CPSC 6109 Algorithms Analysis and Design

Assignment 02: Big-Oh Exploration

Possible points: 100

Assignment source: http://csc.columbusstate.edu/carroll/6109/projects/project1.html.

Objectives:

- Execute sorting algorithms
- Practice algorithm analysis
- Gain an appreciation for what the growth rate of a function means

Background:

Algorithm analysis gives us the foundation to evaluate and compare different algorithms in terms of their general execution time as the input increases.

Description:

For this project, the main focus is on comparing theoretical growth-rate functions verses empirical timing results to provide a deeper understanding of both sorting algorithms and growth-rate functions. To do this, execute sorting algorithms in Java and measure their execution times on random data. Then, compare the expected trends and the observed trends. Report on the similarities or differences. Note, the focus is not on the speed of the algorithms, but how their execution times change as the input size changes. For example, do your empirical results support that the algorithm you ran follows linear logarithmic growth, quadratic, linear or something else?

Requirements

1. Implement and run four of the sorting algorithms. At least one of these algorithms must have a growth-rate function different from the other(s).
2. At least two of them are implemented using recursion.
3. Execute each sorting algorithm with various number of inputs to establish trends (for example, between 5/7 different input sizes). Only use input sizes that result in times greater than or equal to 10 milliseconds.
4. Execute multiple runs for each data point to minimize unrelated artifacts
5. Use the same input values (and their order) for each algorithm. **Make sure that you do not pass a sorted array to be sorted.** Assigning an array reference in Java to another array does NOT copy the values in the array. Choose multiple input sizes with the upper bound requiring at least multiple seconds to execute to get representative timing results.
6. You need to write a report of your analysis. Your report needs to have at least an introduction, methods, results, and discussion sections.

7. To facilitate grading, have one or more `class`es but just one `main` method.
8. Your program needs to NOT require input but calculate all of the timings without user input.

Rubric:

```
Points      Item
----------  -------------------------------------------------------------
_____ / 30  Java Program
          ▪ Documentation: (written for another software developer)
                ▪ All source code files include Javadoc header block with
                  description, @author, @version, etc.
                ▪ Javadoc (with block tags, for example @param, @return)
                  before each method
                ▪ All non-trivial variables are commented
                ▪ Comments included before major portions of code
          ▪ Four sorting algorithms implemented
          ▪ Timing code

_____ / 18  Experimental Design
          ▪ Multiple replicates
          ▪ Sufficient data to establish trends (including multiple values of
            n)
          ▪ Each sorting algorithm sorted the same values original (and not a
            sorted version)

_____ / 50  Report
          ▪ Appropriate sections (e.g., Introduction, Methods, Results,
            Conclusion)
          ▪ Data reported (graph / table)
          ▪ Analyze and discuss the similarities/differences between
            empirical and theoretical results (this could include a graph of
            expected execution times for each growth-rate function)
          ▪ Appropriate length (about 1-2 pages plus graph(s)/table(s))

_____ /  2  Completed rubric (estimates for each line including hours spent)

_____ /100  Total


_____  Approximate number of hours spent

I, (REPLACE WITH YOUR FULL NAME), affirm that the code that I submitted is my
own work and that I did not receive help that was not authorized by the
instructor.
```

Copy and paste the above rubric into a file named `rubric-project1.txt` file (not a `.docx`, `.doc` nor `.rtf` file). Think of this as a checklist to ensure that you receive the maximum points possible. For each grade item, fill in your estimate for the grade you deserve. Additionally, include your estimate of how many hours your spent. Lastly, replace "`(REPLACE WITH YOUR FULL NAME)`" with your full name to indicate that what you are submitting is entirely your own work.

Submission

Before submitting this assignment, make sure that there are no compilation errors and that (if you're using NetBeans) `DEBUG` is set to `false`. If there are errors, fix them before submitting. You may need to edit and compile your program multiple times before it compiles successfully. Verify that the output is correct. Once you are able to successfully run the program with the correct output, you are ready to submit the program. Submit 1) your source code (as one or more `.java` files), 2) report and 3) the `rubric-project1.txt` to the appropriate Assignment tab/folder in CougarVIEW.

Notes

1. Double check that the code does not have a reference to the original array of random values. If it does, the other algorithms will be sorting a sorted array and that can be very different from a random array for most sorting algorithms. You could temporarily print out values for a small input size to verify that they are indeed random before passing it to the second sorting algorithm.
2. You are **NOT** required to write the sorting algorithms. If you use published code, make sure that you indicate the source of it as a comment in your code and in your report.
3. Make sure that you start the time measurement just before sorting and end it directly after. Do not include any I/O statements (e.g., `System.out.print`*) in the elapsed time.
4. Your code does not have to generate the graphs. You MAY use **python** to only generate the plots using the output data. (That would be cool but not required.)