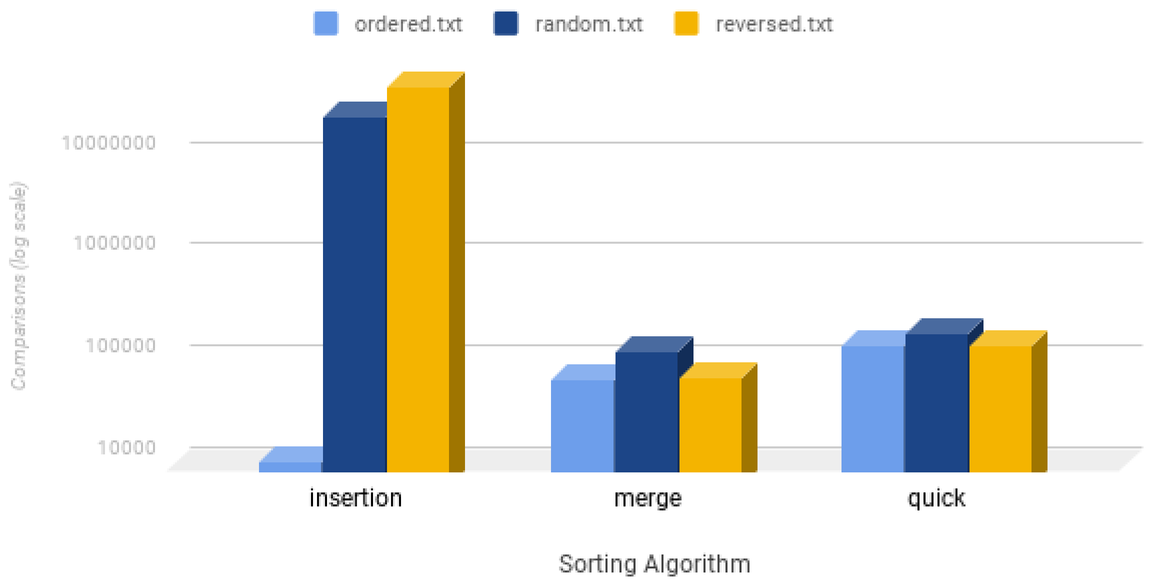
Assignment 4 – Sorting Algorithms

David Luo

This assignment involved the implementation of three different sorting algorithms - Insertion Sort, MergeSort, and QuickSort. Insertion Sort was implemented iteratively and relatively easily. MergeSort and QuickSort were implemented recursively, with helper methods to initiate the recursion process with the entire data set as input. Additionally, QuickSort was implemented using the middle item in each subset as the pivot and divisor of two partitions. All three methods sort a vector in-place, and use that as the sole data structure required, with the exception of MergeSort, which used another vector to store the sorted sub-list while comparisons on the main vector are done. Shown below is a chart and graph of each algorithm's performance on ordered, random, and reversed data.



	ordered.txt	random.txt	reversed.txt
insertion	9999	24765442	50004999
merge	64608	120535	69008
quick	143615	181626	143616

In general, MergeSort did better than QuickSort, and both did drastically better than Insertion Sort. From these results, we can conclude that MergeSort is best as a general use sorting algorithm; however, if the data is highly ordered, then Insertion Sort is much faster. Additionally, the implementation of Insertion Sort is simple, so non-performance-critical applications may also find use in Insertion Sort. QuickSort, however, may find performance improvements in a different pivot or partitioning schemes. Already, a median pivot is much faster in ordered and reversed data sets than pivots at the beginning or end of a partition, so a pivot calculated using methods such as a median-of-three may better improve performance.