# Sorting Report

Daniel Naryshev

CPSC 350-02 Assignment 6

The algorithms I used in this assignment were QuickSort, MergeSort, SelectionSort, InsertionSort, and BubbleSort. To analyze the algorithms, I generated three files of different line lengths filled with double values and recorded the speed at which each algorithm sorted in milliseconds. These were the results:

| | **1000 items** | **10000 items** | **100000 items** |
|---|---|---|---|
| **QuickSort** | 0.196 ms | 2.171 ms | 24.11 ms |
| **MergeSort** | 0.205 ms | 2.281 ms | 27.683 ms |
| **Insertionsort** | 0.845 ms | 71.88 ms | 12175.5 ms |
| **SelectionSort** | 1.687 ms | 118.358 ms | 6609.61 ms |
| **BubbleSort** | 4.475 ms | 529.237 ms | 51078.1 ms |

After implementing and testing the five different algorithms I found that QuickSort and MergeSort worked best almost every time. For data sets < 10,000, QuickSort, MergeSort, InsertionSort, SelectionSort, and BubbleSort were are all great. For data sets of  > 10,000 InsertionSort, SelectionSort, and BubbleSort, start to fall behind. Merge sort and Quick sort have the best runtime complexities, so I expected them to run well. However, what I did not expect was for BubbleSort to take as long as it did on the 100k item dataset.

One tradeoff is increasing the complexity of an algorithm generally increases its performance. Although BubbleSort was the easiest to implement, it's performance during runtime was one of the worst. Although MergeSort was the hardest to implement, it had one of the best performances during runtime.

Even though empirical analysis can provide a good visualization of performance, it still has its drawbacks. Firstly, we can't account for all possible inputs like time and resource constraints. The performance of the empirical analysis will differ because it depends on the hardware and software it is conducted on. Secondly, empirical analysis can be time consuming due to the fact that we need to implement and execute any algorithms in order to test them.