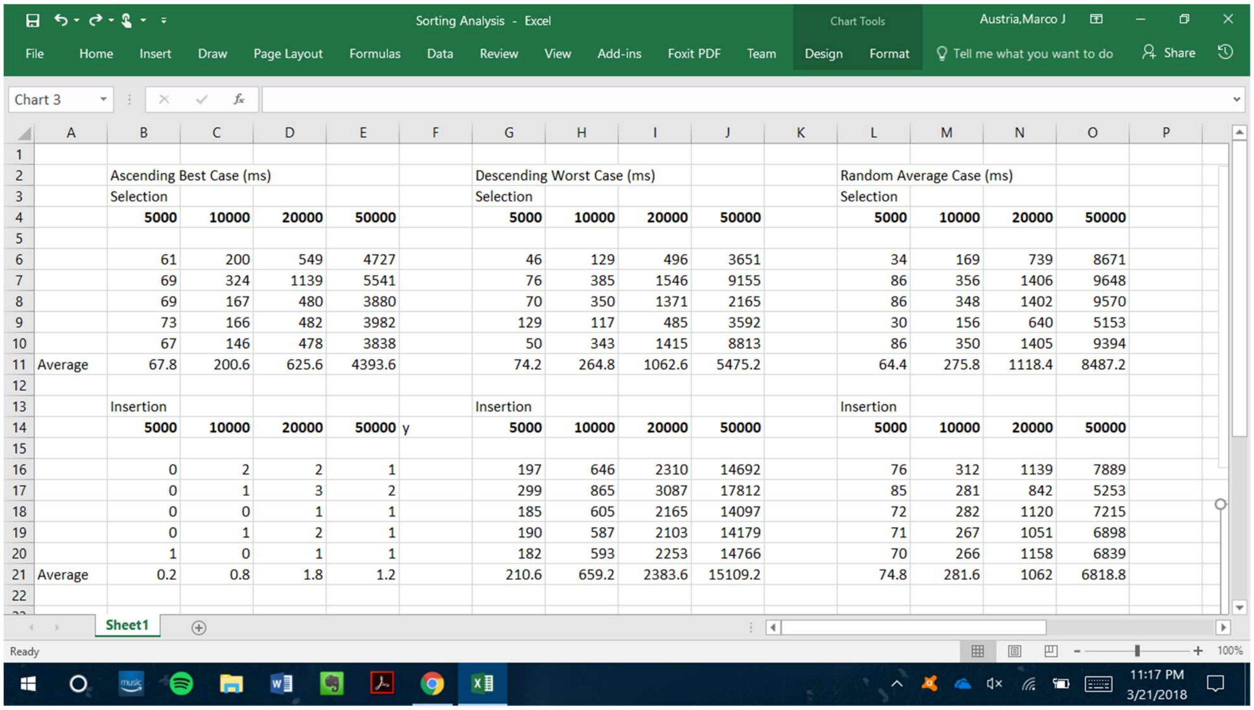
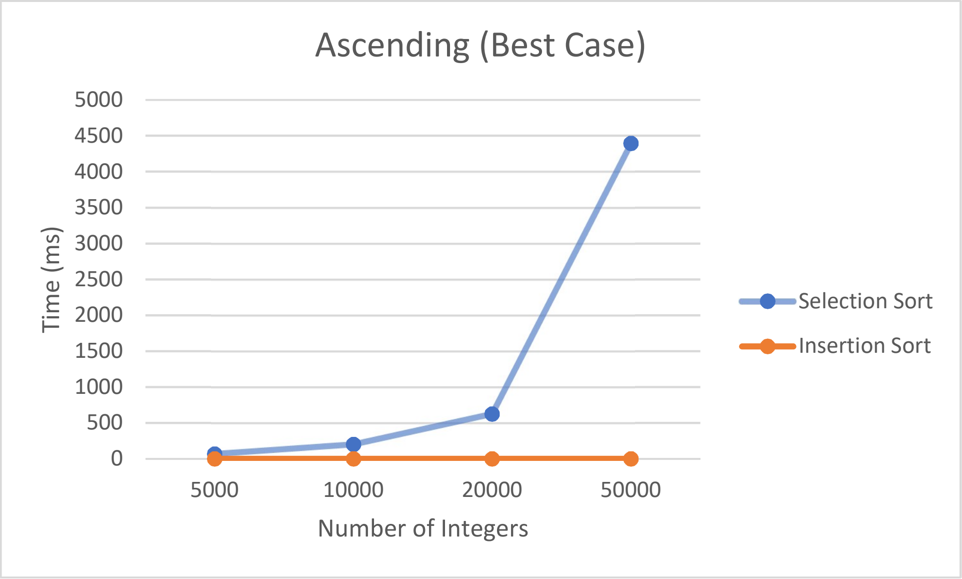
COP3530

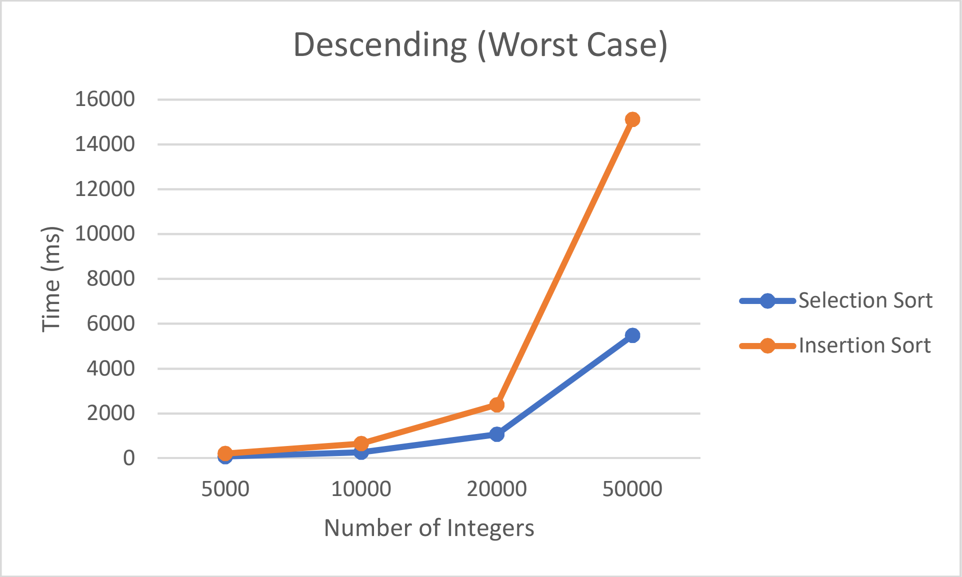
Marco Austria

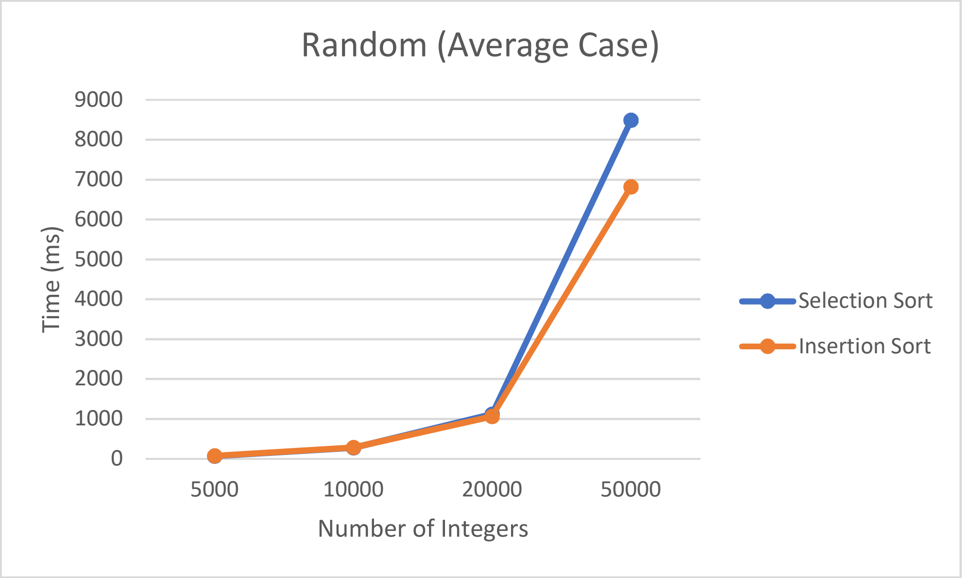
Sorting Analysis Project



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2		Ascending Best Case (ms)					Descending Worst Case (ms)					Random Average Case (ms)				
3		Selection					Selection					Selection				
4		**5000**	**10000**	**20000**	**50000**		**5000**	**10000**	**20000**	**50000**		**5000**	**10000**	**20000**	**50000**	
5																
6		61	200	549	4727		46	129	496	3651		34	169	739	8671	
7		69	324	1139	5541		76	385	1546	9155		86	356	1406	9648	
8		69	167	480	3880		70	350	1371	2165		86	348	1402	9570	
9		73	166	482	3982		129	117	485	3592		30	156	640	5153	
10		67	146	478	3838		50	343	1415	8813		86	350	1405	9394	
11	Average	67.8	200.6	625.6	4393.6		74.2	264.8	1062.6	5475.2		64.4	275.8	1118.4	8487.2	
12																
13		Insertion					Insertion					Insertion				
14		**5000**	**10000**	**20000**	**50000**	y	**5000**	**10000**	**20000**	**50000**		**5000**	**10000**	**20000**	**50000**	
15																
16		0	2	2	1		197	646	2310	14692		76	312	1139	7889	
17		0	1	3	2		299	865	3087	17812		85	281	842	5253	
18		0	0	1	1		185	605	2165	14097		72	282	1120	7215	
19		0	1	2	1		190	587	2103	14179		71	267	1051	6898	
20		1	0	1	1		182	593	2253	14766		70	266	1158	6839	
21	Average	0.2	0.8	1.8	1.2		210.6	659.2	2383.6	15109.2		74.8	281.6	1062	6818.8	
22																




Descending (Worst Case)
16000
14000
12000
10000
8000
6000
4000
2000
0
Time (ms)
5000
10000
20000
50000
Number of Integers
Selection Sort
Insertion Sort


Random (Average Case)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Time (ms)
5000
10000
20000
50000
Number of Integers
Selection Sort
Insertion Sort

After evaluating the sorts, we see that the insertion sort is very good in the ascending best case scenario, worse than selection sort in the descending worst case scenario, and almost the same as selection sort in a random average case scenario. In general, as the number of integers increases, the time it takes to organize becomes significant with both sorts. This displays how inefficient these sorts are, unless the list size is relatively small, or the list is almost sorted and near a best case scenario with the insertion sort. In the ascending best case, the insertion sort is O(n), as it only has to visit every integer in the unsorted side to see that they're all in order with no swaps being necessary, and therefore no search through the sorted side needed. Otherwise, in all other cases, both sorts are O(n^2), requiring 2 loops(one nested in the other).