Connor Cowher

CPSC 350

Prof. Rene German

12/19/2020

Sorting Report

Quick Sort - Did not integrate runtime. Quick sort was easy to code, not too many lines, saving runtime.

Merge Sort - Did not integrate runtime. Inefficient when we talk about RAM. Merge makes temporary arrays which takes up the computer's RAM, inefficient.

Selection Sort - Did not integrate runtime. Selection finds the smallest index and puts it in its correct order. More efficient than bubble (i.e. less swapping) but doesn't mean it's fast.

Insertion Sort - Did not integrate runtime. Less swaps, better performance. I believe insertion would be the best fit for the majority of simple text input files.

Bubble Sort - Did not integrate runtime. Works (theoretically) by swapping positions with indexes next to each other. Performance is relatively quick but lots of swaps between elements when it could be inefficient.

Overview - I chose to use C++ to code because it has been fresh in my mind for this semester. I really enjoy coding in java, where I began, but I have had so many computer issues with Eclipse and other applications. Some tradeoffs to consider when using a sporting method is to value time or efficiency. There is usually only one and not the other. My program did not compile and I could not see all the algorithms in their glory. Time stamping was also not integrated into assignment 6. If I attempted this project in java, the results may have changed. I am willing to bet I would have procrastinated the same way I did with all the C++ assignments. Each algorithm has its perks and drawbacks. I enjoyed struggling and learning through failure. Unfortunately I did fail a lot during this project/semester and did not grow and reflect on these times.