CMPSC 112
Introduction to Computer Science II
Spring 2015

Laboratory Assignment Seven: Doubling Experiments to Assess Time Complexity

Introduction

The current module of the course has stressed the importance and purpose of both empirical and analytical evaluations of algorithm performance. For this laboratory assignment, we will connect these two evaluation strategies as we investigate the idea of a doubling experiment. First, we will confirm the results that the authors of our textbook reported on the performance of two methods for string concatenation. Then, you will select an algorithm from those provided, implement your own doubling experiment framework, conduct your own experiments, collect tables of data, and draw well-supported conclusions about the worst-case time complexity of the chosen algorithm.

Accessing and Using the String Experiment 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 `lab7/` 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.

After you have carefully reviewed the source code for `StringExperiment.java`, you should compile and run this program, ensuring that you correctly set the `CLASSPATH` as you complete this task. Please run the `StringExperiment` multiple times and record the data tables that it produces. What trends do you see in this data set? How do your tables of data compare to the one that the authors present on page 152 of your textbook? Can you clearly explain why these data values are evident in your data set and the one produced by the textbook's authors?

Creating and Using Your Own Experimentation Framework

You should notice that the course repository contains three additional Java files: `ArrayFind.java`, `ArrayMax.java`, and `DisjointSet.java`. The classes all contain new methods that are amenable to an empirical determination of their worst-case time complexity. For instance, `ArrayFind.java` contains a method called `public static int arrayFind(int[] data, int val)`. Pick one or two of the methods that are provided by these Java classes and implement a simple doubling experiment tool that follows the pattern of `StringExperiment.java`. That is, your tool should repeatedly double the size of the input to one of these methods and then report how the execution time varies as the input increases in size. Since the methods furnished by these three Java classes take arrays as inputs, you will additionally need to implement an array generation method that can produce arrays, of a specified size, with randomly produced `int` values. Please see the course instructor if you have questions about the task of implementing your own framework.

Don't forget that the worst-case time complexity of a method may only become evident when the size of the input is large enough. As such, you may need to carefully pick the starting size of the generated array and/or the number of trials to ensure both that your chosen method(s) must handle big arrays and that you can make a firm conclusion about time complexity. Depending on the method that you select, you may encounter issues where the Java virtual machine runs out of heap space because it is allocating very large arrays. If this does happen, then you should contact the course instructor to learn more about how to increase the maximum heap size.

It is worth pointing out that your textbook contains several useful insights into the pattern that you should follow when making observations about time overhead. For instance, when describing the results from running the `StringExperiment`, page 153 notes that "[A]s the value of n is doubled, the running time of `repeat1` typically increases more than fourfold." What does this suggest about the likely worst-case time complexity of the `repeat1` method? Additionally, page 172 includes the following statement when describing the performance of `repeat2`: "the running times in that table ... demonstrate a trend of approximately doubling each time the problem size doubles." Again, what would this observation suggest about the likely worst-case time complexity of `repeat2`?

As you study the results from running your own doubling experiments, you should use these sentence patterns when you characterize the ratio between the execution time of running the algorithm with an array of size $2 \times n$ to the time needed to run the algorithm with one of size n. After you finish implementing and using your own doubling experiment framework, you should write a report that contains both your data tables and a detailed commentary on how these results are suggestive of the worst-case time complexity of your chosen methods. Whenever possible, you should confirm the correctness of your results by either proving the "big-Oh" of your method or citing and explaining a textbook page and/or Web site that contains the appropriate proof.

Summary of the Required Deliverables

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

1. A sample output from running all doubling experiments in all of their relevant configurations.
2. A description of all of the configurations supported by your own doubling experiment tool.
3. The final version of the commented source code for your own doubling experiment tool.
4. A comprehensive written report that fully explains the results of your experimental studies.
5. A reflective commentary on the challenges that you faced when implementing your program.
6. 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.