a. The sorting algorithm that worked best with the text file sortedWords.txt was the insertion sort, while the worst was the three-way merge sort.
b. The sorting algorithm that worked best with the text file bible13.txt was the three-way merge sort, but only by a few milliseconds lower than the traditional merge sort (after 5 runs of the program, the three-way merge sort was always faster than the traditional merge sort, but only by about 2-10 milliseconds). On the other hand, the worst sorting algorithm was the insertion sort, which peaked at almost 4000 milliseconds.
c. Insertion sort worked best on the text file that was already sorted because the spot where the word is currently at, is also the same spot where it will be inserted. There is no need to compare to the other thousands of words from the unsorted list, because the system will read that the first element of the unsorted part of the text file is the last element that will join the sorted list. On the other hand, the two merge sorts will still do its job of separating the elements into groups of 2 or 3, until they are singular, and then merge them together which, ultimately, will take more time for a text file that has already been sorted. As for the text file that has yet to be sorted alphabetically (bible13.txt), the three-way merge sort worked best because it grouped the elements into three groups at a time until they are all singular, and then merged them into one string. The reason this is faster than the traditional merge sort, albeit by a few milliseconds, is because separating into three groups until they are singular, and then merging into one, is a faster process than separating into two groups until they are all singular, and then merging into one. It is exceedingly faster than the insertion sort (which took up to 4000 milliseconds, because the system takes its time to analyze an element against tens of thousands of elements again and again.