CSI 213 - Data Structures

Lab 09 - Sorting - Insertion Sort

April 3, 2017

1 Insertion Sort Overview

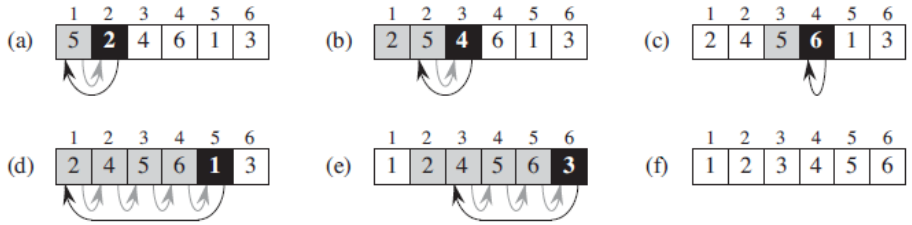
Insertion Sort is a simple sorting algorithm, that is easy to understand, although it's less efficient than the advanced sort algorithms, such as Merge Sort, Quicksort, etc.

Advantages of insertion sort are:

- Easy to implement (the most concise implementation being 3 lines of C code. You need to write 11 lines in Java, including the curly brackets).
- Time efficient for small datasets.
- Space efficient, $\mathcal{O}(1)$, it does not require extra memory space to sort the elements of the array.
- Stable (it doesn't change the relative order of elements with identical keys).

Insertion sort involves sorting the data, one element at a time. For each element, the algorithm inserts that element into the appropriate spot in the already-sorted portion of the list.

Let's look at the following figure from the "Introduction to Algorithms" book by the MIT Press:



Let's work our way through the figure:

You will not change anything on the first element, "5", nothing the compare yet, you're just starting to sort. Start from the second element in the array, in this example "2".

(a) Compare *2* with the previous elements, in this case, it's only *5*. Since $2 < 5$, then swap the elements *2* and *5*.

(b) Now we're at the element *4*. Check *4* with *2* and *5*. Since $4 < 5$, then swap elements *4* and *5*.

(c) Element *6* will stay in its place, because the other elements are less than *6*.

(d) Element *1* is smaller than every element in the list, so it will first change places with *6*, then *5*, then *4* and at last with *2*.

(e) Element *3* is smaller than *6*, *5*, and *4* so it will swap its' place with these elements.

(f) Finally the list is sorted.

2 Instructions

Implement the method "insertionSort" in the attached code. You will have an outer loop (hint: "for" loop) that starts from the second element in the list (1st element in the array, since the first index in arrays are 0 in java), till it reaches to the end of the list (length of the array).

Then, there will be inner loop that checks the element with the previous elements, to see if the new element is smaller than the already sorted list (previous elements). This loop(hint: "for" loop) will start from the current element in the outer loop, then till it reaches the beginning of the list, it will decrement.

In the inner loop, there will be an "if" that checks if the element *"j"* from the inner loop is smaller than *"j-1"*. If it is, then swap these elements.

3 Expected Output

Before Insertion Sort:
6, 30, 10, 54, 2, 62, 80, 42
After Insertion Sort:
2, 6, 10, 30, 42, 54, 62, 80

4 Grading

BEFORE YOU LEAVE, SHOW YOUR PROGRESS TO YOUR TA TO GET CREDIT

Grading for this lab is based on completion level:

5.0 = Successfully finished the assignment at the lab, on time and helped other students.

4.0 = Successfully finished the assignment at the lab, on time.

3.0 = Most of the assignment is finished with couple errors/missing parts

2.0 = Some of the assignment is finished and left the room early.

1.0 = Did not successfully finish, but attended the lab and showed effort.

0.0 = Did not attend the lab section, or did not attempt the lab.