I implemented merge sort, selection sort, and insertion sort algorithms and compared their performance on files of size 1,000, 10,000, and 100,000 decimal numbers.

Results

Algorithm	File Size		
	1k	10k	100k
Merge Sort	< 1 sec	< 1 sec	< 1 sec
Selection Sort	< 1 sec	< 1 sec	~ 12 sec
Insertion Sort	< 1 sec	~ 1 sec	~ 34 sec

I was surprised by the time difference between insertion sort and selection sort for the largest file. I know that they both have time complexities in O(N^2) compared to O(N log N) of merge sort. But, there is still a 20+ second difference due to implementation of the algorithms and order of the unsorted numbers.

The main tradeoff between the speed of merge sort and the other two is the auxiliary space complexity. Selection and insertion have a constant auxiliary space complexity, while merge has O(N) because of the temporary vectors.

One shortcoming of this analysis is that we are only using one test input for each size. The order of the unsorted numbers affects the runtime of the algorithms, so the specific file affects how they compare to each other. For example, insertion sort has a time complexity of O(N) for nearly sorted vectors, but is O(N^2) in this case.

I had a really rough time implementing these algorithms in C++ because I kept trying to use variable length arrays which would result in stack overflows and I could not diagnose the problem. Eventually, I read a stack overflow post (ironic) that suggested using vectors instead, and eventually I was able to get everything to work.