CS-260 Lab #1: Sorting, Searching & Empirical analysis

General

At the top of the every source file that you create always write a class JavaDoc that includes:

```
/**
* Description of the class
*
* @author YOUR NAME
* @version CS260 Lab #, mm/dd/yyyy (replace with the last edit date)
*/
```

Concepts

This lab will give you some basic tools and techniques for ***empirical*** time (and space) analysis of programs. You will focus on two techniques; using counters for counting operations (for this lab we are counting comparisons and memory moves) and clock time measurement.

Background

Review the online text material on time and space analysis (complexity).

Assignment

For this assignment you will be learning to add counters to programs to measure targeted operations and clock time measurements to compare execution durations. Download the source code "SortSearch.java" from the class web site. This code fills an array with random integers and then provides several sorting routines (bubble, selection, insertion). You will be modifying this program.

Note: for this first lab, you can use whatever Java environment you like; although I would suggest that you get familiar with Eclipse. Please ensure that your single source file compiles correctly from the command line.

Optional challenge activity: Counting memory moves is also a very useful bit of information that impacts performance of sorting. Once you have the comparison counting worked out, you might also count the number of memory moves that occur and chart them similar to what you do with comparison counts.

Part #1: "measuring quadratic sorts"

#1) Add in counters to measure the number of ***data element comparisons*** in all of the sorting methods. You will need to provide output of these values each time the array is sorted.

#2) Add in ***clock time*** measurement of the time it takes to execute the each sort independently of each other so that you can compare them. Provide some form of output for these times.

#3) Modify the main (or simply run the program with modified values) for 10 different test cases for EACH sort, starting with an array size of 10000 elements and increasing it's size by 10000 each step until you reach an array size of 100,000. This is a linear increase in the size of the array. Track each of the time and comparison counts for all of the sorts at each step.

#4) Build a spread sheet from these figures and construct two charts, one for ***time duration***, and one for ***comparison counts*** (you should use the array size as your x-axis). So you will have a time duration chart comparing the times for all three sorts, then a chart showing numbers of comparisons for each sort. I am leaving the charting open ended to see what you come up with.

Part #2 "Linear & Binary searching"

#1) Implement and test the two shelled out searching methods, searchLinear should implement a linear search and searchBinary should implement a binary search approach. Test both methods to make sure that they are working correctly.

#2) Add in a counter to measure the number of ***data element comparisons*** for both searching methods.

#3) Add in ***clock time*** measurement of the time it takes to execute the each search.

#4)To get somewhat valid search measures, run 100 random searches (generate a random number in the range and search for it in the array) for each listsize using the same sequence of sizes as in #3 for sorting. Calculate the AVERAGE for each run of 100 random searches on each of the different list sizes.

#5) Build a spread sheet and chart the data from the AVERAGE search counts for each of the listsizes that shows a comparison between the average for linear and binary search. Similarly chart the clock time for the searches.

Submission

Turn in your modified ECLIPSE project export, and your spread sheets using the Moodle site for this class. The upload link allows for multiple file uploads. PLEASE DO NOT ZIP EVERYTHING TOGETHER, submit your spreadsheet/charts separately from your exported java project.

Submit in this lab via Moodle using the "Lab #1A" and "Lab #1B" links. Your assignment must be uploaded by the due date and time. Moodle will automatically close the link at that time. It is strongly highly recommend you do not wait until the very last minute to submit your work. Submit your exported project and the spreadsheet file with charts separately (not as one big zip).

The submission for **"Lab1A"** must show partial work on this project, including at least some work on the search methods, some work collecting comparisons and time on the sort methods, and some work on the overall data collection needed for this lab. The submission for "**Lab1B**" will be the completed project, which will be the exported Eclipse project and the spread sheet (Excel, OpenOffice, or LibreOffice) with data and graphs.