1 Assignment 1

Submit source code and running instructions to EAS[1]. Submit the textual components in a single text/pdf file, providing the assignment numbering to help your marker. Do the code in Java. Do not use Java's search or sort methods, or any Java data structure libraries! Place textual responses for questions 3, 4 and 5 in a separate text/pdf file!

This assignment should use the Command-Line, reading the String[] provided to *main*. **Do not use Scanner, do not use anything else**. Command-Line input is what you must use. Learn about String[] args in the main method!

Posted: Friday, Jun 29th

Due: Sunday, Jul 8th

Grade: 5%

1. Given an array of unsorted integers, for example *99 37 17 5 12 33*, write a program that performs an Insertion Sort and outputs the results in ascending order. This program should be in $\Theta\left(n^2\right)$ for time in the average case.
   (a) Make sure your program allows as input on the command-line a space-separated list of integers (the above being just an example).
      i. The file name should be ISort.java
      ii. format: java ISort <vals...>
      iii. e.g.: java ISort 99 37 17 5 12 33
   (b) When the program completes, it should display a line with the sorted values, and indicate how many nanoseconds it took to sort the values, and how many comparisons between values were made. Also show the sorted values;
2. Write a second version of the program that does a "structuring" pass first, and then calls Insertion Sort written above, but starts after the first run. It should walk through the unsorted values looking for ascending or descending runs. Runs should be output, giving a starting index (inclusive) and an ending index (exclusive), as well as a direction. Runs in descending order should be reversed. A final run of length one can be ignored if it exists (don't print it).
   (a) Allow for the same input as used with ISort.java
      i. The file name should be ISSort.java
      ii. format: java ISSort <vals...>
      iii. e.g.: java ISSort 99 37 17 5 12 33
   (b) When the program completes, it should display a line with the sorted values, and indicate how many nanoseconds it took to sort the values, and how many comparisons between values were made. It should also display what structures it found, one per line. It should also display the sorted values.

Insertion Sort Example

```
%java ISort 99 37 17 5 12 33
Sorting: 99 37 17 5 12 33
Completed in 6943ns
Comparisons: 13
Sorted: 5 12 17 33 37 99
```

```
%java ISSort 99 37 17 5 12 33
Sorting: 99 37 17 5 12 33
99 37 17 5 <
12 33 >
Completed in 144891ns
Comparisons: 12
Sorted: 5 12 17 33 37 99
```

!!Don't implement columns! I'm just saving space here.

[1] https://fis.encs.concordia.ca/eas/

3. In clear, natural language, describe the average runtime of this program after you have empirically tested it with random data up to 500000 values (don't use debug or your times will be really bad). Calculate an operation cost based on this. Show your data table! Do this for both algorithms, with at least 20 data points, testing big enough values of n to see the asymptotic shape.
   (a) This textual response should be no more than 5 lines / 50 words.
   (b) Include as your data table the length of the array, the times run and the average time in ns (on each line)
4. Show the table of data for the comparisons for both ISort and ISSort, similar to the times above.
5. In clear, natural language, describe how the Structured Insertion Sort made things better or worse, referring to both runtimes and the number of comparisons, and mentioning the relevant elements of asymptotic analysis.
   (a) This textual response should be no more than 5 lines / 50 words.