Assignment 6 Write Up

Ethan Maniss

December 2019

The four sorting algorithms that I used in this assignment were Bubble Sort, Selection Sort, Insertion Sort, and Merge Sort. The file of doubles that I used to test my code had 39,600 numbers. Bubble Sort took the longest to sort the file at 4 seconds. Selection Sort took 2 seconds. Insertion Sort took 1 second. Merge Sort was output at "0" seconds, however I was not able to convert this to milliseconds. The time differences were more drastic than I expected. I knew that Merge Sort would be the quickest by far, but I did not expect the run time to cut nearly in half from Bubble Sort to Selection Sort. The Big-Oh run time of both of these algorithms is quadratic asymptotically, but when using empirical analysis the difference can be more clearly seen. The primary trade-off of choosing one algorithm over another is run time vs. ease of implementation. Bubble Sort, Selection Sort, and Insertion Sort are all relatively easy to implement, however the run time is much slower than Merge Sort. On the other hand, Merge Sort is much more complex to implement but has a run time of logarithmic rather than quadratic. Using C++ for this assignment rather than Java or Python meant that the run time was going to be quicker, since compiled code is transferred straight into machine code, and there is explicit memory management. When I ran my program, the CPU spiked to 10.7 percent of my computer's capacity. Lots of CPU space is used during the program run time because array size was determined and algorithms were all executed during the run time. The shortcomings of this empirical analysis were that all the algorithms needed to be performed in the same environment, and be fully implemented and executed. Also, certain factors such as if the file already had some partially sorted numbers, this would give Insertion Sort a major advantage over Bubble Sort and Selection Sort.