Jamal Ballard

Assignment 4 Report

The way I did this assignment was to complete and check every sort type before moving on to the next one. For each sort that I tested, I would print out the first five numbers, that should have been sorted, then the last number. I finished insertion sort first. It was simplest of the sorts since it did not use a divide and conquer algorithm. The next one I implemented was the merge sort. The merge function was harder to make than I wanted it to be. Initially I had less parameters since I did not want to keep track of more variables, but I ended up getting lost because I had to change those variables more often. Also I could not get my temporary array to have a variable for its size so I set it the max size to the original array, 1000. The quick sort was the last sort I finished. I made the pivot point the middle of the array. There's no guarantee that it prevents the worst case but it makes it less likely. And for the split, I would find values greater than the spilt on the left side then swap them with values less than split that were on the right side which would continue until the left index exceeded the right one.

I did not use any other data structures other than arrays, because I feel I would not be implementing the sorts if was not in array form. For example, you can get the same properties of an insertion sort by implementing a sorted linked list but there would not be a swap property that an insertion sort does. I did use additional memory though. Additional temporary arrays had to be used in the merging with merge sort.

Provide total number of comparisons done by each of the algorithm.

- For Ordered file as input.
  - Insertion: 9999
  - Merge: 69008
  - Quick: 125439
- For Random file as input.
  - Insertion: 25039782
  - Merge: 120388
  - Quick: 117426
- For Reversed file as input.
  - Insertion: 49995000
  - Merge: 64608
  - Quick: 115454

Based on the results, I would choose Merge sort as the best sort. Insertion is only good if the data is already basically sorted. Quick is consistent, but has more compares than merge sort. Merge has about half the compares of quick when processing reversed, and ordered input while having a negligible difference when it comes to random input.