Seth Kirksey and Christos Zourzoukis

Data Structures

Assignment 4

For the quicksort algorithm, we chose to make the pivot the last element in the array and pivot based on that. You then break the array into two parts on either side of the partition and use the last element of those arrays for the pivot of those arrays and so on until the subarrays are sorted, (are of length one). Like merge-sort, quicksort splits the array to be sorted into subarrays and is therefore a recursive algorithm. The way that quicksort uses subarrays differs however from the way merge sort uses them. For merge sort, the combine step is where most of the comparisons and sorting takes place however in quicksort most of the comparisons come from the divide step.

For the ordered file the number of comparisons for insertion sort is 9999, merge sort has 74607 comparisons, and quicksort has 99999999 comparisons. Insertion sort has a complexity of n for an ordered list while merge sort has a complexity of about 7n for an ordered list while quicksort has a complexity of n squared which is by far less efficient. For ordered files insertion sort is preferred with merge sort close behind. For the random file the number of comparisons for insertion sort is 24765442, merge sort has 23211 comparisons, and quicksort has 249989 comparisons. For the random file merge sort is the most efficient with quicksort having over 10 times as many comparisons. For the reverse file the number of comparisons for insertion sort is 50004999, merge sort has 79007 comparisons, and quicksort has 74999999 comparisons. Having seen the comparisons results I would use merge sort for sorting, because it performed the best by far on average given the variety of text orders.