Insertion Sort Algorithm
Homework #2

By
Eduardo Castro

CS 303 Algorithms and Data Structures
September 10, 2014

1. Problem Specification

The goal of this assignment was to write a program that will read a text file with a lot of different inputs and sort it using the Insertion Sort algorithm. The sorting should be tested by another class provided by the professors. All the inputs should be integer values.

2. Program Design

This program requires a InsertionSort class, a driver class(InsertionSortTest) to manage what will be sorted and a FileReadSortCheck class that is responsible for checking if the sorting was done right. The InsertionSort just receives an array and it's size and try to sort it according to the Insertion Sort algorithm. We also need to measure the time that the sorting took. The InsertionSortTest class reads a text file with all the inputs, put it into a new array and call/send it to the InsertionSort class who will make all the work alone.

The following steps were required to develop this program:

a) write the InsertionSort class to contain the algorithm for Insertion Sort
b) use the driver class to read in the input file and store it into an array of integers
c) use methods invoked from the InsertionSort class to sort all the elements
d) use methods invoked from the FileReadSortCheck class check if the sorting was done correctly
e) measure the time that the sorting took
f) the driver class displays if the sorting was made right or not and the time that it took.

The following constructors and methods were defined within the InsertionSort class:

a) sortAnArray ()

Function that receives an array and it's respective size and sort it according to the InsertionSort algorithm.

The following constructors and methods were defined within the InsertionSortTest class:

a) main ()

Main function of all the program. It reads an input txt file, gets all the integer elements and insert it into an array. Then it measures the size of this array and send both the array the its size to the sortAnArray method from InsertionSort class. To let the measuring easier I created a for loop to measure the time that the sorting takes 10 times in-a-row and print it.

b) getHowManyElementsOnTheInputsFile ()

Read the inputs text file, iterates throughout all its elements and count how many elements there are into it and return this number.

The Scanner class provided by Java was used to read in the necessary values within the provided driver program. The println method of the System.out object was used to display the inputs and results for the provided driver program.

3. Testing Plan

The driver program, InsertionSortTest, was used to call the sorting method and the FileReadSortCheck class to test the sorting. We used different input files with a big range of elements, starting with a file with around 128 elements and finishing with the biggest file that has around 100000 elements. All these inputs files were provided by the assignment.

4. Test Cases

The test cases are shown in the table below:

Test Case Number	File Number	Expected Output

1	1	`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 0`

2	2	`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 0`
		`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 1`
		`It's sorted. The elapsed time is: 0`

3	3	It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 2 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 0 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 0 It's sorted. The elapsed time is: 0 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 2 It's sorted. The elapsed time is: 0
4	4	It's sorted. The elapsed time is: 3 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 8 It's sorted. The elapsed time is: 0 It's sorted. The elapsed time is: 0 It's sorted. The elapsed time is: 0 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 0 It's sorted. The elapsed time is: 0 It's sorted. The elapsed time is: 1
5	5	It's sorted. The elapsed time is: 4 It's sorted. The elapsed time is: 4 It's sorted. The elapsed time is: 5 It's sorted. The elapsed time is: 18 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 1
6	6	It's sorted. The elapsed time is: 20 It's sorted. The elapsed time is: 21 It's sorted. The elapsed time is: 43 It's sorted. The elapsed time is: 2 It's sorted. The elapsed time is: 2 It's sorted. The elapsed time is: 2 It's sorted. The elapsed time is: 1 It's sorted. The elapsed time is: 2 It's sorted. The elapsed time is: 2 It's sorted. The elapsed time is: 2

7	7	It's sorted. The elapsed time is: 23 It's sorted. The elapsed time is: 32 It's sorted. The elapsed time is: 10 It's sorted. The elapsed time is: 7 It's sorted. The elapsed time is: 7 It's sorted. The elapsed time is: 7 It's sorted. The elapsed time is: 8 It's sorted. The elapsed time is: 8 It's sorted. The elapsed time is: 7 It's sorted. The elapsed time is: 7
8	8	It's sorted. The elapsed time is: 64 It's sorted. The elapsed time is: 57 It's sorted. The elapsed time is: 11 It's sorted. The elapsed time is: 11 It's sorted. The elapsed time is: 11 It's sorted. The elapsed time is: 11 It's sorted. The elapsed time is: 11 It's sorted. The elapsed time is: 12 It's sorted. The elapsed time is: 11 It's sorted. The elapsed time is: 11
9	9	It's sorted. The elapsed time is: 3611 It's sorted. The elapsed time is: 3631 It's sorted. The elapsed time is: 1104 It's sorted. The elapsed time is: 1088 It's sorted. The elapsed time is: 1081 It's sorted. The elapsed time is: 1087 It's sorted. The elapsed time is: 1093 It's sorted. The elapsed time is: 1125 It's sorted. The elapsed time is: 1115 It's sorted. The elapsed time is: 1108

5. Analysis and Conclusions

All the inputs file were corruptly sorted by the InsertionSort class and I had no much difficult doing it, everything was fine and executed as expected.
It's interesting that the program is really fast for the first 5 inputs, almost instantaneous. For the following we can notice that it is taking a bit longer for each one, especially for the last input file with 100000 elements.

I don't understand why the program returns so much different times between each sorting, per example the sorting for the input9.txt: the first two measures were around 3600ms and the other

eight measures were around 1100ms. Probably this is a signal that I need to fix something in the code but the sorting is working fine, only not optimised.

6. References

The sample report was provided to the class by the professors and the parameters was provided in the Canvas.