Computer Science 501
Data Structures & Algorithms
The College of Saint Rose
Fall 2015

Lab 3: Timing Java

Due: 6:00 PM, Wednesday, September 23, 2015

Your primary task this week to write a Java program or programs that will perform emprical analysis of a number of operations. This is the first part of a project that will culminate in you taking timings of these operations, which you can analyze in comparison with the theoretical expectations, and to write up a formal report of your results.

You may work individually or in a group of 2 or 3 on this project.

Getting Set Up

To get your BlueJ environment set up for this week's lab assignment, start BlueJ and choose "New Project" from the "Project" menu. Navigate to your folder for this course and choose the name "Lab3" (no spaces) for the project.

Create a document where you will record your answers to the lecture assignment and lab questions. If you use plain text, call it "`lab3.txt`". If it's a Word document, you can call it whatever you'd like, but when you submit, be sure you convert it to a PDF document "`lab3.pdf`" before you submit it.

Lecture Assignment Questions

We will usually discuss these questions at the start of class on the lab due date, so no credit can be earned for late submissions of lecture assignment questions.

? LA Question 1:
Bailey Problem 5.2, p. 112 (2 points)

? LA Question 2:
Bailey Problem 5.4, p. 112 (2 points)

? LA Question 3:
Bailey Problem 5.8, p. 113 (2 points)

? LA Question 4:
Bailey Problem 5.10, p. 113 (3 points)

LA Question 5:

Suppose you have $n > 2$ identical-looking coins and a balance scale. All of the coins also weigh exactly the same amount, with the exception of one of the coins, which is counterfeit. However, you do not know which coin, nor do you know if it is heavier or lighter than the others. Design a constant-time algorithm to determine whether the counterfeit coin is lighter or heavier than the others. (3 points)

Timing Java Code

Read the lab description on pages 115-117 of Bailey. We will not be doing this lab specifically, but it explains the ideas and techniques you will need to be able to write a program that can generate good timing data.

Experiments

You will be designing, implementing, and running code to support experiments to analyze the efficiency of each of the following operations:

- The construction of an n-element array of `int`.
- Inserting n numbers into an n-element array of `int`.
- Accessing the element at index 0 of an n-element array of `int`.
- Accessing the element at index $n - 1$ of an n-element array of `int`.
- Adding n integer values to a default `Vector` from the structure package.
- Adding n integer values to a `Vector` whose initial capacity is 1 element, and which increases the size of its internal array by 1 element each time an element is added that does not fit in the internal array. Hint: use the proper constructor of your `Vector` to achieve this behavior.
- Adding n integer values to a `Vector` whose initial capacity is 10 elements, and which increases the size of its internal array by 10 elements each time an element is added that does not fit in the internal array.
- Adding n integer values to a `Vector` whose initial capacity is 1 element, and which increases the size of its internal array by doubling each time an element is added that does not fit in the internal array.
- Adding n integer values to a `Vector` whose initial capacity is n elements. The resizing discipline is not relevant here, as the `Vector` will never need to be resized.
- Inserting n integer values into position 0 of a `Vector` whose initial capacity is n elements. The resizing discipline is not relevant here, as the `Vector` will never need to be resized.

- Accessing the element at index 0 in a `Vector` that contains n integer elements.
- Accessing the element at index $n-1$ in a `Vector` that contains n integer elements.

Question 1:
For each of the above, state and justify the theoretical efficiency you expect to observe (*e.g.*, $\Theta(1)$ or $\Theta(n^2)$). (15 points)

Programming Assignment

Write one or more Java classes that will allow you to gather timings for the operations listed above. As you proceed, here are some things to keep in mind:

- All of the bullet items in Bailey on p. 115 and the top of p. 116!
- For each of the operations you will be timing, you will want to be able to gather results over a range of problem sizes (*i.e.*, n), and you will want to run each instance (each n) many times. This is a job for automation!
  - Ultimately, you will be running each experiment hundreds or even thousands of times. You will definitely want scripts to manage this, so your programs should take some command line parameters to specify things like the value of n, the initial capacity of your `Vectors`, and the resizing discipline of the `Vectors`. And you definitely do not want to have any keyboard interaction.
  - Various experiments will require different ranges for the values of n and different numbers of repetitions, so it will not work to run them all at once. Your program should be able to run one experiment for one value of n and for a specified number of repetitions.
  - Extraneous output can complicate matters when it comes time to take your timing results and generate tables and graphs. Keep your output concise but complete.
  - A sample Java program that times the construction of `Strings` by repated concatenation and a Bash script to run the program over a variety of problem sizes are available in the class shared examples area under `TimingExample`. You are welcome to borrow parts of the program and/or script in your own work, with attribution. See the `README` file there for more information.
- There are many reasonable ways to organize code for this, but any choice will have its own advantages and disadvantages.

For this week, you do not need to present timing results, but your programs should be able to generate the results in a form that will be usable to present in both tabular and graphical formats next week. A good job on this week's programming assignment will make next week's experiments and writeup much easier!

Question 2:
Describe in some detail how to run experiments in your framework. For example, if I wanted to compute the time it would take to insert n integer values into a default `Vector`, what command line would I use? (6 points)

Question 3:
Describe in detail why you believe your program can compute accurate timings. For example, how to you account for overhead of loop code to perform an operation repeatedly? How do you decide how many times to repeat an operation? (6 points)

Submitting

Before 6:00 PM, Wednesday, September 23, 2015, submit your lab for grading. There are two things you need to do to complete the submission: (i) Copy your file with the answers to the lecture assignment and lab questions into your project directory. Be sure to use the correct file name. If you prepared your answers in Word, export to a PDF file and submit that. (ii) Email a copy of your lab (a `.7z` or `.zip` file containing your project directory) to *terescoj@strose.edu*. Please use a meaningful subject line such as "Joe Student Lab3 Submission".

Grading

This assignment is worth 70 points, which are distributed as follows:

Feature	Value	Score
LA Question 1 (5.2)	2	
LA Question 2 (5.4)	2	
LA Question 3 (5.8)	2	
LA Question 4 (5.10)	3	
LA Question 5 (coins)	3	
Question 1: theoretical expectations	15	
General Java timing framework	10	
Java code for specific operations	15	
Java code style, documentation, and formatting	6	
Question 2: description of how to run experiments	6	
Question 3: description of timing accuracy	6	
Total	70	