Assignment No.11

Problem Statement:

Sorting: Write a C++ menu driven program to store the percentage of marks obtained by the students in an array. Write function for sorting array of floating point numbers in ascending order using

1. Selection Sort
2. Bubble sort
3. Insertion sort
4. Shell Sort
5. Quick sort
6. Radix sort
7. Display top five scores

Implement 4 methods of sorting. Provide choice to user to take input from user or using random numbers.

Use Standard Template Library (STL) sort function for above data.

Objective:

1. To understand sorting techniques
2. To understand use of different sorting algorithms for different use cases

Theory:

Selection sort:

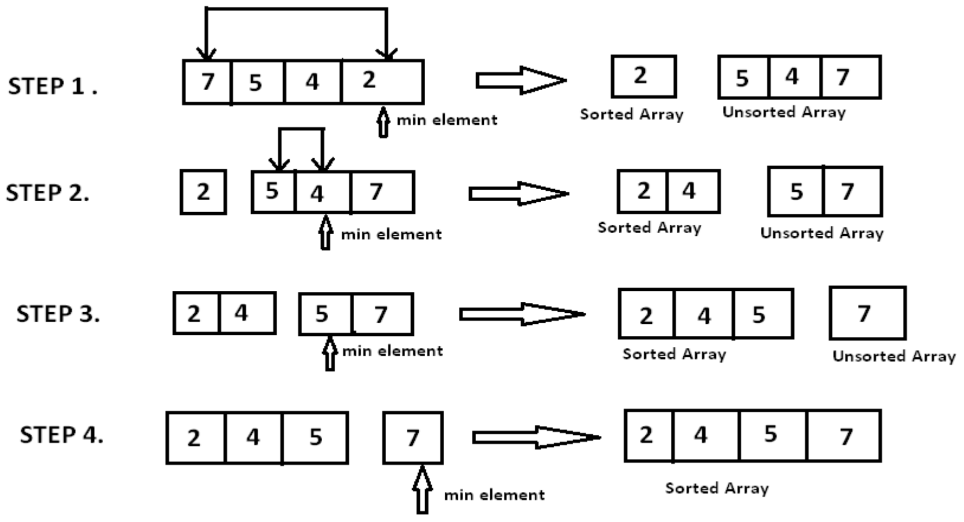
Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

The Selection sort algorithm is based on the idea of finding the minimum or maximum element in an unsorted array and then putting it in its correct position in a sorted array.

Assume that the array A=[7,5,4,2]A=[7,5,4,2] needs to be sorted in ascending order. The minimum element in the array i.e. 22 is searched for and then swapped with the element that is currently located at the first position, i.e. 77. Now the minimum element in the remaining unsorted array is searched for and put in the second position, and so on.

Graphical representation of selection sort:



Algorithm:

Input: Array of students marks
Output: Sorted array of students marks

Step 1: Start

Step 2: Set MIN to location 0

Step 3: Search the minimum marks in the list

Step 4: Swap with value at location MIN

Step 5: Increment MIN to point to next marks

Step 6: Repeat until list is sorted

Step 7: Stop

Bubble Sort:

Bubble sort is based on the idea of repeatedly comparing pairs of adjacent elements and then swapping their positions if they exist in the wrong order.

Example:

First Pass:
(**5** **1** 4 2 8) –> (**1** **5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.
(1 **5** **4** 2 8) –> (1 **4** **5** 2 8), Swap since 5 > 4
(1 4 **5** **2** 8) –> (1 4 **2** **5** 8), Swap since 5 > 2
(1 4 2 **5** **8**) –> (1 4 2 **5** **8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.
Second Pass:
(**1** **4** 2 5 8) –> (**1** **4** 2 5 8)
(1 **4** **2** 5 8) –> (1 **2** **4** 5 8), Swap since 4 > 2

(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)
Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.
Third Pass:
(**1 2** 4 5 8) –> (**1 2** 4 5 8)
(1 **2 4** 5 8) –> (1 **2 4** 5 8)
(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)

Algotithm:

Input: Array of students marks
Output: Sorted array of students marks

Step1-Start

Step2- for all elements i=0 to N

Step3- if marks of i th student is greater than marks of i+1 th student then swap both marks

Step4- increment I by 1

Step5- Continue loop till N students

Step6-Stop

Time Complexity:
Compare different all sorting algorithms with respect to best case , worst case of time and space complexity

Practice problem:

1. Write algorithm of the following
   - Insertion sort
   - Shell Sort
   - Quick sort
   - Radix sort

2. Consider the array {56,12,90,78,34,78,52,2,89,60,32} and step by step solve it using following sorting techniques
   1. Selection Sort
   2. Bubble sort
   3. Insertion sort
   4. Shell Sort
   5. Quick sort
   6. Radix sort