Programming Assignment 3

Posted on February 17th, due on March 12th

The goal of this programming assignment is two-fold. The first goal is to observe empirically the complexities of different algorithms solving the same problem. The second goal is to discover how accurate the theoretical estimates of complexity are when compared to real execution times.

You can implement the code in C, C++, Java. For other programming languages, please consult with the instructor. In this assignment, you will implement the following three sorting algorithms: **INSERTION-SORT**, **MERGE-SORT**, and **QUICK-SORT**. The implementation should follow *exactly* the algorithms (pseudocode) learned in class.

The final submission will consist of:

- All source code, submitted on Blackboard
- A report consisting of:
    1. A graph showing a comparison of the running times of the three algorithms for various input sizes (see below for details)
    2. Three tables, one for each algorithm, showing a comparison of the actual and theoretical running times. For each algorithm, compute an approximation of the hidden constant in the O-notation

More details:

- !!! Note that the assignment is explained in class on February 17th, 2015
- Run experiments varying n (where n is the number of elements) from 100 to 1000, with increment of 100. For each n value, run each algorithm $m = 5$ times on different input arrays and take the average of the running times.
- To generate numbers, use a random number generator. Then for each array instance run the algorithms. You can use rand() which generates a random number between 0 and RAND_MAX = 32767. Include #include<stdlib.h>.
- To measure the execution time of your algorithms, you should measure the time in microseconds (μs). If you use unix, you can use for example "gettimeofday", which expresses the time in sec and microseconds. Use "man gettimeofday" to see the manual description.
- You need to plot the running time of the algorithms as a function of the number of elements n
- Approximate the value of the hidden constant in the O-notation by dividing the running times with the theoretical values and then taking the maximum value over all input sizes.
- File format accepted: Microsoft Word (.doc files), PDF files, ZIP files.

Grading

- The three algorithms implemented correctly: 30 points

!!! Note that algorithms implemented differently than those designed in class will receive 0 points.

- Computation of the average RT implemented correctly: 20 points
- Report containing the graph and three tables: 40 points
- Maximum Total: 90 points