Assignment 3: Efficiency and Experimentation

Assigned: Friday, February 14, 2014
Due: Monday, February 24, 2014, 11:59 P.M.
Type: Team

Problem Overview

This assignment is a departure from previous assignments in that its focus is not on program construction, but instead on experimentation, analysis, critical thinking, and applying the scientific method. The provided resources for this assignment are the following.

- `Clock.java` — Source code of a class that you will use to measure elapsed time.
- `TimingLab.jar` — A jar file containing a the `TimingLab` class for which you have to experimentally determine the big-Oh running time of a method.
- `TimingLabClient.java` — Source code of a Java class that illustrates basic usage of the Clock class to measure the time required to execute a method of the `TimingLab` class. *This class is for illustration only.* You can modify it for your own purposes or you can use it as an example to create a similar class for your own purposes.
- `SortingLab.jar` — A jar file containing the `SortingLab` class for which you have to experimentally identify the five different sorting methods that are implemented.
- `SortingLabClient.java` — Source code of a class that illustrates basic calls to public methods of `SortingLab`. *This class is for illustration only.* You can modify it for your own purposes or you can use it as an example to create a similar class for your own purposes.

There are two parts to this assignment, each of which has its own deliverable and is to be done independently of the other.

Part A: Experimental Discovery of Running Time

You must develop and perform a repeatable experimental procedure that will allow you to empirically discover the big-Oh time complexity of the `timeTrial(int N)` method in the `TimingLab` class. The parameter N represents the *problem size*. Thus, by iteratively calling `timeTrial` with successively larger values of N, you can collect meaningful timing data.

The constructor `TimingLab(int key)` will create a TimingLab object whose `timeTrial` method is tailored specifically to the given key value. You will use your group number in Canvas as the key required by the constructor. For example, group 42 would invoke the constructor `TimingLab(42)` and group 23 would invoke the constructor `TimingLab(23)`, creating two distinct objects whose `timeTrial` methods would have different time complexities. You are guaranteed, however, that no matter the key value, the associated time complexity will be proportional to N^k for some positive integer k.

You must use the provided `Clock` class to gather the experimental timing data. To analyze the experimental data you must use the following property of polynomial time complexity functions $T(N)$.

$$T(N) \propto N^k \implies \frac{T(2N)}{T(N)} \propto \frac{(2N)^k}{N^k} = \frac{2^k N^k}{N^k} = 2^k \tag{1}$$

You must write a lab report that fully describes the experiment used to discover the big-Oh time complexity of the given method. The lab report must discuss the experimental procedure, data collection, data analysis, and interpretation of the data. The lab report must be well written, it must exhibit a high degree of professionalism, and it must be structured like the provided sample. Your experiment must be described in enough detail that it could be reproduced by someone else.

Part B: Experimental Identification of Sorting Algorithms

You must develop and perform a repeatable experimental procedure that will allow you to empirically discover the sorting algorithms implemented by the five methods of the `SortingLab` class — `sort1`, `sort2`, `sort3`, `sort4`, `sort5`. The five sorting algorithms implemented are merge sort, randomized quicksort, non-randomized quicksort, selection sort, and insertion sort. Insertion sort, selection sort, and merge sort are implemented exactly as described in lecture and in the note set. Quicksort has two implementations, both of which choose the left-most element of a partition (e.g., a[left]) as the pivot for that partition. Both use the same partition algorithm, albeit a slightly different one than described in lecture and in the note set. The randomized quicksort implementation makes the worst case probabilistically unlikely by randomly permuting the the array elements before the quicksort algorithm begins. The non-randomized quicksort exposes the algorithm's worst case by never shuffling the array elements.

You must use the `Clock` class to gather the experimental data. Although timing data will be an important part of your experimental procedure, it will (likely) not be sufficient to distinguish between merge sort and randomized quicksort. To do this you should think about *stability*.

The constructor `SortingLab(int key)` will create a SortingLab object whose sort method implementations are tailored specifically to the given key value. You will use your group number in Canvas as the key required by the constructor. For example, group 42 would invoke the constructor `SortingLab(42)` and group 23 would invoke the constructor `SortingLab(23)`, creating two distinct objects whose sort methods would have different implementations.

You must write a lab report that fully describes the experiment used to discover the big-Oh time complexity of the given method. The lab report must discuss the experimental procedure, data collection, data analysis, and interpretation of the data. The lab report must be well written, it must exhibit a high degree of professionalism, and it must be structured like the provided sample. Your experiment must be described in enough detail that it could be reproduced by someone else.

Teamwork and Grading

You are required to complete this assignment in teams of two, as assigned in Canvas. Each person must contribute to each part of the assignment in roughly equal amounts. Two rubrics (*Experimentation Rubric* and *Written Communication Rubric*, both available on Canvas) will be used to score each lab report. Marks of Little/No Ability, Basic, Intermediate, and Advanced are worth 1, 2, 3, and 4 points respectively. The lab report scores will account for 90% of your assignment grade. The remaining 10% will come from your attendance and participation in two **mandatory** lab sessions for this assignment: Tuesday February 18 and Thursday February 20. An official University excuse will be required for missing either lab. If you are not present for either lab without an approved excuse, you will earn zero points for the assignment. Table 1

Table 1: Sample assignment scoring.

	Mark	Score
Part A: Experiment Rubric		**3.125**
Experimental Design	Advanced	4
Data Collection	Intermediate	3
Data Analysis	Intermediate/Basic	2.5
Interpretation	Intermediate	3
Part A: Writing Rubric		**3.2**
Content	Intermediate	3
Organization	Advanced	4
Style	Basic	2
Grammar	Intermediate	3
Figures and Tables	Advanced	4
Part B: Experiment Rubric		**3.625**
Experimental Design	Advanced	4
Data Collection	Intermediate	4
Data Analysis	Intermediate/Basic	2.5
Interpretation	Intermediate	4
Part B: Writing Rubric		**3.2**
Content	Intermediate	3
Organization	Advanced	4
Style	Basic	2
Grammar	Intermediate	3
Figures and Tables	Advanced	4
Tue 18 Feb attendance	Present	5
Thu 20 Feb attendance	Present	5
Total Score		**85.5**

shows a sample assignment scoring with a final numeric grade calculated.

Assignment Submission

You must turn in a single zip file per group. This zip file must contain two PDF documents, one being the lab report for Part A and the other being the lab report for Part B. Submissions made within the 24 hour period after the published deadline will be assessed a late penalty of 15 points. No submissions will be accepted more than 24 hours after the published deadline.