CMPSC 112
Introduction to Computer Science II
Fall 2016

Laboratory Assignment Seven: Studying the Performance of Iteration and Recursion

Introduction

The Fibonacci sequence includes the numbers in the integer sequence that develops in the following fashion: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$. More formally, we can define the n-th Fibonacci number, denoted F_n, by the following equation $F_n = F_{n-1} + F_{n-2}$. In this laboratory assignment, you will explore and extend an implementation of iterative and recursive algorithms for calculating the numbers in the Fibonacci sequence. Moreover, you will investigate how the bit-depth of the variable that stores the output of the Fibonacci calculator can influence the correctness of the final result. After practicing how to conduct a detailed experimental study of an algorithm, students will develop and refine their writing skills as they create a comprehensive report of their results.

Accessing and Using the Fibonacci Benchmarking Framework

To start this laboratory assignment, you should return to the `cs112F2016-share` Git repository and type the "`git pull`" command in the terminal window. Now, you should have a `lab7/FibonacciBenchmark` 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 assignment, you can type the command "`ant compile`" in a terminal and it will compile the three 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. Now, make sure that you study the three files that are provided to you via Git.

As part of this assignment, you will examine the following different implementations: (i) `RecursiveFibonacci` using `int`, (ii) `RecursiveFibonacci` using `long`, (iii) `IterativeFibonacci` using `int`, and (iv) `IterativeFibonacci` using `long`. The `UseFibonacci` class relies on the `Profiler` class is in the `profiler.jar` file in the `lib/` directory in the `FibonacciBenchmark` directory. This means that the `UseFibonacci` program will not run correctly unless you have both the `bin/` directory and the `profiler.jar` archive inside of your `CLASSPATH` environment variable.

Try to execute the `UseFibonacci` program for different input values from 0 to 50. You should see the output from the computation and timing information that is related to the execution of the different algorithms. For example, try to execute the following command in your terminal window: "`java edu.allegheny.benchmark.UseFibonacci 10 all`". What is the output's meaning?

Implementing Additional Features

If you study the source code carefully, you will see that the `UseFibonacci` program currently accepts two command-line arguments. What are they? As part of this laboratory assignment, you must extend the `UseFibonacci` class and the command-line arguments that it can accept. You should add a third command-line argument that corresponds to one of the words `int`, `long`, or `all`. After

`UseFibonacci` extracts this new command-line argument, it should use this additional information to run a specific experiment. One valid execution might be: "`java edu.allegheny.benchmark.UseFibonacci 10 all long`". This command line would indicate that you should only execute the methods within `RecursiveFibonacci` and `IterativeFibonacci` that operate on variables with the `long` data type. You will need to add conditional logic to `UseFibonacci` in order to correctly add this new feature. Finally, enhance the code so that it runs an experiment for multiple trials.

Conducting Experiments to Evaluate Efficiency

Now you are ready to conduct an experiment to evaluate the performance of the four separate configurations of the Fibonacci algorithms. You should execute the program with ten different input values from 1 through 50. For each of the ten input values that you select, you should run the experiment for five trials and report the arithmetic means. Make sure that your experiment determines when the `int` primitive type can no longer represent the final answer that is returned by the method that calculates the Fibonacci number. Your experiment also should identify whether the iterative or the recursive algorithm is more efficient. Finally, you should try to determine whether the use of `long` or `int` yields better performance for the iterative and recursive algorithms. Whenever possible, your report should explain why these trends are evident in your data sets.

Your report should also explain your experimental goals and design by clearly describing the commands that you typed and the order in which you typed them. It should also include tables of results that list the running times for each of the different configurations of `UseFibonacci`. Make sure that you label the tables and directly reference them in the text of your report. Reports will will receive grade reductions if they contain tables of data and the paragraphs of analysis that are not properly formatted. Please remember that you should not solely submit the source code of your program and the output from running it. Instead, you should also write a comprehensive report that identifies and explains the noteworthy trends in your data sets. Students who want to learn more about recursion should review Sections 5.1 through 5.5, paying close attention to Section 5.5.

Summary of the Required Deliverables

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

1. Using diagram(s), an explanation of how recursion works in the Java programming language.
2. A short discussion of the different primitive types that are available in the Java language.
3. The final version of the commented source code for your Fibonacci 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 enhancing the benchmarks.
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 `cs112F2016-<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.