Assignment 1
Due: Sep 28, 2023 at 11:59PM

1 Assignment 1 Specification

1.1 Goals

1. Practice converting pseudocode to Java
2. Implement and test two sorting algorithms

1.2 Implementation

The skeleton code at `https://classroom.github.com/a/6kImbZvk` is the start of a program to sort integers quickly. Complete the program by implementing counting sort and radix sort, as presented in class and the lecture slides. Your program will take a character input for the name of the sorting algorithm followed by a list of integers to sort. You can read through the code and experiment with the two already-implemented sorting algorithms. To complete the assignment, replace the three **TODO** method stubs with the indicated algorithms.

Accept the empty `https://classroom.github.com/a/Muo0S76-` project as well. This is where I will post your graded lab worksheets.

Write comments for at least those three methods. Add comments to any other blocks of code you write that might need additional explanation. You can write additional methods to help with your solution if you want – declare them as **private**, as is good practice with helper methods. The minimum and maximum values possible in the input are 0 and 262143 ($2^{18} - 1$).

For your radix sort implementation, you **must** use both a count array and position array, as demonstrated in class.

If you are stuck on how to implement either algorithm, try to complete sections of the algorithm and verify that those sections are working for a small example problem. For example, in radix sort you can solve subproblems and verify that they work before moving on: populate the `count` array, populate the `pos` array, and finally move values from the input to output array. There is a visualization of the steps to complete radix sort under the lecture section of the website, as well.

1.3 Testing

You can run **A1.java** and test specific inputs to verify that your implementations are working. You can also print out internal information (like the count or position array) to verify that separate pieces of your code are working. In your submitted version, you should comment out or remove any debug print statements in your code.

For this assignment, full JUnit tests are provided in the **edu.wit.cs.comp2350.tests** package. You can run these tests to see if your code is performing correctly. Your assignment grade will be based entirely on these tests. In future assignments I will not provide all of the grading tests so you will have to be thoughtful with testing other inputs with your code.

A **ChartMaker** class is included, which will create a chart of runtimes of each sorting method based on the number of integers to sort. You can use this chart to verify that your two sorting algorithms' runtimes are what you expect. It does not check for correctness.

2 Grading

Counting sort: 50%

GetDigit: 10%

Radix sort: 40%