Week 4: Sorting Algorithms – Selection and Insertion Sort

Last week, we implemented Bubble Sort and Cocktail Shaker Sort.
This week you will continue writing the Java code for the other two coding algorithms studied so far: Selection Sort and Insertion Sort.

Before starting the tasks below, make sure you remember the difference between the four sorting algorithms studied so far. Make use of you notes, lectures' slides and Google if you need. You can also find the pseudo-code of Selection Sort and Insertion sort at the end of this document. You can use them as a guide to write the Java code, but make sure you understand what is happening at each stage.

We have written below the sorting strategy of each algorithm.

- Bubble sort – The heaviest "bubble" goes at the end the array at each step.
- Cocktail-Shaker Sort - The heaviest "bubble" goes at the end the array and the lightest "bubble" goes at the front of the array, at each step.
- Selection Sort – Select the minimum element and swap it with the item at index 0. Select the next smallest element and swap it with the item at index 1. Repeat until the array is sorted
- Insertion Sort – If the first two items are not sorted, swap them. Compare the third item with all the items on its left (at index 0 and 1) and insert it in its correct position (either at index 0, 1 or 2). Mark the fourth item, compare it with all the items on its left and insert it in its correct position. Repeat for all the elements of the array.

Part A – Selection Sort and Insertion Sort

Use the source code provided last week to implement two more sorting algorithms (you can find it on Blackboard under Week 3\Lab Session\SortingTest.java). Attempt the following exercises.

1) Implement the SelectionSort algorithm. Check that the results are as expected (i.e. the integer values are sorted in ascending or descending numerical order). Have a look at this video and the pseudo-code if you need.

2) Implement the InsertionSort algorithm. Check that the results are as expected (i.e. the integer values are sorted in ascending or descending numerical order). Have a look at this video and the pseudo-code if you need.

Part B – measuring task

You will now have four sorting methods at your disposal:
bubbleSort_opt2(), shakerSort(), selectionSort() and insertionSort()
Add a counter to the methods selectionSort() and insertionSort() that counts the number of **comparisons** made. Run the methods with arrays of different sizes, but the same you used last week (say, n = 10, 20, 100 and 500 random integers).

How do these four algorithms compare? On what size, if any, does the difference in the number of comparisons become significant? Build a table of results.

As indicated last week, a more comprehensive analysis would require you to keep two types of counter, one to keep track of the number of comparisons made, and one to keep track of the number of assignments made (remember, a shift operation or an insert operation is worth one assignment, but a swap operation takes three assignments). Obtain a feel for the relative cost of the three sorting algorithms seen so far by running them on the same arrays of data and comparing their counter values. Complete the table of results you started last week. Do you come to any form of conclusion? Is there a natural way to rank these algorithms?
Do you remember the time complexity of these algorithms?

Part C – investigative task

If you have finished with the above, investigate the Bucket Sort algorithm by doing some research on the net.
Attempt to implement it. Once you have finished, compare your solution to this.

Extension Task 1 - Interview Questions

Answer the exam questions below in your exercise book.

Question 1

Using the data provided in the following array [8, 3, 9, 6],

- Draw a step-by-step trace of the Selection Sort algorithm.
- Provide a description in English of the mode of operation of Selection Sort.
- Provide the pseudo-code for the Selection Sort algorithm.

Question 2

Using the data provided in the following array [8, 3, 9, 6, 5, 3],

- Draw a step-by-step trace of the Insertion Sort algorithm.
- Provide a description in English of the steps of Insertion Sort.
- Provide the pseudo-code for the Insertion Sort algorithm.

Question 3

Using the data provided in the following array [8, 3, 9, 6],

- Draw a step-by-step trace of the Bubble Sort algorithm.
- Provide the pseudo-code for the Bubble Sort algorithm.
- Write the time complexity formula for Bubble Sort.
- Provide a detailed description in English of the mode of operation of Bubble Sort.
- Provide the name of another sorting algorithm quicker than Bubble Sort.

Extension Task 2 – Project Euler

Project Euler is a series of challenging mathematical/computer programming problems that will require more than just mathematical insights to solve. Although mathematics will help you arrive at elegant and efficient methods, the use of a computer and programming skills will be required to solve most problems.

Use the link below to access the website, start from problem 1 and keep practicing every week. You can sign-up to save your progress (it's free), or you can simply implement the problems without signing up.

https://projecteuler.net/archives

Appendix 1 – Pseudo-code Selection Sort

```
procedure selection sort
   list  : array of items
   n     : size of list

   for i = 0 to n - 2
   /* set current element as minimum*/
      min = i

      /* check the element to be minimum */

      for j = i+1 to n - 1
         if list[j] < list[min] then
            min = j;
         end if
      end for

      /* swap the minimum element with the current element*/
      if  min!= i  then
         swap list[min] and list[i]
      end if
   end for

end procedure
```

Appendix 2 – Pseudo-code Insertion Sort

- *a* is an array of size *N*

```
for i from 1 to N-1
   key = a[i]
   j = i - 1
   while j >= 0 and a[j] > key
      a[j+1] = a[j]
      j = j - 1
   a[j+1] = key
next i
```