In the final assignment of the year, we were asked to implement and analyze different sorting algorithms to see which ones the most efficient and which ones were lackluster. The six algorithms that were used were: bubble sort, selection sort, insertion sort, quick sort, merge sort, and an algorithm of our choice, so for me, shell sort. By creating large files with tens of thousands of numbers, we were able to see which algorithms took longer than others to sort, and which ones took up more CPU and memory usage by looking at the docker desktop statistics. For my specific case, I found a file of 100,000 numbers to give us good results to be able to compare with one another. The slowest algorithm to no surprise was bubble sort. In a file of 100,000 randomly sorted numbers, bubble sort would usually take about 41 or 42 seconds to sort all the numbers. This is due to the algorithm's slow nature of moving from number to number and comparing two numbers each iteration. The next slowest algorithm that I observed was selection sort taking roughly 17 seconds to complete. Next was insertion sort, with an average runtime of 10 seconds. The next two algorithms, shell sort and merge sort, had pretty similar runtimes with shell sort running at 0.061 seconds and merge sort running at 0.055 seconds. The fastest algorithm that was observed was quicksort, with an average runtime of 0.04 seconds. When running bubble sort, the CPU usage averaged above 100%, and there was a 1.4 MB increase in the memory usage during runtime. Insertion sort also averaged above 100% CPU usage and there was a 1.5 MB increase in the memory usage. For selection sort, the CPU averaged around 100% usage, but there was only a 1.2 MB increase in the memory usage. When using shell sort, the CPU only used about 10% of its usage, and the memory usage only increased by 0.1 MB. There were very similar stats recorded when using merge sort, as the CPU usage only went to about 10% on the memory usage barely increased during runtime. Finally, quick sort also increased the memory usage by only 0.1 MB, but its CPU usage averaged around

8%. I would say quicksort is the best algorithm based off my analysis because it has the quickest runtime, takes the least amount of CPU usage, and is comparable to other algorithms in terms of memory usage.