Data Structures and Algorithms: Assignment 1

The goal of this assignment is to implement, study and compare a number of sorting algorithms.

1 Implementation

The file `SortingAlgorithms.java` defines the class `SortingAlgorithms`. This class contains the signatures of a number of sorting algorithms. It is your job to implement these algorithms. The javadoc comments above each method contain additional information on the method parameters and the expected return value. To make your life a bit easier, we created an `ant`[1] build script. To compile and run the main method, simply execute `ant` on the command line in the `code` directory (the directory that contains the `build.xml` file). To run the JUnit tests, execute `ant test`. Finally, the command `ant clean` removes all files generated by the aforementioned commands (i.e. the `build` directory). Feel free to use `javac` or an IDE such as Eclipse to compile and run your code instead of the ant build script.

You can use the source code given in the book or online as a starting point for your implementation. Do not modify the name of the class or the signature of the public methods. Do not modify the package declaration (i.e. the class must remain in the `gna` package). If your code does not compile or if you do not adhere to the rules described above, we do not inspect your submission any further and your grade for this assignment is zero.

The code of conduct ("gedragscode") posted on Toledo applies to this assignment.

2 Report

Perform a number of experiments on your implementation and briefly discuss your results.

2.1 Generic Sorting Algorithms

Plot (in a single scatter plot) the number of comparisons needed by selection sort, insertion sort, merge sort and quick sort to sort random permutations of

[1] `https://ant.apache.org/`

the numbers 0 to N-1, where N ranges from 0 to 100. Run each algorithm 100 times for each value of N (and include each data point in the plot). Briefly discuss your results. For example, describe for each algorithm whether your experiments match the expected outcome (based on the theory). You do not have to explain why an algorithm has a particular time complexity.

Plot (in a single scatter plot) the number of comparisons needed by selection sort, insertion sort, merge sort and quick sort to sort ordered arrays containing the numbers 0 to N - 1, where N ranges from 0 to 100. Again, briefly discuss your results.

Hint: There are several ways to create charts. For example, you can write your data to a file, import this file into a program such as Microsoft Excel, gnumeric or gnuplot, and create a chart using this program. Alternatively, you can use a Java library such as JFreeChart (available at `http://www.jfree.org/jfreechart/`) to create charts directly in your Java code.

2.2 K-way Merge Sort

Plot (in a single scatter plot) the number of comparisons needed by k-way merge sort to sort random permutations of the numbers 0 to N-1, where N ranges from 0 to 100, for k is 2, 3 and N. Briefly discuss your results.

Plot (in a single scatter plot) the number of data moves needed by k-way merge sort to sort random permutations of the numbers 0 to N-1, where N ranges from 0 to 100, for k is 2, 3 and N. Briefly discuss your results.

2.3 Doubling ratio experiment

Run a doubling ratio experiment (as described in the book) for insertion sort. Include a table in your report showing the running time and ratio for increasing array lengths. What is the expected ratio for insertion sort? Why? Do your experiments confirm this ratio? Based on your experiments, predict the running time of insertion sort on an array of length 512000 on your computer.

Run a doubling ratio experiment for quick sort. Include a table in your report showing the running time and ratio for increasing array lengths. What is the expected ratio for insertion sort? Why? Do your experiments confirm this ratio?

3 Submitting

You must upload your solution to this assignment via Toledo before Friday March 15th 2013 12pm. Your solution is a single zip file which is created as follows:

- Place your report (pdf) in de `code` folder. The report must be stored in a file named `report.pdf`.
- Navigate to the `code` folder.

```
user@machine:~\$ cd code
user@machine:~/code\$ ls
build.xml  lib  report.pdf  src
```

- Compile your code and runs the JUnit tests (`SortingAlgorithmsTest.java`).

```
user@machine:~/code\$ ant test
```

- Create a zip-file. (Note that this step fails if your code does not compile or if a test fails.)

```
user@machine:~/code\$ ant release
```

  The zip file is placed in the `build` directory. Replace the name of the zip file with your own first and last name.

- Check that the zip files contains `report.pdf` and all Java files in the `src` directory.
- Upload your solution via Toledo.
- Print your report and drop it in the box at the secretariat

Questions about this assignment can be asked via the discussion board on Toledo.

Have fun!