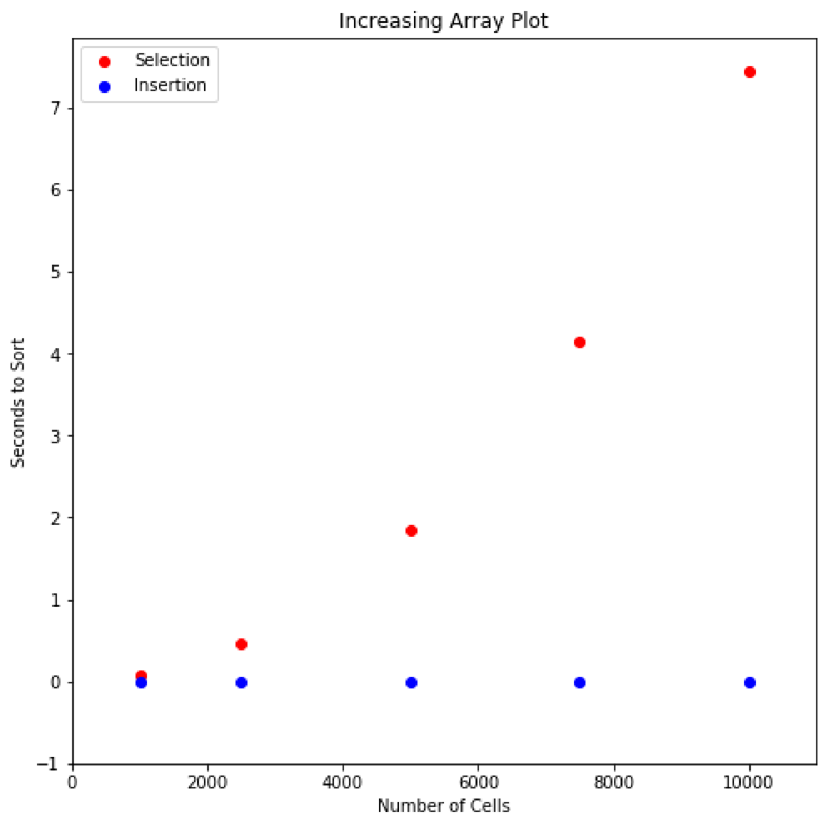
Project 1: A Battle of Sorts

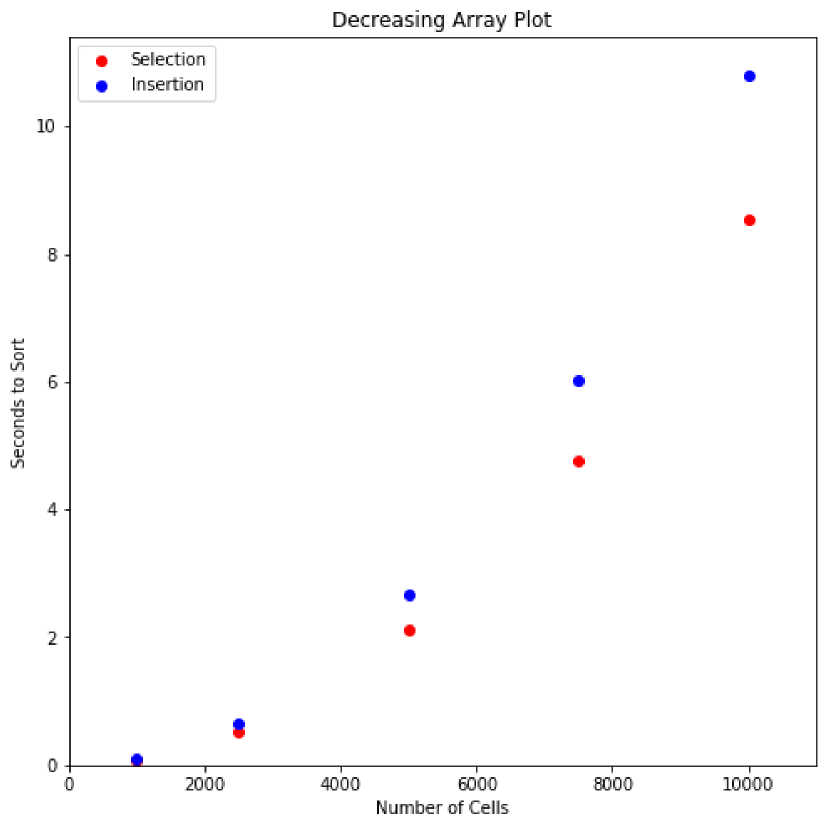
In this project, we compared the time it takes for insertion sort and selection sort to sort various arrays into ascending order. These arrays have cell counts that range from 1000 to 10000 cells, and are in ascending order, descending order, or a random order. Insertion sort looks at each cell one at a time and performs any necessary swaps so that its value is correctly sorted for all of the cells that have already been seen. A swap is essentially switching the values of two cells. This means that insertion sort only checks each cell in the array once, but it does not have a fixed number of swaps. Selection sort goes through the entire array and finds the smallest value, then performs only one swap to make it first. It repeats this for every value, finding the next smallest value and swapping it to second, and then third, until the array is ordered correctly. This means that regardless of the order, it must search through the entire part of the array that has not yet been checked n-1 times, with n being the length of the array. It is n-1 and not n, as the remaining value at the end must already be the largest value. In addition, searching accounts for most of the time that selection sort takes, with swaps making only a minor difference.

All of the combinations have timings that grow at increasing rates, except for increasing insertion. This is because with the other five combinations (increasing, decreasing, and random selection and decreasing and random insertion), for every cell added to the length, on average, there are more steps added. For example, adding one more cell to an array of length 10000 and then using selection sort would not have one more step, but instead, it would have 9999 more steps, as it would need to search that cell once for every value in the list, except for the last value. The reason why increasing insertion grows at a fixed rate is because, regardless of how many cells long an array is, adding another cell will only require insertion sort to perform one more step, as it is only looking through each cell once and not making any swaps.

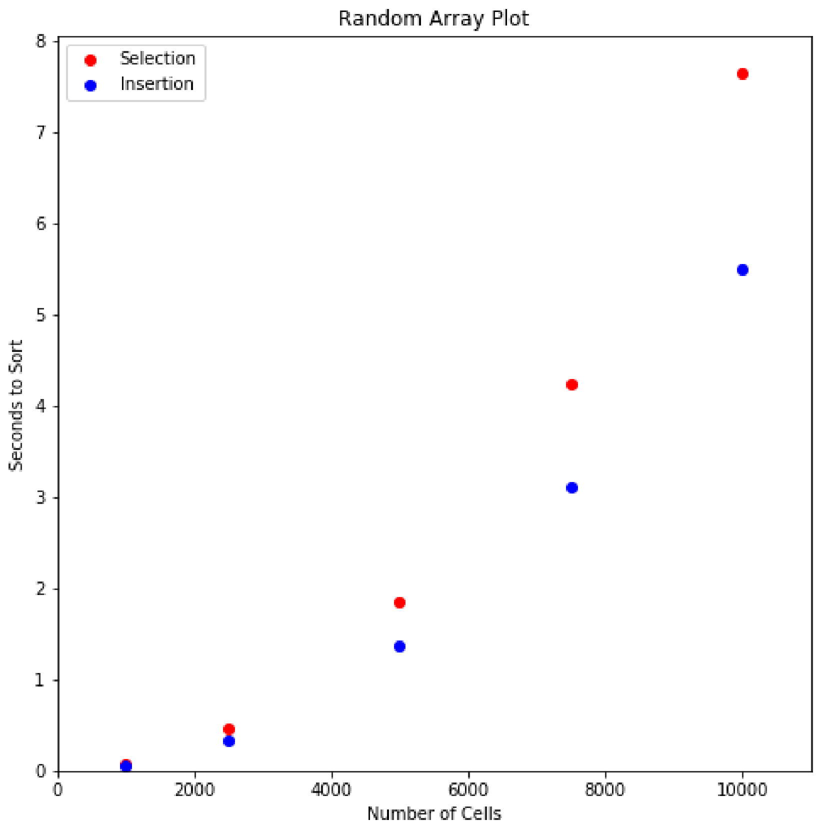
For the reason listed above, increasing insertion sort performs much faster, in fact, hundreds or even thousands of times faster, than increasing selection sort. Neither sort needs to perform any swaps, but insertion sort only performs the check on each cell once. On the other hand, selection sort still has to search through the array n-1 times. Each search also requires more steps the longer the array is, so as n grows, it has an increasing number of steps added. This difference can be seen very clearly in the graph below.



With decreasing arrays, selection sort performs slightly faster. Now, insertion sort must perform the maximum number of swaps, as each value must be swapped from its cell to the first cell. This is because each value that insertion sort sees is the smallest value it has seen. Selection sort, despite needing to search through the array n-1 times, it only has to perform n-1 swaps, which is what accounts for it taking only slightly longer than increasing selection sort. Insertion sort must perform one more swap for every cell it sees, finally reaching n-1 swaps in the last step. This means that, although selection sort had to search through the array many more times, insertion sort had to swap many more values.



With random arrays, insertion sort seems to barely outperform selection sort. Again, selection sort takes almost the same amount of time as before, as it needs to search through one time for every cell in the list. With insertion sort, since the order is random, there will be a lot of variation in the number of swaps needed for each iteration. Although selection sort will require fewer swaps, random order means that insertion sort only performs, on average, about half the number of swaps as with a decreasing array. As seen in the graph, random insertion sort also takes about half as long as with decreasing insertion sort.



As we can see, the two sorting methods function at various speeds, depending on the initial array. Insertion sort performs much faster with arrays in ascending order, and slightly faster with arrays in random order. Selection sort performs slightly faster in descending order, but it also has much more precision with its timings. The fastest sort is increasing insertion sort with timings in the fractions of seconds, while the slowest is decreasing insertion sort with timings above 10 seconds.

Data for Project 1: A Battle of Sorts

	sort	order	cells	run one	run two	run three	run four	run five
0	insertion	decreasing	1000	0.09171431600407230	0.09426971600623800	0.09811315400293100	0.0996122619981179	0.09536059400124940
1	insertion	decreasing	2500	0.5835944460122850	0.6580775829934280	0.6783091429970230	0.6551586050045440	0.6743938319996230
2	insertion	decreasing	5000	2.4561263780051400	2.6716969340050100	2.730816223003790	2.712936907992120	2.7113146259944200
3	insertion	decreasing	7500	5.707030924997530	6.06240650700056	6.088170306000390	6.075943998002910	6.170243002998180
4	insertion	decreasing	10000	10.182914517005000	10.857592713000500	10.988559623991000	10.961606985001700	10.959624540002600
5	insertion	increasing	1000	0.00017546799790579800	0.0001734079996822400	0.00022871399414725600	0.00018810298934113200	0.0001910739956656470
6	insertion	increasing	2500	0.00044419300684239700	0.00046168299741111700	0.0004458939947653560	0.0004688179906224830	0.00047635599912609900
7	insertion	increasing	5000	0.0008634950063424190	0.0009009430068545040	0.000898561003850773	0.0009405710006831210	0.0009142890048678960
8	insertion	increasing	7500	0.0013614549970952800	0.001353636005660520	0.0014381679939106100	0.0014116600068518900	0.001475762008340100
9	insertion	increasing	10000	0.0018469509959686500	0.0018314580083824700	0.0017965519946301400	0.0018666950054466700	0.0018934100080514300
10	insertion	random	1000	0.049154238004121	0.04555000600521450	0.04851274099200960	0.050613358995178700	0.04925665799237320
11	insertion	random	2500	0.3173830229934540	0.32870087499031800	0.3276577219949100	0.33496226600254900	0.3368407579982890
12	insertion	random	5000	1.2974811110034400	1.360754822002490	1.3819178550038500	1.3891660749941400	1.439609367007510
13	insertion	random	7500	2.954713469996930	3.1177904270007300	3.1706770830060100	3.162384534996820	3.1695467000099600
14	insertion	random	10000	5.325015003007140	5.505544769999690	5.591303232009520	5.463934977989990	5.598276378004810
15	selection	decreasing	1000	0.07959830899199010	0.0793725550029194	0.08238744399568530	0.08854623000661380	0.09168372499698310
16	selection	decreasing	2500	0.46858364398940500	0.5335914340103050	0.5415026209957430	0.557681873004185	0.5550286320067240
17	selection	decreasing	5000	1.8931689759920100	2.1770495300006600	2.146401523001260	2.1977272239892000	2.1759126400138500
18	selection	decreasing	7500	4.235689060995360	4.9306597630056800	4.911870209994960	4.893861327000200	4.903353171990600
19	selection	decreasing	10000	7.679463369000590	8.546430645001240	8.76003628699982	8.723082711992900	9.002501833005230
20	selection	increasing	1000	0.0663988260057522	0.0741962309984956	0.08171988799585960	0.07505742298963010	0.07357683600275780
21	selection	increasing	2500	0.41119058900221700	0.45654103500419300	0.4795845750049920	0.46906715699879000	0.4678407550090920
22	selection	increasing	5000	1.7071876300033200	1.8059823609946800	1.9133602990041300	1.89292462899175	1.912955271996910
23	selection	increasing	7500	3.7125594050012300	4.151349972002210	4.258506836995370	4.338515103998360	4.3058951669954700
24	selection	increasing	10000	6.614614413003440	7.459770196001050	7.627904488996140	7.835653660004030	7.7012544099998200
25	selection	random	1000	0.06569710800249600	0.07875397900352260	0.07925752599840050	0.07567214700975460	0.07594542800507040
26	selection	random	2500	0.41568900700076500	0.4719332230015430	0.47364128900517200	0.47349329201097100	0.4727825500012840
27	selection	random	5000	1.6970956949953700	1.8583811030112000	1.9174356760049700	1.8890375509945400	1.8998322060069800
28	selection	random	7500	3.8283402870001700	4.2906534309877300	4.280932011999540	4.418240054001220	4.367245351997550
29	selection	random	10000	7.0383942350017600	7.642478013003710	7.7241466639970900	7.850252390999230	7.942492324000340