CSC 323 Algorithm Design and Analysis
Fall 2018
Instructor: Dr. Natarajan Meghanathan

Project 2: Implementation and Performance Comparison of the Bubble Sort and Insertion Sort Algorithms

Due by: Sept. 27, 11.30 AM

Implement the Bubble Sort and Insertion Sort algorithms discussed in Module 1. Note that the implementation of the Bubble sort algorithm should involve the swapping-based optimization discussed in the slides.

You need to generate 'thousand' input arrays, each of size n = 1000, 10000, 100000 filled with random elements (ranging from 1 to m, where m = 500, 5000, 50000) and run the above two algorithms in an automated fashion (i.e., for all the thousand arrays of a particular size and range). For each array size (n) and data range (m), average the running time (measured in an appropriate time unit say, milliseconds or nanoseconds, clearly state it though) observed for each of the two sorting algorithms.

For each value of 'm', plot in Excel the array size 'n' vs. the average running time for each of the two sorting algorithms. If the average running times for the two algorithms are significantly different, then use logarithm scale to plot the run time of both the algorithms.

You are given a sample code to review the time functions (that were already covered in CSC 228).

Note (For C++): For each iteration, if you create the array using dynamic memory allocation (for example, shown below), then delete the allocated memory at the end of the iteration. This will prevent you from running out of memory while running the 1000 iterations, especially with larger array size.

```
int *array = new int[numElements];  // at the beginning of an iteration
............
...........
delete[ ] array; // at the end of an iteration
```

Submission (through Canvas):

Report: (i) Your programming code for the two algorithms
(ii) Screenshots of sample run of the two algorithms for a particular array size
(iii) Excel plots (as mentioned above for each value of m) and your interpretation of the plots