The results for the time it took to execute each algorithm on the file of random numbers was as expected. Bubble sort was by far the slowest algorithm. It took an average of 21.74872ms to execute on the file of 100000 integers. Selection sort had a much faster average at 9.162794, taking less than half of the time of bubble sort. Insertion sort took only a fraction of the time of selection sort, at 1.654775ms. Quicksort had the fastest average by an extreme margin at 0.008865955ms. After this first initial set of tests, 10 random numbers were added to the end of the sorted data, and the tests were run and timed again. Bubble sort benefitted quite a bit from the data being sorted, and only took an average of 6.77231ms to run the second time. Selection sort didn’t benefit much from the data being partially sorted, and took an average of 8.61684ms. This is hardly any faster than the first time it ran. Next is the second run of insertion sort, which took an amazing 0.000552053ms. Insertion sort works very well on data that is almost sorted, and so this was a very good use of insertion sort. Quicksort took 0.277055ms to run the second time, which actually makes it slower than insertion sort. Overall, processing the data was faster for every function when the data was almost sorted, though selection sort didn’t see a huge benefit. Quicksort is very good for sorting a large number of items which are not very sorted. Insertion sort was by far the fastest for data which is mostly sorted. Bubble sort performed the worst in every case, but did benefit from having data which was mostly sorted. Selection sort didn’t perform worse than bubble sort, but didn’t really benefit from having mostly sorted data.