0, 5, 8, 1]

b) [0, 1, 8, 5]

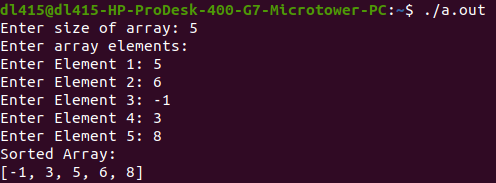
Pass 3: a) [0, 1, 5, 8]

Hence the array is sorted.

**Program:**



**Output:**



**Conclusion:**

In this experiment we were able to learn different types of sorting, their algorithms and time complexities. We also learned how to implement these algorithms in C programming language.

**Outcome:**

Implement sorting and searching techniques for real-world applications.

**Submitted By:**

**Name:** Faizan Dodiya

**Div:** SY/A

**Roll No:** 24