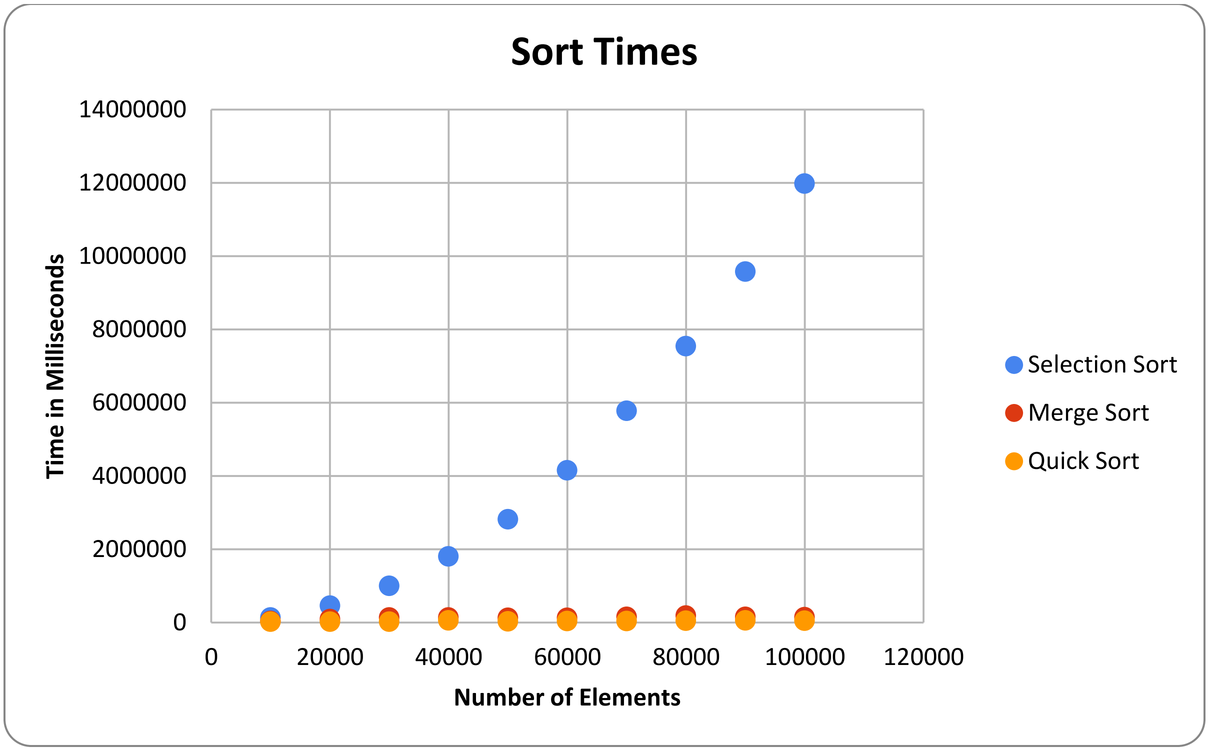
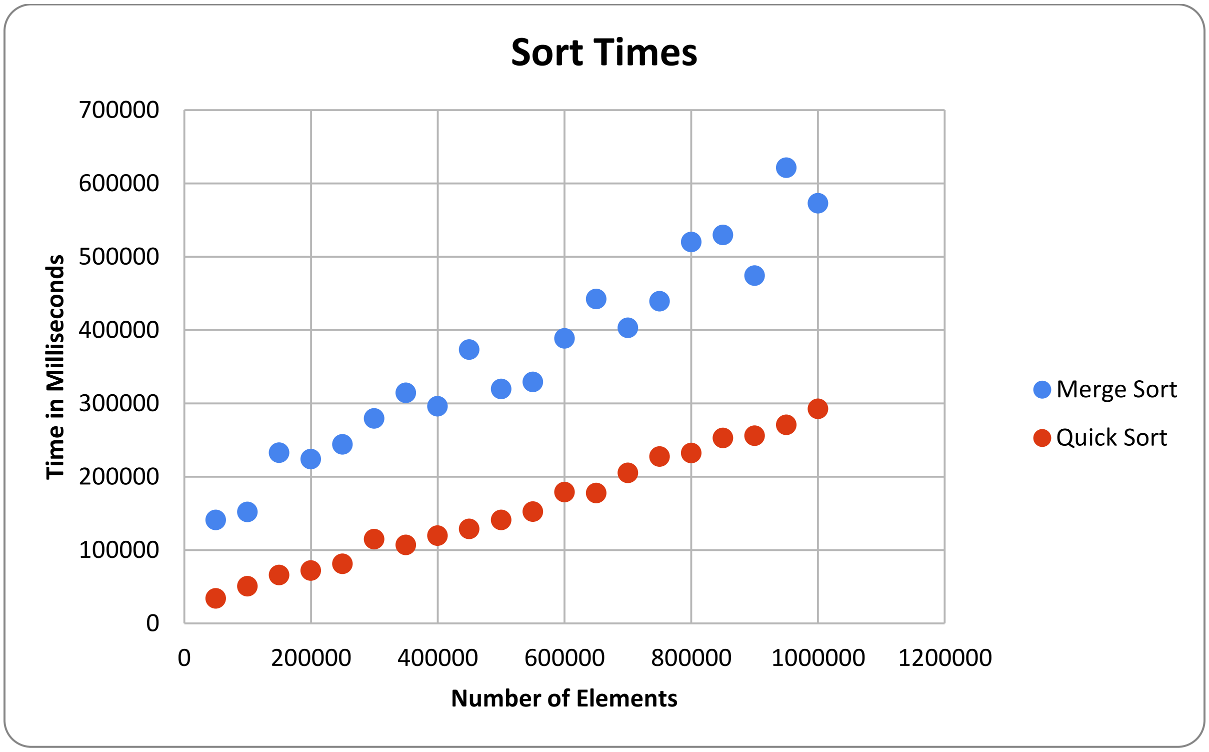
George Miller
CS 102 – Joanna Klukowska
Programming Project 2 - Report

The results of my project didn't come out quite as I expected, for a few reasons. I thought that my merge sort timing would have been closer to my quick sort timing because they both have the same big O performance, but however, merge sort was quite a bit slower on every occasion. I think this is because of my implementation of merge sort. When the subarray gets down to one element, I create an array of one element and return it because the method has to always return an array. I tried to fix this, by changing around the parameters of my merge method and the return of my merge sort, but I couldn't come up with a way that made the method work without bugs. If there was a way to not force the computer to create a whole array for just one element, that would probably make the whole process a lot faster. I also found that my merge sort method had a much larger range of process time than my quick sort. My quick sort's time was fairly linear while my merge sort's time jumped around a lot and had a much larger range of values. This is odd, considering how one would think that quick sort would be the one with more variation, because it would depend on whether the computer picked a better pivot. Since I generate pivots randomly, this even more supported my thought that it would sway in time but the opposite is true. It was also very interesting to see how the performance of quick sort changed as I changed the way I chose pivots, as in some cases it made quicksort slower than merge sort. Overall, I'd say this whole project was a great way to see how little improvements in an algorithm can drastically change their efficiency. It definitely changed the way I code because it reminded me that every line matters and if you have any that aren't needed; they could take up an unacceptable amount of power and resources.

Num. of Elements	Selection Sort	Merge Sort	Quick Sort
10000	135729	42423	25589
20000	461054	92287	21260
30000	1004281	135081	27244
40000	1806921	138166	56580
50000	2822917	125537	28863
60000	4155049	127205	44155
70000	5779083	156429	40406
80000	7546731	190323	46475
90000	9577472	159332	55002
100000	11978465	144972	54406

Num. of Elements	Merge Sort	Quick Sort
50000	140893	34056
100000	151971	50603
150000	232505	65905
200000	223945	71917
250000	243991	81192
300000	279521	114562
350000	314151	106728
400000	295981	119494
450000	373378	128933
500000	319646	140854
550000	329031	152292
600000	388373	178759
650000	442454	177671
700000	402871	205185
750000	439246	227228
800000	520101	232334
850000	529810	252845
900000	474077	255647
950000	621142	270589
1000000	572738	292328


Sort Times
Time in Milliseconds
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
0
20000
40000
60000
80000
100000
120000
Number of Elements
● Selection Sort
● Merge Sort
● Quick Sort


Sort Times
Time in Milliseconds
700000
600000
500000
400000
300000
200000
100000
0
0
200000
400000
600000
800000
1000000
1200000
Number of Elements
● Merge Sort
● Quick Sort