We would like to determine whether Selection Sort or Insertion Sort is faster (in both worst case and average case).

1. Implement each algorithm. You may use any reasonable programming language. You must actually write a program for each algorithm.

2. Compare the running times of your programs on worst case and random case inputs as the size of the problem increases. You should assume that there are no duplicate elements.

3. Graph your results.

4. Compare the two algorithms. What do we learn?

Here are some random thoughts:

1. Comments are considered helpful.

2. You should create random arrays of size $n$ by creating a random permutation of 1 to $n$.

3. Graphs are supposed to be helpful in understanding data.

4. It is better to use a line graph than a bar graph.

5. The independent variable goes on the bottom. The dependent variable goes on the left.

6. If you sort very large lists, which may be a good idea, then it is good to use an exponential scale for the dependent variable. The graph is then readable for all ranges of list size. You can get the same effect by taking the log of both the dependent and independent variables. This also has the advantage of graphing a polynomial as a straight line, where the degree of the polynomial is the slope of the line.

7. You will need to run a “large” number of samples for each value of $n$ (that you use), especially for small $n$. You do not want to include the overhead of starting and stopping the program and setting up the random inputs in your times. One way to do this is to run the same program without actually sorting, and subtract the times. It is plausible that an optimizing compiler could notice that you have a loop that does not produce any data and simply remove the loop altogether, so you want to be aware of this issue.

8. For some extra credit you may extend the assignment and/or include bells and whistles.

NOTE: The plan is to submit the program through the Submit Server. We are still working out the details. Also, it is possible that the assignment could be modified in insignificant ways or clarified as we progress. For example, how will we handle test cases? So stay tuned.