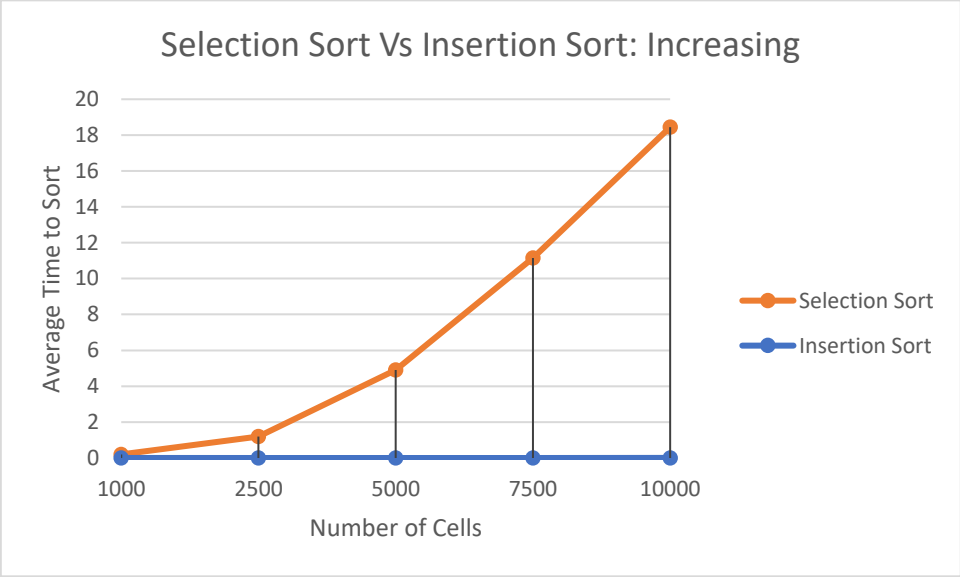
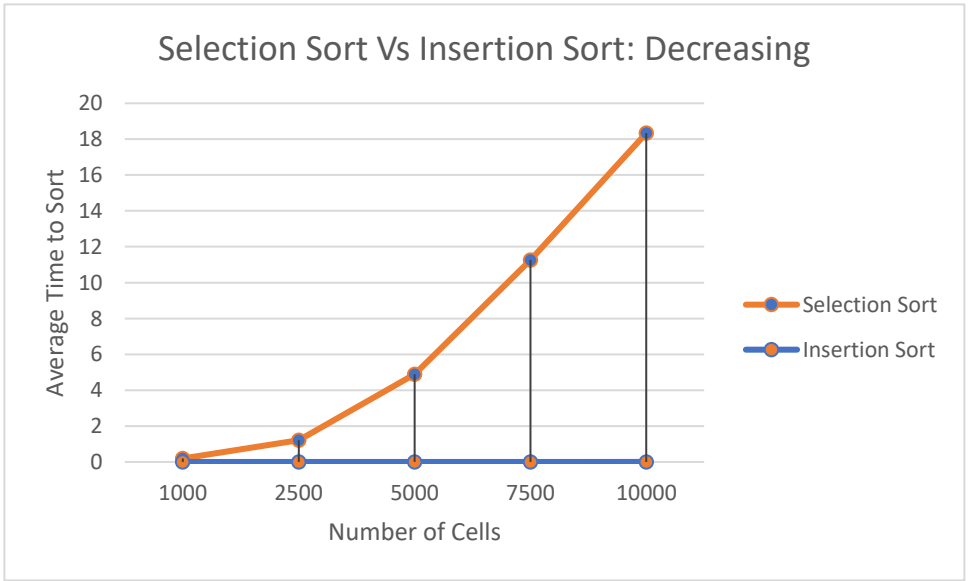
Insertion Sort, looks at the next variable from the starting position, and questions on whether or not that value is greater than, or less than the one in the current position. From here, it then determines whether or not to swap with this value, depending upon if it is of greater value. If not, it iterates to the next variable and continues the process of comparing the values until the entire range has been completely searched. The Selection Sort goes through the entire array first looking for the smallest variable. Once found, it swaps it with the beginning position and looks through the array again, looking for the next smallest number. This continues on, until the entire array has been sorted from least valuable to most valuable number.

This project was to give a feel and introductory to how these two data-sorting methods work, and how much time it takes each one to complete the task, utilizing Big-O notation as a means to calculate the overarching performance.



Looking at the first graph, you can see that Insertion Sort maintains near-linear performance of processing all of the data, no matter how big the cells become. Comparing this to Selection Sort, you can see that for every point, the amount it takes to process the next size up, the time goes up quadratically. This is due to what I discussed earlier, with regards to Selection Sort taking a look at the cell in it's entirety, and then one it has filled, the sorting method begins to proceed in swapping the values, whereas Insertion Sort, does it as the data is being inserted into the array.

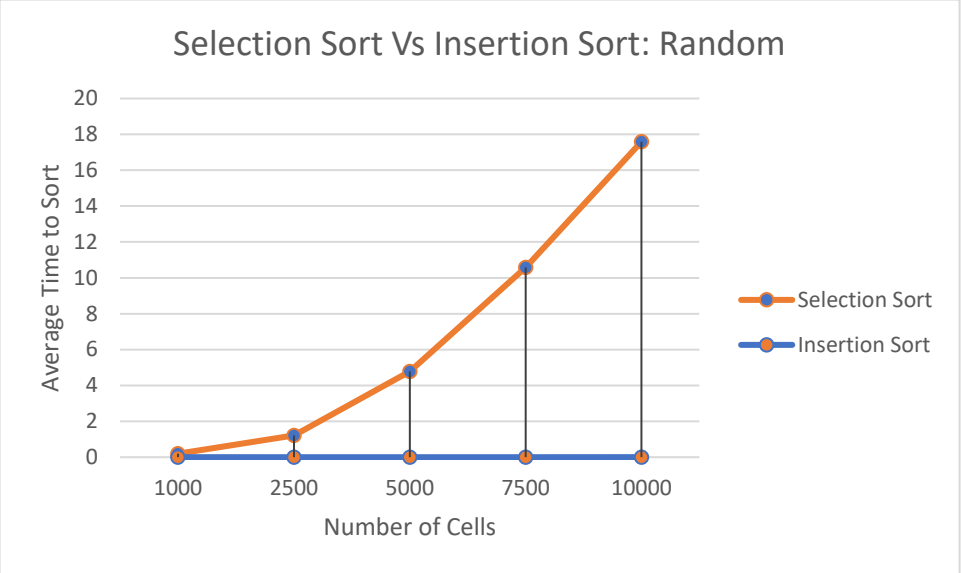
The next graph we will be looking at, deals with the Selection Sort and Insertion Sort, in decreasing order.



As similarities can be seen in the previous graph, the sorting methods take about the same time. The Insertion Sort does things in near-linear time, where as Selection Sort takes quadratic time, in order to sort the data in the array.

The final graph we will be taking a look at, is comparing the Selection Sort, with the Insertion Sort, using random numbers.

As seen in the previous graphs, you can tell that Insertion Sort does things in near-linear time, as opposed to Selection Sort, which does things in quadratic time.



Conclusion: The results of this project had great merits, as it helped me better understand how the two sorting methods differentiated from one another. Since Selection Sort sorts data by selecting and placing elements consecutively in sorted locations, this inhibits slow-time, as all the data needs to be known at the beginning, giving it a quadratic $O(n^2)$ performance. Comparing this to Insertion Sort, which sorts data by inserting it into an existing sorted file, it allows for a much easier and optimized performance, as the elements are known beforehand, it just needs to find the location to store them. Because Insertion Sort does this in "real-time" it is O(n) performance, for sorting and storing data.