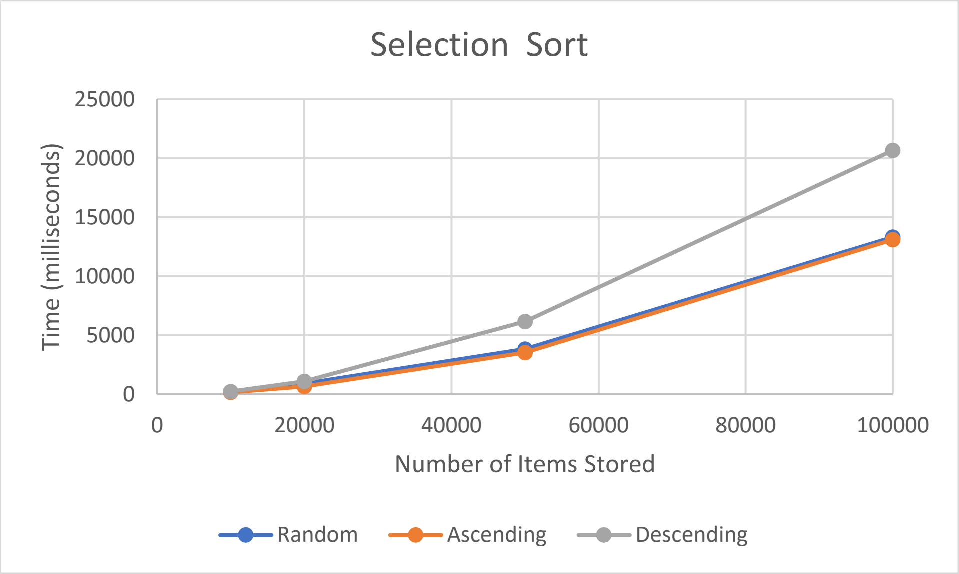
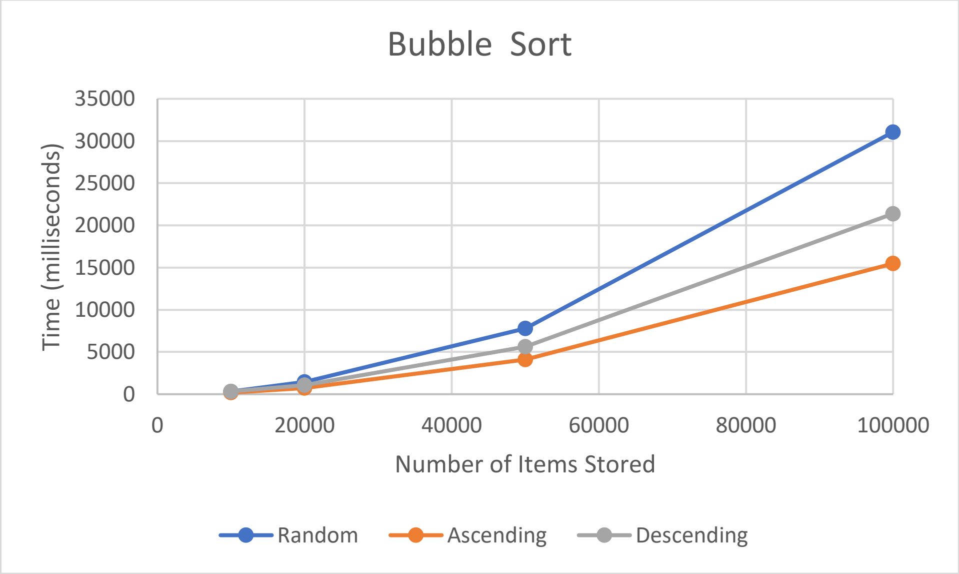
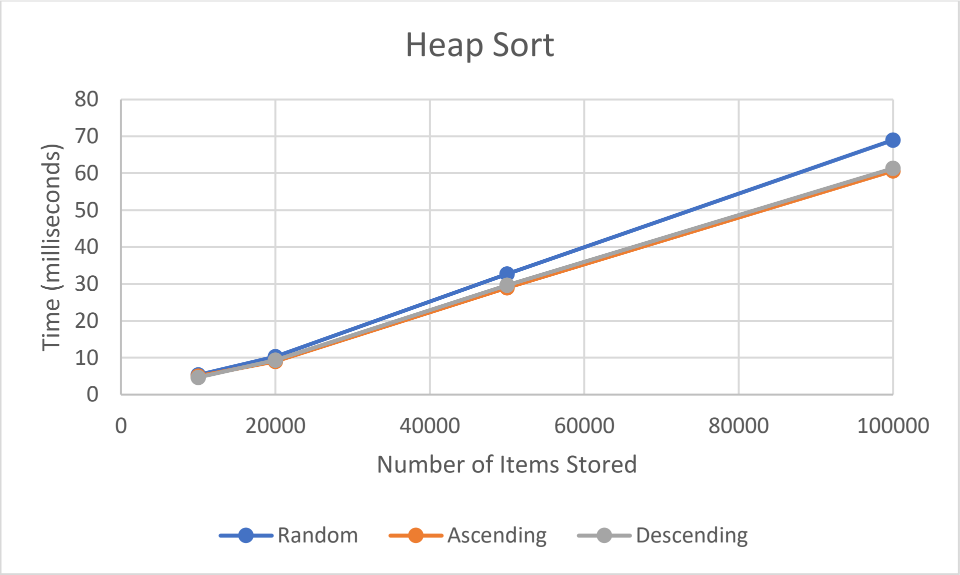
Sorting Analysis Report

For this sorting analysis project, I chose to implement a selection sort, bubble sort and heap sort. Each sort was tested at least three times before the average execution time of the sort was used with the given file. This means that for each of the 12 different files used, I tested the program three times before using the average as the final time on this report.

Below are the x-y plots for each sort's run time versus file size:







The run times of the different sorts are evident through the scale of the y-axis. Selection sort and bubble sort are relatively similar in how fast it takes for the algorithms to run. Heap sort was much faster in its performance, being able to complete the algorithm with each given file in less than 100 milliseconds while the other sorts exceeded 10,000 milliseconds. It is clear that the size of a file seems to have the most impact on the time required for each sort to complete. This trend is demonstrated in the increasing slope of each plot line for every sort.

The plots show that, on average, selection sort and bubble sort are O(n^2). It also indicates that heap sort is O(nlog(n)). Selection sort's graph increases in a quadratic fashion, which coincides with its complexity. A quadratic function increases in a gradual way before its slope starts becoming much steeper. This is also the case for bubble sort, where a clear parabolic curve starts to appear in its plot. Heap sort, on the other hand, is much more gradual throughout. It might even seem slightly linear if you do not take into account the beginning of the plot. This slower growth corresponds with the log linear graph. As the slope does not seem nearly as steep, the growth must be slower than a quadratic one. It also does not grow with a constant slope, which means that its faster than a linear function. Thus, the heap sort's graph matches its time complexity.

Observing the different cases for each sorting algorithm, there is a general trend of ascending order being the fastest to pass through the sorts. Interestingly, while descending order was the worst case for selection sort, it ended up as more of an average case for the bubble and heap sorts. Random, while typically an average case as seen in selection sort, ended up as the worst case for bubble and heap sorts. This might have something to do with the processor optimizing the descending order files after running through the selection sort algorithm multiple times. It’s also possible that a clear pattern may be easier for bubble and heap sorts to parse through.