Computer Science 112

Computer Science with Java II

Spring, 2016

Community College of Philadelphia Computer Science

Lab Report – Week [6] - [Benchmarking Sorting Algorithms]

[Leo Vergnetti]

[CSCI 112-online] [Spring]

Assignment Analysis and Design

The assignment was to do benchmarking testing on the various sorting algorithms we've covered so far in class. Each algorithm would sort an array of randomly generated elements 100 times. Each time this completes successfully, the a larger dataset would be tested. The datasets were 10000, 20000, 100000, 200000, 1000000 and 2000000 elements in length. The program is organized into several static methods. First, there are five methods to sort data using the five sorting algorithms we've covered so far: bubble sort, selection sort, insertion sort, merge sort and quicksort. There is a method to generate a random array to user desired length. I declared a set of constants to hold the current sorting mode being tested, which i use for code readability while iterating through each one. The lengths of the datasets to be generated and sorted are stored in an array called arraySizes. I use an for statement to iterate each sorting algorithm, and an inner for statement to test for each specified dataset. The testSort method is the actual method that conducts the testing. It accepts three arguments, an integer to keep track of the sorting mode, an integer to specify the array size, and an integer to keep track of the desired time out length in seconds. First, it sets up a few long integers to be used for record keeping, and a boolean value called lengthOk. The method returns this boolean upon completion, which informs the calling method if the test executed successfully or if it was terminated. The timer is started after a switch statement to avoid the slight delay that would be caused while the calling program fell through each one. Finally when the 100 arrays are sorted the program outputs the best case, worst case, and average case, and returns true. If it took too long (60 seconds for this assignment) then the method cuts off and returns false, which is caught in the main method and the next sort method is tested.

Assignment Code

Source code included in attached file.

Assignment Testing

In order to test this project, I had to test each sorting method first. I constructed a method to output the sorted arrays to ensure that they were sorting properly. This was deleted after the assignment was completed. The testSort method was checked simply by checking that the formatted output was looking presentable. I also had to make sure the method was exiting properly when the dataset was taking too long to sort. I was able to test this in smaller time scales by setting the timeout length variable to 20 seconds, rather than 60 seconds, which allowed for quicker testing.

Assignment Evaluation

This assignment took a long time to complete. It was helpful to actually see how the algorithms measure up against each other. Designing the testSort method and main method took longer than expected. I tried to make it visually easy to understand how the program functioned. I was a little frustrated with how many static methods were jammed into the class, but given the nature of the assignment, I couldn't think of a more elegant way to write it . The design allows for the easy integration of additional sorting methods / testing methods with minimal modifications to the existing software (aside from implementing the methods themselves). All in all, while this project was difficult I'm proud of the design.