CPSC 350 – Data Structures
Fall 2018
Assignment #6: Sorting
Due: 12-14-18, 11:59pm

Overview:

We've learned several sorting algorithms, as well as which algorithms are "asymptotically" better than others. But though it is important to know that, on paper, a $O(n^2)$ algorithm runs slower than a O(n lg n) algorithm, it is also important to realize just how drastically these algorithmic choices can effect your program's performance at run time. To drive this point home, the (simple) assignment for this week is to implement 4 sorting algorithms and time them on large input.

The Specifics:

All of you will implement QuickSort, Insertion Sort, Bubble Sort and another sorting algorithm of your choice not implemented in class. Your program will then run as follows:

1. Take as a command line argument text file containing a list of double values. The first line of the file will be the number of items to sort.
2. Read in the values into an array. Because you don't know the size of the array at compile time, you will have to use dynamic memory allocation.
3. Run each of the three algorithms on the unsorted data (hence, you will need 3 copies of the array above). For each algorithm, output its name, the time the sort started, and the time the sort ended.
4. Write a short (1 page or less) report on your experience. Were the time differences more drastic than you expected? What tradeoffs are involved in picking one algorithm over another? How did your choice of programming language affect the results? What are some shortcomings of this empirical analysis? Include this report as a pdf with your deliverables. The report should be done in LaTex using the IEEE proceedings template (or an appropriate template of your choosing).

To implement the time stamping you can make use of the C standard library time class. Read the API for its correct use. (This is a rather crude method for timing your code, but for our purposes it should get the job done.)

The Rules:

1. You must work individually
2. Use whatever references you like, but all work must be your own.

Grading:

Grades will be based on correctness of implementation, adherence to this spec, comments, coding style, and **efficiency**.

Submissions

You know the drill by this point.