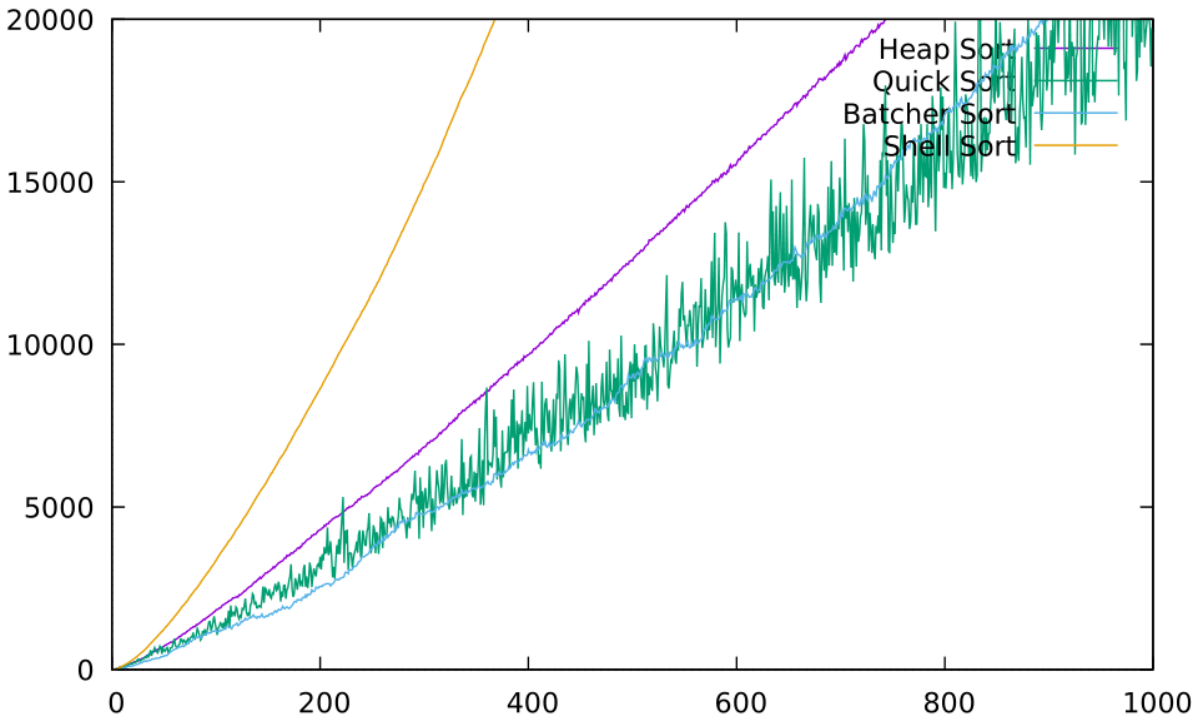
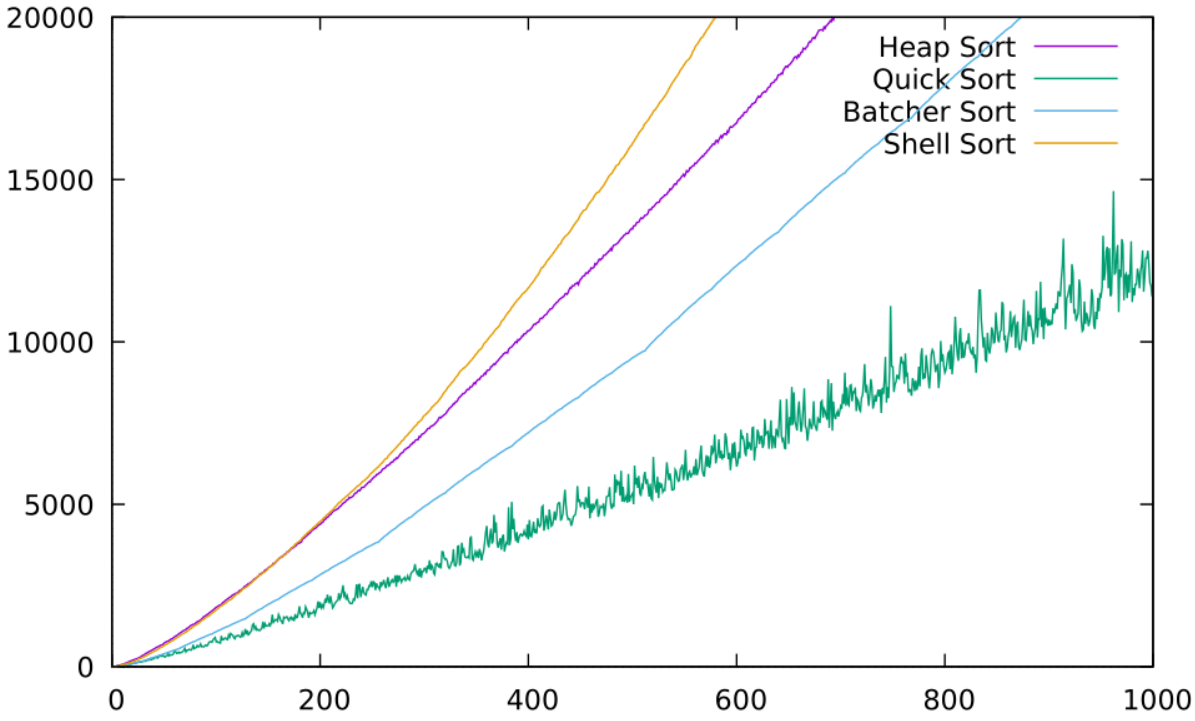
I learned that sorting algorithms work better with arrays that have less elements to sort and ones that are almost sorted already. The first graph below shows the number of elements compared to the amount of moves done across all 4 algorithms, and the second shows the number of elements compared to the amount of comparisons. Quick and Batcher sort performed the best under the pressure of more elements, with heap and shell doing the worst. Quick sort had a very sporadic line, probably because out of these 4 algorithms it's the one who's sort time is most dependent on how unsorted the array was before sorting. Since I believe swapping is more memory intensive and comparisons are more CPU intensive, you could use these graphs to determine what's better to use for your specific use case and hardware target. Generally it seems like Batcher and Quick sorts are the best for most cases.




20000
15000
10000
5000
0
0
200
400
600
800
1000
Heap Sort
Quick Sort
Batcher Sort
Shell Sort