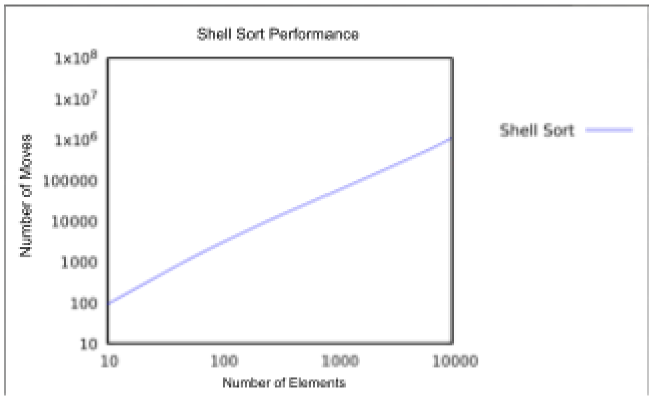
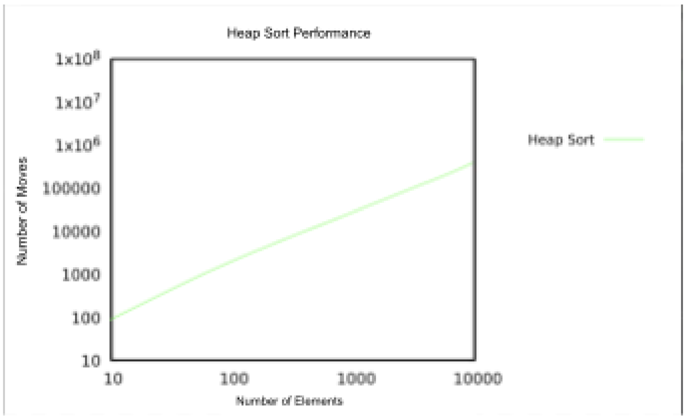
For this assignment we were tasked to recreate 4 sorts using python pseudocode. These sorts all have different time complexity and performance. Below, we can see the performance graph which has the number of elements sorted vs the number of moves taken to perform this sort.

I learned that these sorts all perform well when they are within a data set range of between 10-10000, upon going beyond thai point the graph tends to show signs of more moves to calculate and a platowing behavior in the curve of the graph.

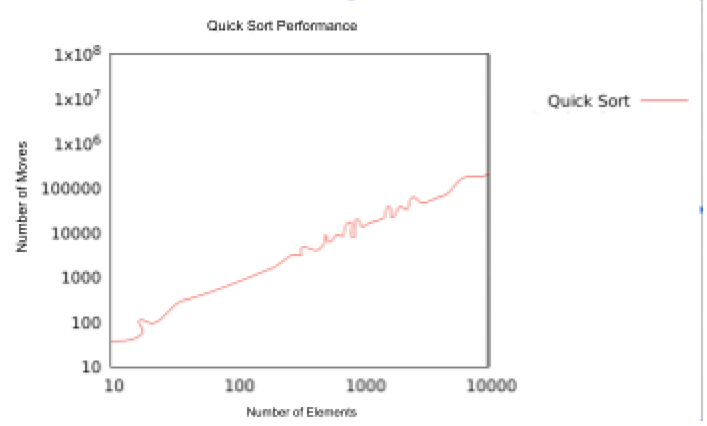
I was also able to understand how the relations between different sorts work and how by incorporating one sort into another will increase efficiency.



Shell sort is the slowest and least efficient of our sport. Because it has the highest number of moves taken to reach 10000 iterations. This can be seen in the graph on the right.



This graph demonstrates the performance of heap sort. Heap sort is a good sorting algorithm but unpredictable in terms if time complexity with a min max of O(nlogn).



This is a graph of the quick sort, which is the fastest sorting algorithm. Quick sort is very fast as it takes less number of moves to iterate through a large number of elements. This part is oftentimes used as a subsection of other sorting algorithms to improve efficiency. Compared to the other graphs, this graph has a much lower point of chant on the y axis.