Assignment for Week 9: Complexity of Algorithms

Goals

The provided software implements four sorting algorithms, two `Comparators`, creates arbitrary collection of test data, and contains a `TimerTest` class which runs timing tests for all four algorithms and a selection of `Comparators` and test data.

The goal of this assignment is to learn how to design timing tests, and to reflect on the performance of these four algorithms.

Details

Do the following to become fa=miliar with the working environment. You do not need to hand in any evidence that you have done so.

1. Print for yourself a copy of the **TestSuite.java** file and the **TimerTests.java** files.

2. Read the code in the **TestSuite.java** and run the following tests:

   - `TestCity`
   - `TestInGuiRange`
   - `TestInFileRange` (use the file cities.txt or minicities.txt)
   - `TestInFileRangeBuffered` (use the file cities.txt or minicities.txt)
   - `TestInConsoleRange`

   Make intentional errors when typing in zip codes, latitudes, and longitudes, and observe how the program deals with them. (If you are curious, read as much code as you wish...)

3. Read the code for `testDataSets` and run the test. Use the file smallcities.txt. This is a test for building a collection of source data for the timer tests. You can choose different sizes of the test data.

4. Try the timer test, read the code, and find out how the timing is measured. You may want to see the Javadoc documentation for the methods and classes used.

   For the following part of the homework, you need to submit either a working code, or the output from the JPT console, or both - as indicated.

5. Design (implement and test) a comparator which compares the cities by latitude. All latitudes are positive numbers, compare them by ascending numerical order. (A new class definition and tests in the `TestSuite` class.)

6. Modify the timer tests, so they include the tests with latitude comparator. (Changes to the `TimerTests` class, and the necessary tests in the `TestSuite` class.)

7. If you run the tests several times, you observe that they produce different results each time. To get meaningful data, the tests should be run several times over the same size data, and the results should be averaged over all comparable tests (same size, same algorithm, same comparator).

   Design a method `runOneTest` in the class `TimerTests` which will for a given sorting algorithm, the given comparator, and a given size of the test data, compute an average over 20 different test data sets. Make sure you include tests for this method in the `TestSuite`.

8. Design a 'driver' for tests that will produce results similar to the original timer tests, but this time, each result will be an average of running the tests for 20 different test data sets for each instance of the test. This should be a new method in the `TimeTests` class, with tests included in the `TestSuite` class.

9. Run the tests, and save the output from the console as a text file.

10. Write a short technical statement that answers these specific questions:

    - Which algorithm(s) perform well for sorting short, randomly shuffled sequences?
    - Which algorithm(s) perform well for sorting long, randomly shuffled sequences?
    - Which algorithm(s) perform well when their input is already sorted or almost sorted?

    Include a justification of your answers, based either on the evidence gathered from the timing tests, or based on your understanding of these algorithms.

    You can append this to the output from the console and submit as a test case item with your homework.

11. Zip the code for your homework and submit as a code part of your submission.

EXTRA CREDIT

Design and implement the classes **MergeListSort** and **MergeVectorSort** by making any needed modification to your solution to the previous homework and add them to the timer tests.