# CSC 232: Data Structures and Algorithms

**Lab 6: Runtime Analysis**

**Due: Today by midnight**

*Submit your Excel spreadsheet through BB before midnight on the due date. Email your programs to me* ***as a last resort*** *if you experience problems with BB.*

*NOTE: Questions to be answered and turned in are in bold red type.*

The purpose of this lab is to:

* Generate and analyze data for two functions, one you will write.
* Experiment with different algorithms to determine that Big-O alone is not the only reason to choose an algorithm.
* Learn how to time a segment of code.
* Be exposed to different sorting algorithms and their characteristics.
* Another good source of sort info and animations is Wikipedia.

**Requirements**

1. **Experiment with different sorting algorithms.**
   1. Open an excel spreadsheet. This is your *only* deliverable for this lab. Put your name in cell A1. Put the question numbers, starting with a “1” in cell A3, in column A. Put your answers to the questions below starting in cell B3. Put additional answers for a question in columns after B.
   2. **Q1: Which sort took the longest to complete?** Go to <http://www.sorting-algorithms.com/>. This site has animations and excellent information about different sorting algorithms. Change the Problem Size to 30, and click on the large two arrow icon in the upper left corner of the table. If you didn’t catch which took longer, run it again. There is not one correct answer – the point is which took longer by your observation.
   3. **Q2: How many seconds (approximate) does insertion sort take for Random, Nearly Sorted, and Reversed? Which was the fastest?** Clicking inside a box at a row/column intersection runs only that sort. Get your phone out and run a stopwatch app (or approximate by counting Mississippi’s). Run insertion sort on each of the three inputs listed in the question.
   4. **Q3: Answer Q2 for bubble sort.** The algorithm works by repeatedly moving (bubbling) the smallest of the remaining items to the “top” of the set of remaining items.
   5. **Q4: What’s the worst big-O for space of these algorithms?** Clicking on an algorithm’s name at the top of a column shows the algorithm in code and provides additional information about its complexity analysis. Although we are focusing on algorithm analysis of time, algorithms are also analyzed for space. With regards to space, no algorithm that operates on an array is better than insertion sort though several are as good.
   6. **Q5: How many seconds (approximate) does quick sort and quick3 sort take for Few Unique Keys? What’s the difference between the two algorithms?** The Quick3 sort was an improvement in the 20-year old quick sort. Click on Few Unique in the bottom left of the table. You can watch all the sorts perform on data that is divided into only 4 values (or keys). Alternatively, time quick sort and quick3 sorts for Few Unique individually.
2. **Modify the provided code.**
   1. Use Lab6provided.cpp as a starting point and modify it as required for this lab. Include the comment template at the top of your program.
   2. This program initializes a vector with random values, sorts the elements in the vector, prints the sorted values, and prints the time that it took to execute.
   3. This code uses the bubble sort algorithm. Like the insertion sort, bubble sort is also O(N2).
   4. Modify the code to accomplish the following.
      1. Change the fixed size of the vector to a size that is input by the user.
      2. Using the bubbleSort() function as a model, implement a function of your own that takes linear time. It doesn’t matter what your function actually does, just make sure it is linear on the size of the input*.*
   5. Use good coding style per the Coding Style Cheatsheet.
3. **Generate and plot data for both functions.**
   1. Uncomment and use the example code for getting the execution time.
   2. Use this code as a “wrapper” around the calls to bubbleSort() and your linear function.
   3. Run your program with different values of N and record the elapsed times for each function in an Excel spreadsheet.
      1. You will need to experiment with different values of N and keep collecting data.
      2. In the spreadsheet, sort the data by N until you have enough to see an observable trend in the runtime of both functions as N increases.
      3. If your choices for N are too small or don’t cover a wide enough range you will not be able to see the difference in trends between the O(N2) function and the O(N) function.
   4. Create one chart showing plots of the collected data.
   5. **Q6: Based on your data, for what N is bubble sort better than your linear function?**