Experiment No : 5

Aim: Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using
a) Selection Sort b) Bubble sort and display top five scores.

Software Requirements: 64-bit Open source Linux
Python IDE like spyder

Hardware Requirement: C2D, 2GB RAM, 500 GB HDD.

Objectives: To understand sorting algorithms using python programming

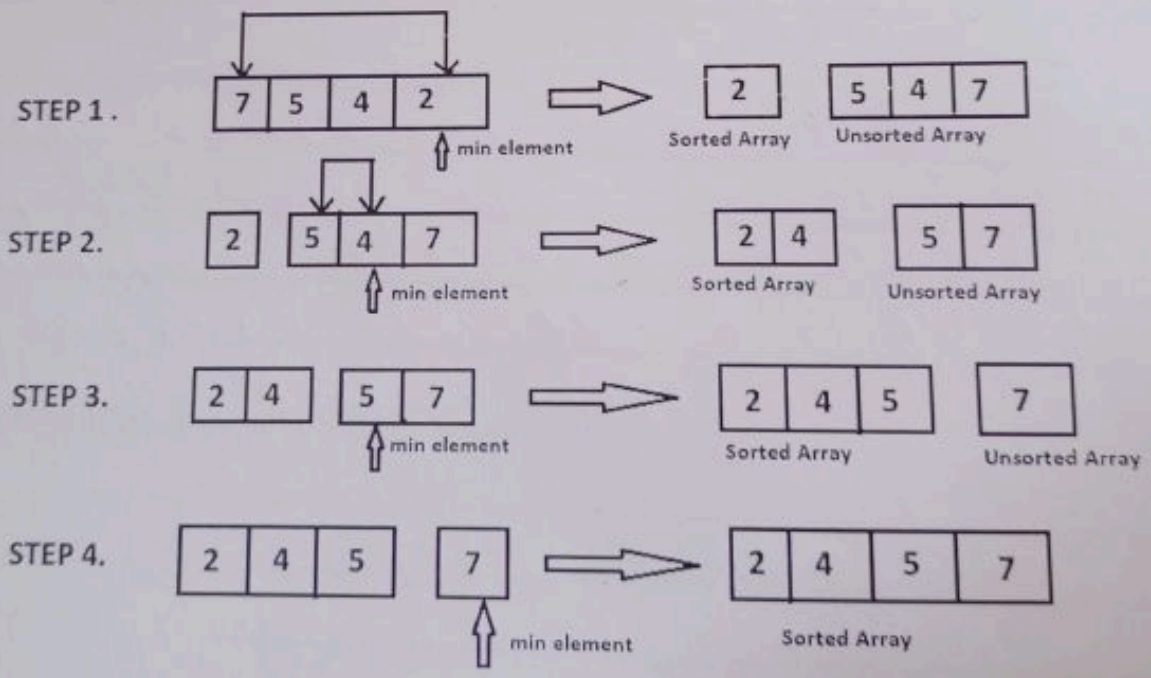
Outcomes: Student will able to select appropriate sorting algorithm as per requirements

Theory:

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 list A= [7,5,4,2] needs to be sorted in ascending order.

The minimum element in the array i.e. 2 is searched for and then swapped with the element that is currently located at the first position, i.e. 7. Now the minimum element in the remaining unsorted array is searched for and put in the second position, and so on.



Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in 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)

Algorithm:

Selection sort

1. Set MIN to location 0
2. Search the minimum element in the list
3. Swap with value at location MIN
4. Increment MIN to point to next element
5. Repeat until list is sorted

Bubble sort

1. start from i=0
2. flag=0, j=i+1
3. Compare A[i] with A[j]
4. Swap A[i]with A[j] if ith element is greater than jth element and make flag=1
5. Increment j to compare next element with A[i] and if j<n goto step 3 again else go to step 6
6. if flag=0 goto 7 else increment i and goto step 2
7. stop

Conclusion: Thus implemented program to sort elements of list using selection and bubb sort algorithm

AVCOE SANGAMNER

Questions:

1. What is time complexity of bubble sort?

$O(n^2)$

2. What is time complexity of selection sort?

$O(n^2)$

3. Which sorting algorithm among above is useful in best case?

Bubble sort

4. How bubble sort is optimized to get quick result?

The code can be optimized by introducing an extra space swapped after each iteration if there is no swapping taking place then there is no need for performing further loops.

Staff Signature & Date