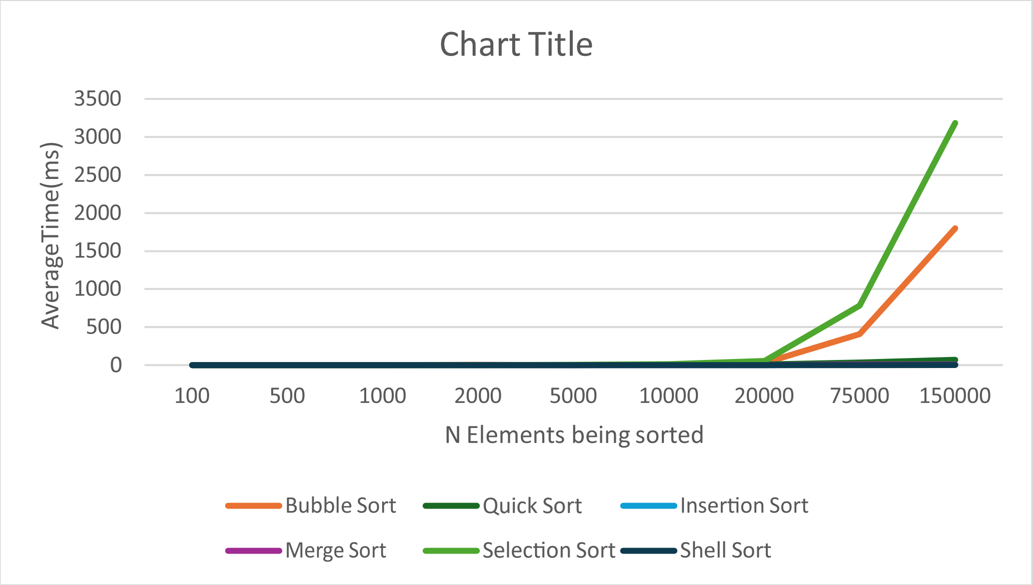
N elements we are sorting	Bubble Sort	Quick Sort	Insertion Sort	Merge Sort	Selection Sort	Shell Sort
100	0.1083	0.0323	0.0165	0.0611	0.0751	0.0215
500	1.5056	0.4806	0.1374	0.289	1.0958	0.1393
1000	0.7932	0.1462	0.4259	0.1165	0.3876	0.3172
2000	2.3736	0.9018	0.6552	0.2965	0.5863	0.6081
5000	1.8827	0.7007	0.483	0.5185	3.5774	0.986
10000	7.3271	2.8706	0.3925	0.7204	14.1113	0.2273
20000	29.6059	7.5277	0.4446	1.6045	55.849	0.3839
75000	409.2893	32.9203	2.6702	7.0316	782.325	0.971
150000	1799.644	69.3245	6.4897	7.2227	3184.107	1.8295



After doing the KSorted data I noticed that everything performed pretty similarly to my first ranking for the random data but did have some differences due to some sorting algorithms performing way better with more elements given. For example, Shell sort performed the best with the 150,000 values given, but in the random data it was one of the worst. Some of these algorithms such as shell sort performed better with more elements, because they were partially sorted and algorithms such as shell tend to do better with partially sorted arrays. Also, just as in problem 9 it was hard to fully represent the graph due to having values that vary so far from each other, which is why it is hard to graph numbers that are so big.