CMPSC 112
Introduction to Computer Science II
Spring 2015

Laboratory Assignment Six: Evaluating the Performance of Sorting Algorithms

Introduction

Sorting algorithms are commonly used in a wide variety of practical computer applications. In this laboratory assignment, we will learn more about sorting while also continuing to familiarize ourselves with the empirical analysis of algorithms. In particular, we will analyze the performance of five sorting algorithms as they are applied to randomly generated input data sets. Students who would like to learn more about sorting algorithms are encouraged to visit the following Web site: `https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html`.

Accessing and Using the Sorting Benchmarking Framework

To start this laboratory assignment, you should return to the `cs112S2015-share` Git repository and type the "`git pull`" command in the terminal window. Now, you should have a `lab6/` directory that you can explore further. Once again, please make sure that you can find the source code in this new directory and you understand why the directories in the assignment are structured the way that they are. Next, you should use GVim to study the source code in the `build.xml` file. As in the past assignments, when editing a Java program you can type "`:Ant compile`" in your GVim window and it will compile all of the Java classes and save the bytecode in the correct subdirectories inside of the `bin/` directory. Please see the instructor if you cannot get this to work.

Adding Features to Support Experimentation

As you study the provided source files, you will notice that they contain several sorting algorithms (e.g., `BubbleSort`) and a `SortingExperiment` class that facilitates experimentation. Currently, the `SortingExperiment` class provides methods that generate random arrays of `int`s, call the appropriate sorting routines, and output all of the results. By default, the benchmarking framework runs each sorting algorithm five times and reports the average time needed to run the algorithms.

Currently, the `SortingExperiment` is hard-coded to sort `int` arrays of size ten. Furthermore, the current values of the configuration variables cause it to run a specified algorithm for five trials. While the `verbose` parameter is currently a command-line argument, the other variables (i.e., `numExperiments` and `experimentSize`) are not. As such, you should add a complete set of command-line arguments that will make it easier to conduct experiments with sorting algorithms. What command-line arguments would best support experimentation with the sorting algorithms?

Conducting Experiments to Evaluate Efficiency

Before you start running experiments to evaluate the efficiency of the different sorting algorithms, you should run the `SortingExperiment` and make sure that each algorithm works correctly. What is the command line that you will need to type to run the sorting experiment? How can you tell that the sorting algorithms are working correctly? Once you are convinced that the algorithms sort

correctly, you should study the source code of each algorithm in order to form hypotheses about their performance. What algorithm will be the fastest? Which one will be the slowest? How did you arrive at these conclusions? You should intuitively answer these questions by looking at how the algorithms use iteration constructs and conditional logic. For instance, you can look at the way in which the `bubbleSort` method in the `BubbleSort.java` file uses a `while` loop, a `for` loop, and an `if` statement. Is `bubbleSort` likely to be fast or slow? How do you know?

To conduct your experiments, you should run `SortingExperiment` in the terminal window five times, for a total of twenty-five different runs for each algorithm. Each execution of the `SortingExperiment` should display information about the execution time, in milliseconds, of the chosen sorting algorithms. You should create tables of data that record all of the execution times for all of the sorting algorithms. In particular, you must run the sorting algorithms for arrays of size 10, 100, 1,000, 10,000, and some value larger than 10,000. Please note that experiments with arrays larger than size 100,000 could be very time consuming because some algorithms are very slow.

You should write a detailed report on your experimental study of the sorting algorithms. Your report should review your experiment design by clearly describing the commands that you typed and the order in which the they were typed. It should also include tables of results that list the running times for each of the sorting algorithms. Please make sure that your report has tables of data and paragraphs of written analysis that are properly justified. You should label each of your tables (e.g., "Table 1") and then reference the tables inside of your report. Students are also encouraged to create graphs that visualize the experimental results.

Your report should note trends that you found in the data set and attempt to explain why these trends are evident. For instance, you must mention which algorithms tend to exhibit the best, worst, and mid-level performance values. Your report should also state which algorithm you would recommend using and the situations in which this algorithm would be suitable. Students can earn extra credit by completing tasks like implementing and evaluating a new sorting algorithm.

Summary of the Required Deliverables

This assignment invites you to submit a signed and printed version of the following deliverables:

1. A description of all of the command-line arguments supported by your benchmarking tool
2. A sample output from running `SortingExperiment` in all of its relevant configurations
3. The final version of the commented source code for the entire benchmarking framework
4. A comprehensive written report that fully explains the results of your experimental study
5. A reflective commentary on the challenges that you faced when conducting the experiments

Along with turning in a printed version of these deliverables, you should ensure that everything is also available in the repository that is named according to the convention `cs112S2015-<your user name>`. Please note that students in the class are responsible for completing and submitting their own version of this assignment. While it is acceptable for members of this class to have high-level conversations, you should not share source code or full command lines with your classmates. Deliverables that are nearly identical to the work of others will be taken as evidence of violating the Honor Code. Please see the instructor if you have questions about the policies for this assignment.