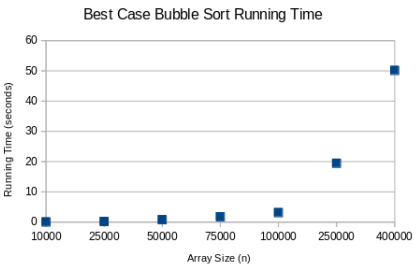
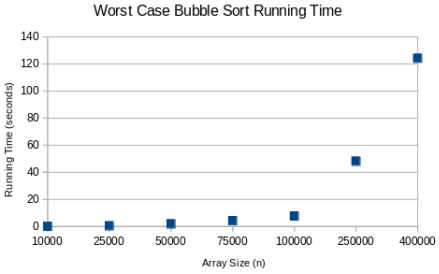
Samuel Dickerson
Dr. Spickler
COSC 320-001
2/20/24

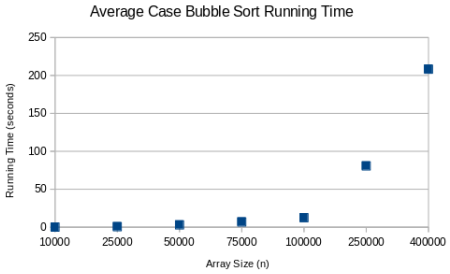
Timing and Counting Analysis of Best Case, Worst Case, and Average Case Scenarios of Four Basic Sorting Algorithms

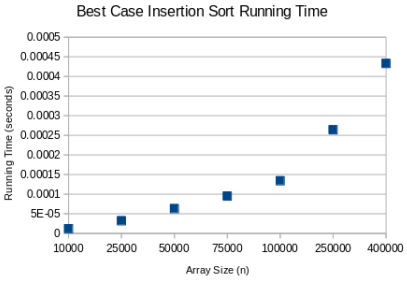
Timing Spreadsheet and Graphs

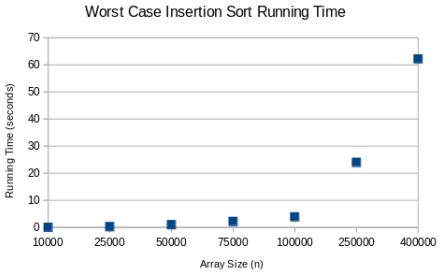
Array Size (n)	BC Bubble (s)	BC Insertion (s)	BC Selection (s)	BC Bubble 2 (s)	WC Bubble (s)	WC Insertion (s)	WC Selection (s)	WC Bubble 2 (s)	AC Bubble (s)	AC Insertion (s)	AC Selection (s)	AC Bubble 2 (s)
10000	0.035848	1.2E-05	0.027758	0.028153	0.077516	0.038175	0.028617	0.129135	0.06811	0.017801	0.027923	0.125964
25000	0.195932	3.2E-05	0.179491	0.189631	0.479799	0.238386	0.179507	0.822238	0.659111	0.10724	0.19757	0.907381
50000	0.79124	6.3E-05	0.711705	0.731068	1.92655	0.956624	0.719215	3.27997	2.98211	0.443534	0.720167	3.84145
75000	1.76917	9.5E-05	1.62773	1.67041	4.36204	2.1694	1.62338	7.4595	6.92572	0.990431	1.66676	8.71668
100000	3.12302	0.000134	2.8055	2.93702	7.7375	3.87915	2.89602	13.226	12.4669	1.763	2.96657	15.6342
250000	19.4535	0.000264	17.5607	18.3468	48.1752	23.9635	18.0752	82.2865	80.7573	10.9872	18.1978	99.2067
400000	50.1582	0.000433	45.6052	47.3819	124.157	62.1994	46.7889	210.96	208.324	28.0751	47.5655	255.536

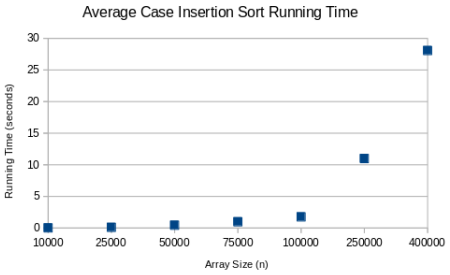


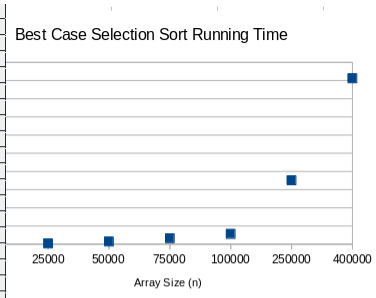


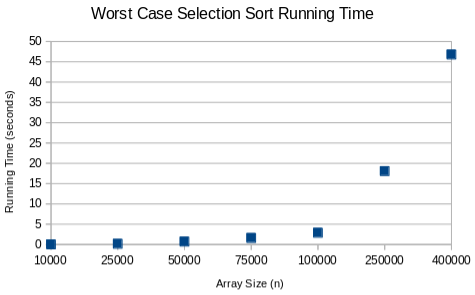


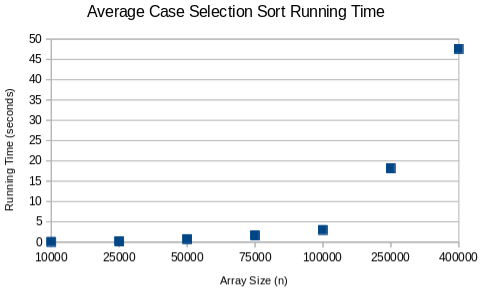


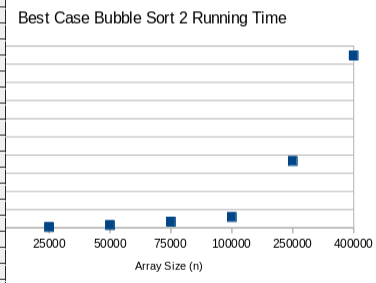


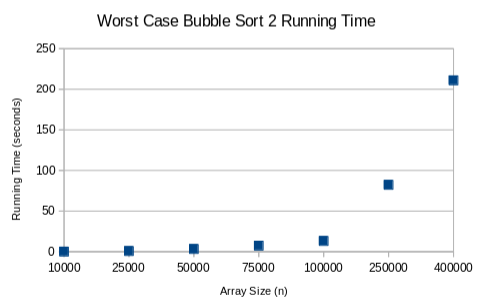


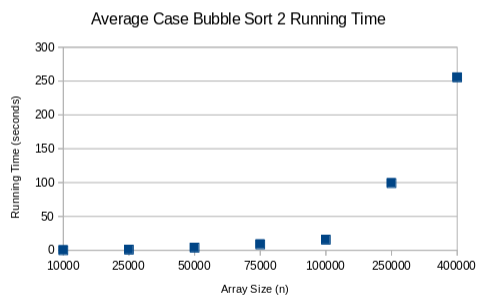






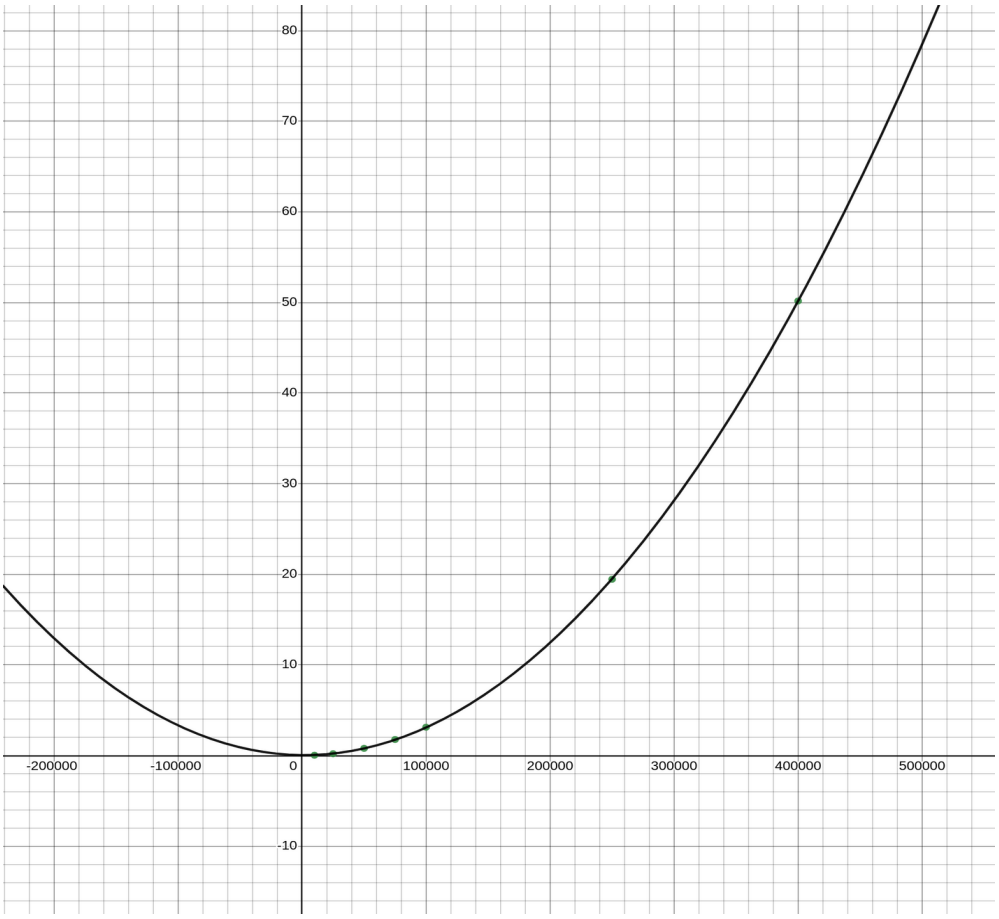




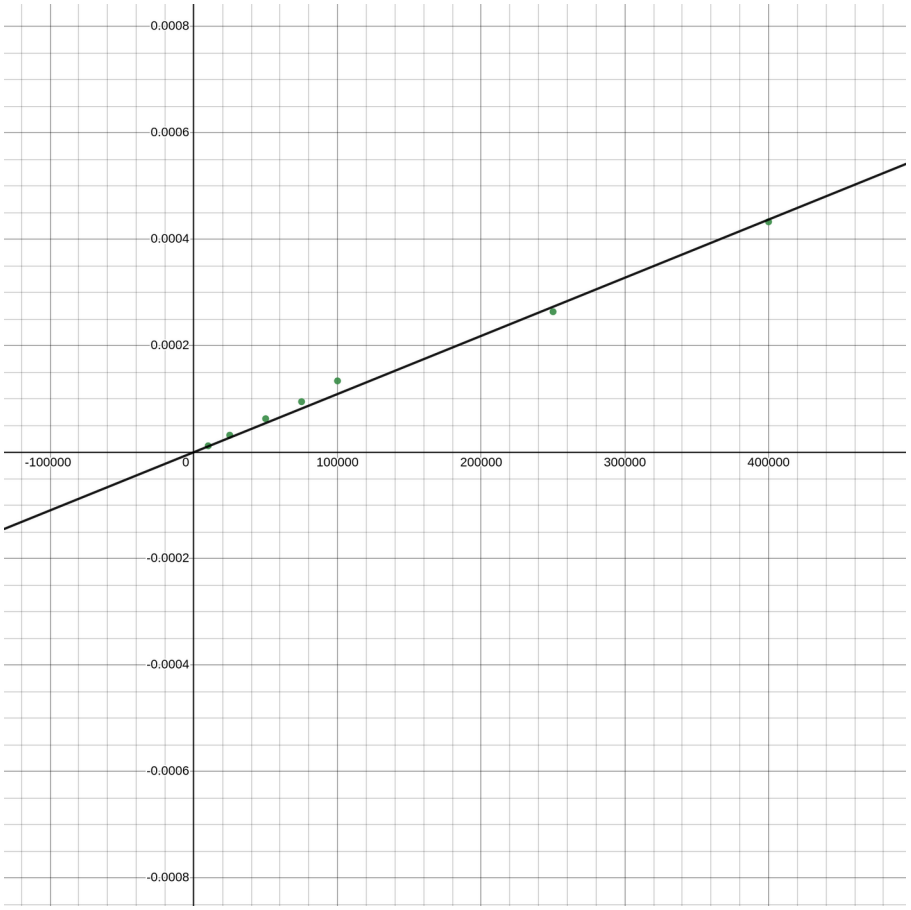


Curve Fitting Graphs for Timing

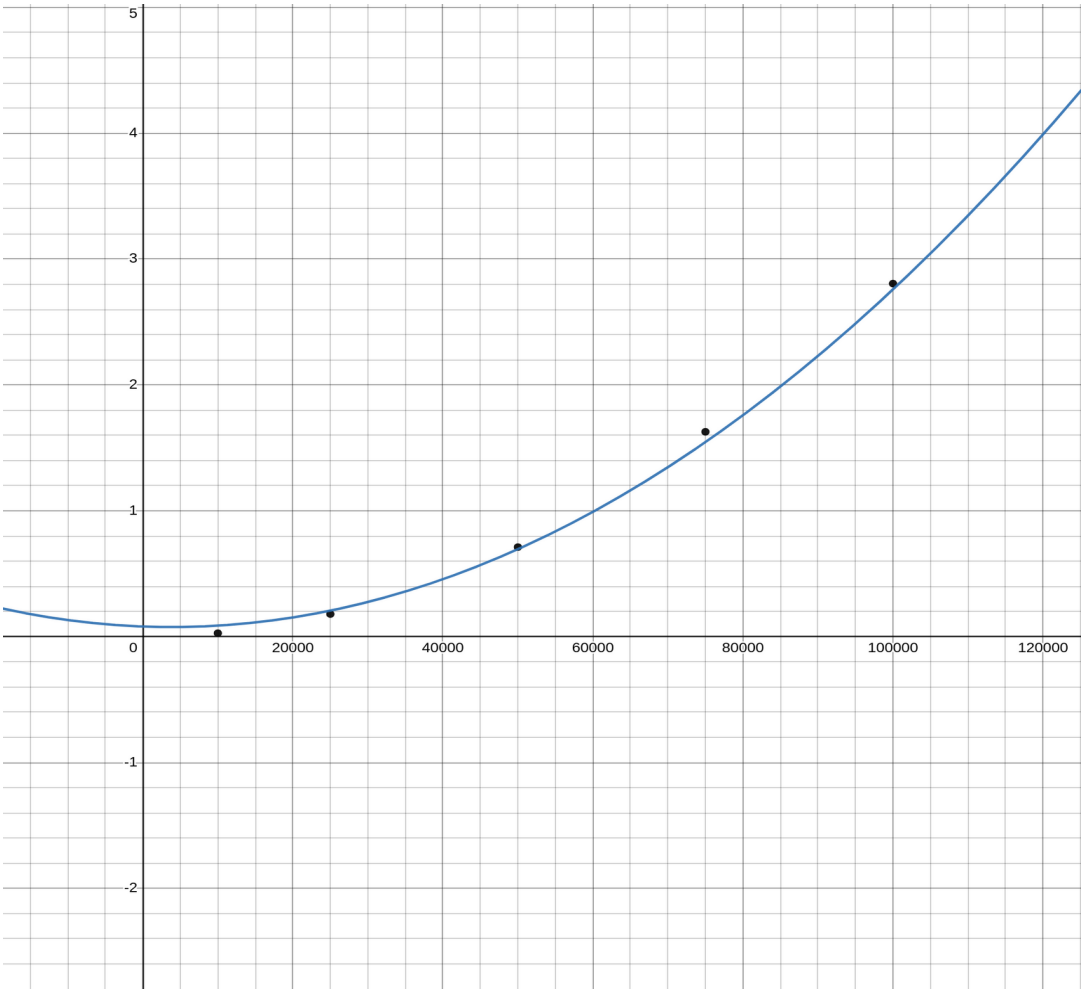
Best Case Bubble Sort:



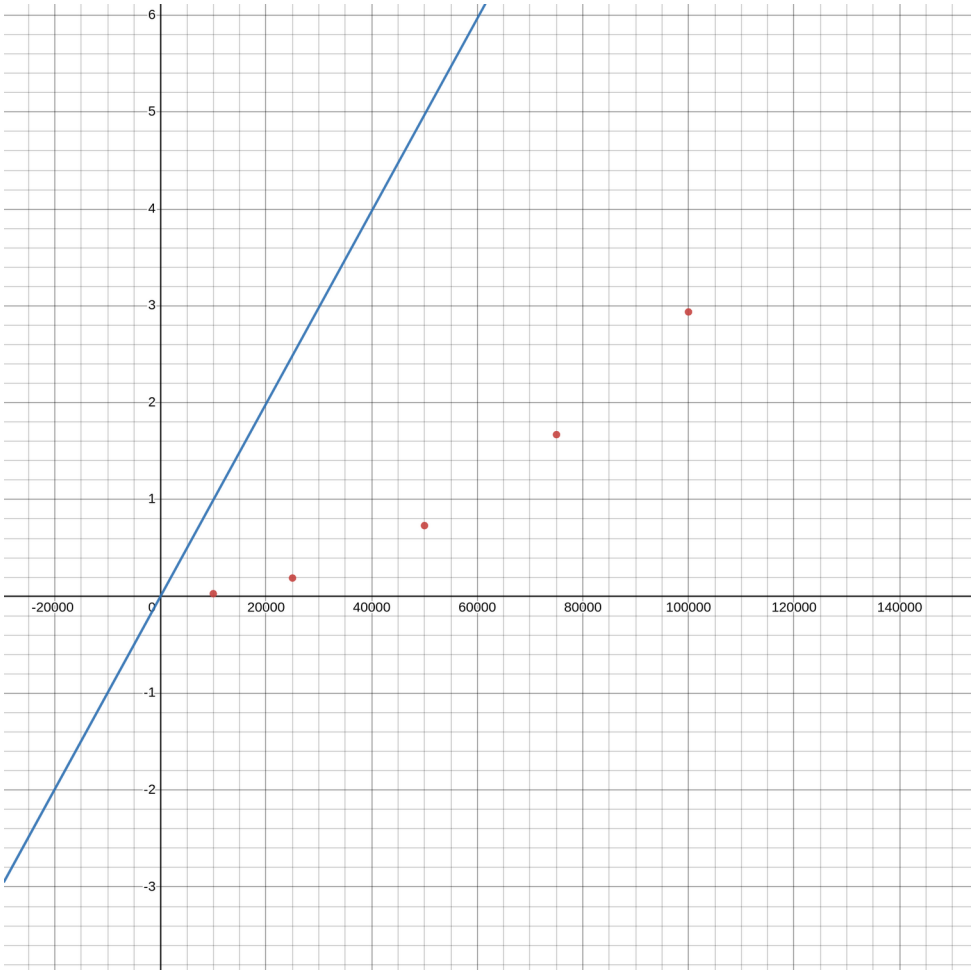
Best Case Insertion Sort:



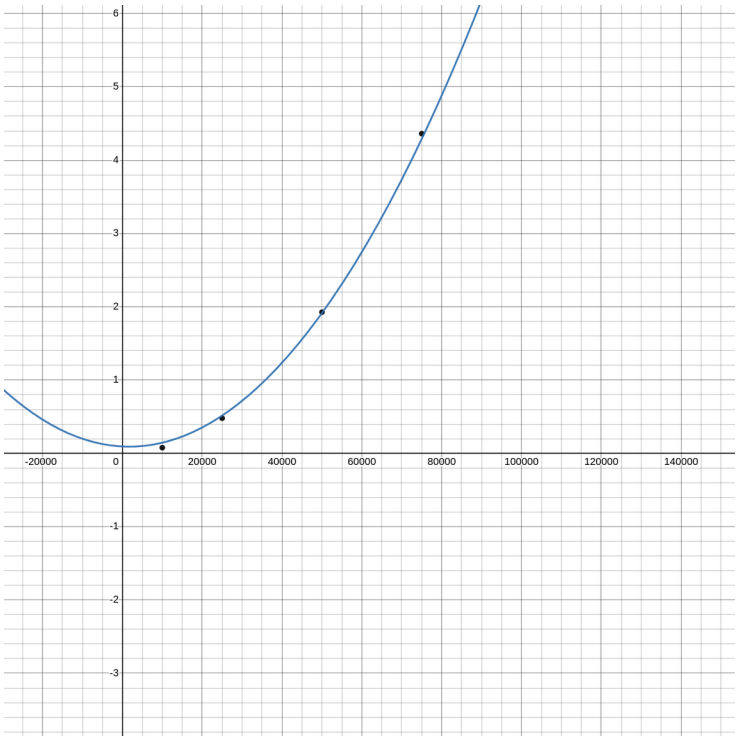
Best Case Selection Sort:



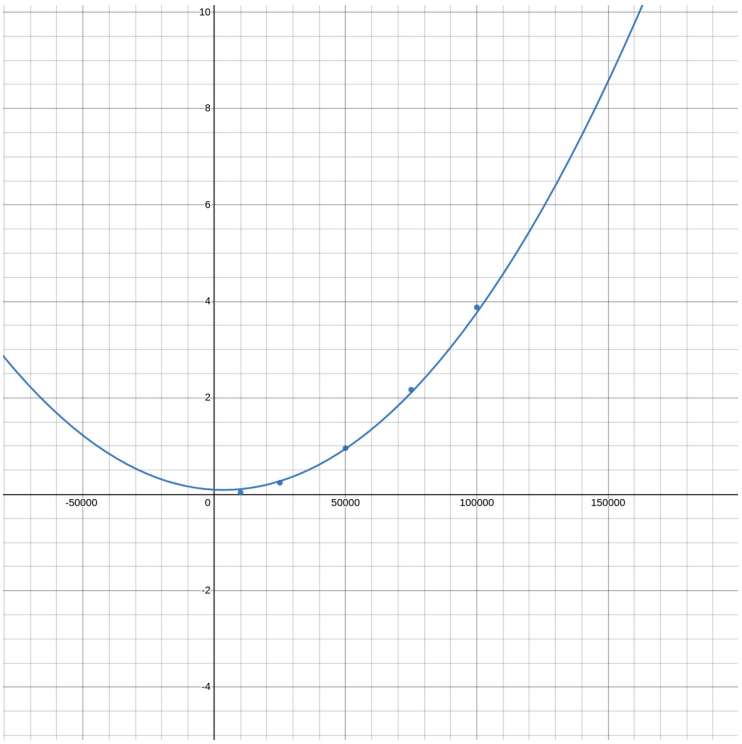
Best Case Bubble Sort 2:



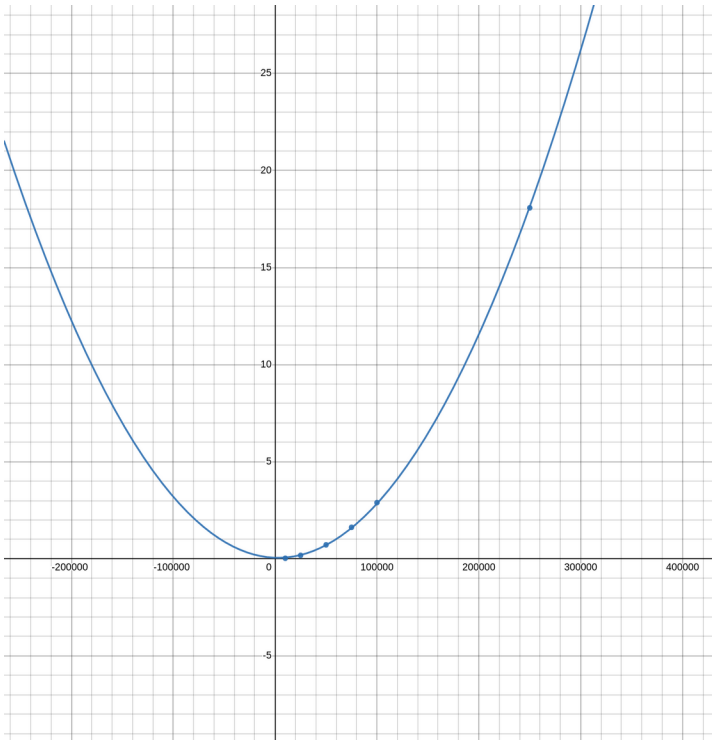
Worst Case Bubble Sort:



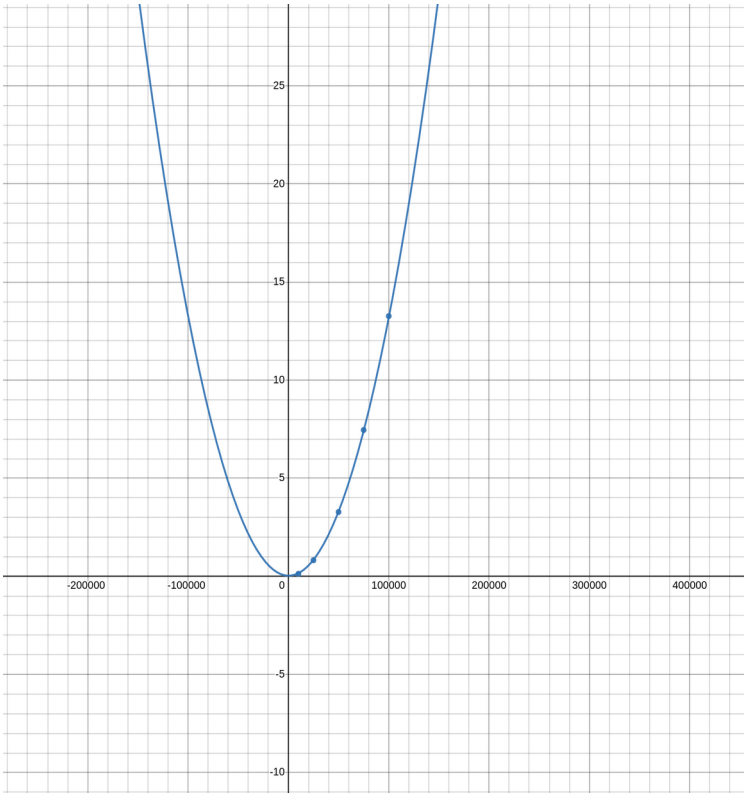
Worst Case Insertion Sort:



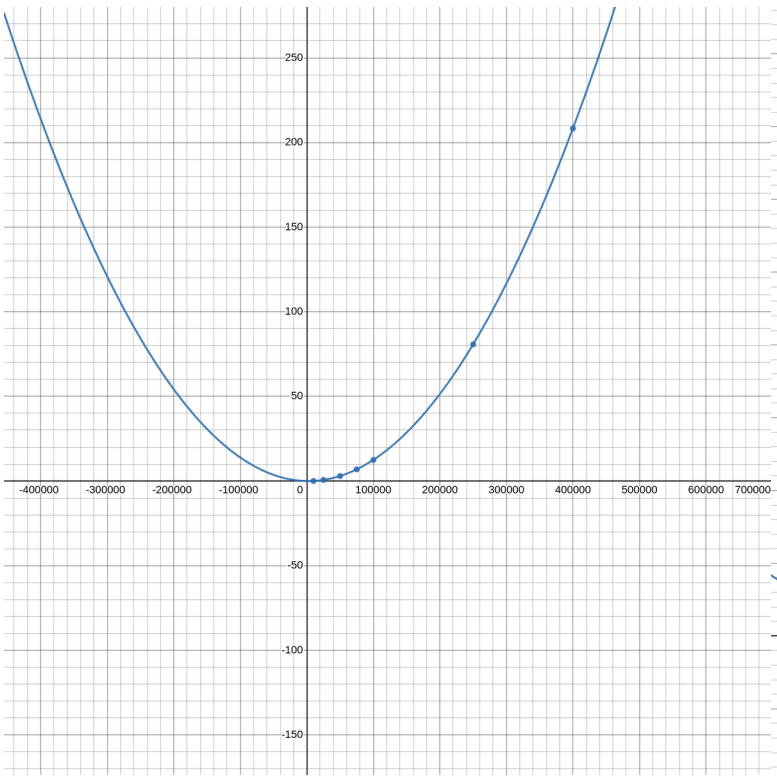
Worst Case Selection Sort:



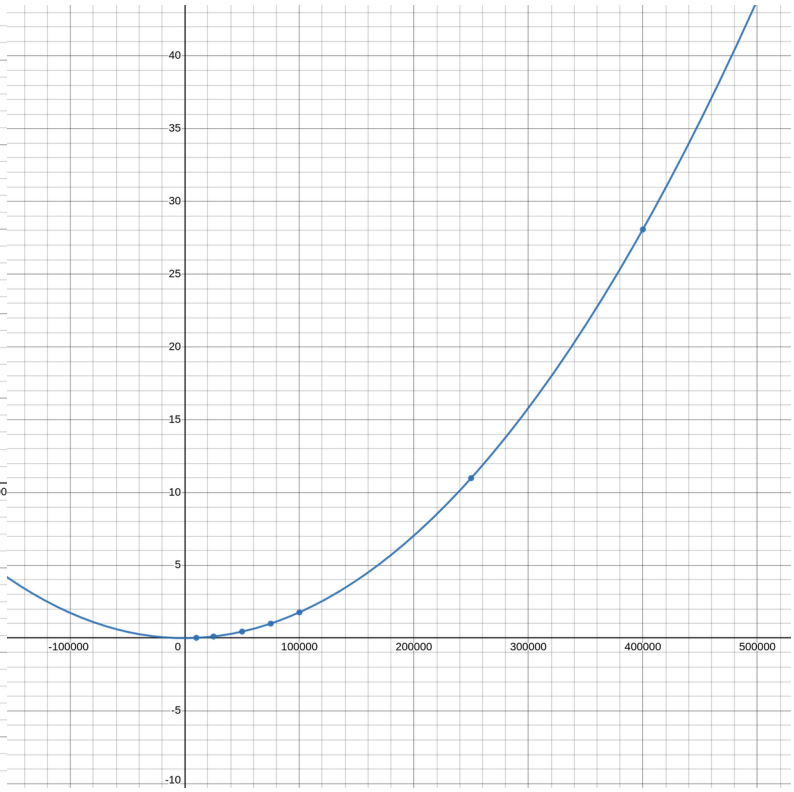
Worst Case Bubble Sort 2:



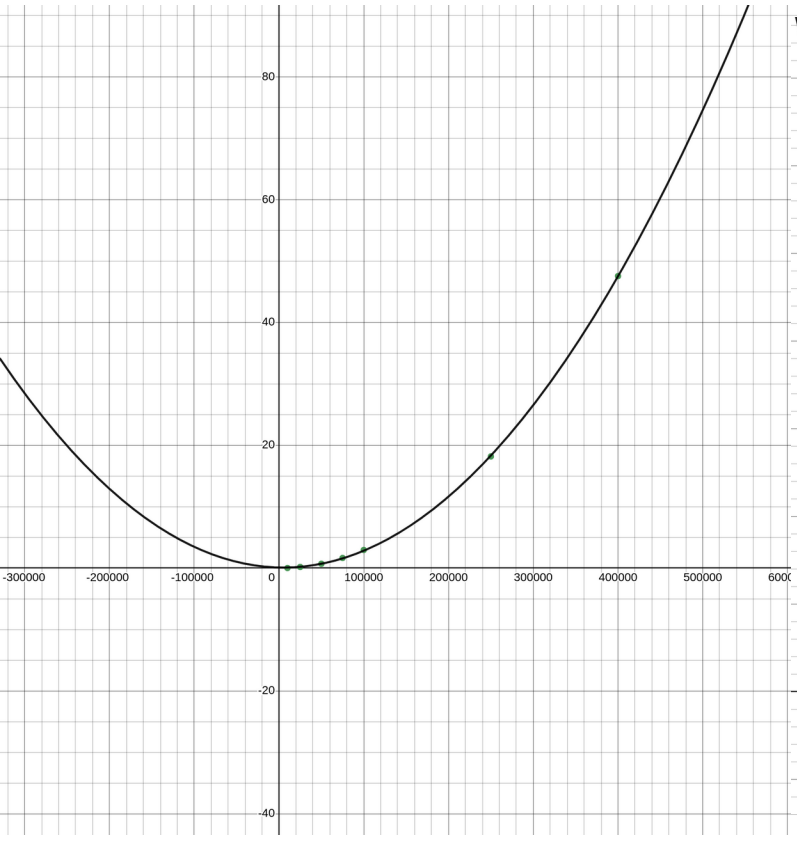
Average Case Bubble Sort:



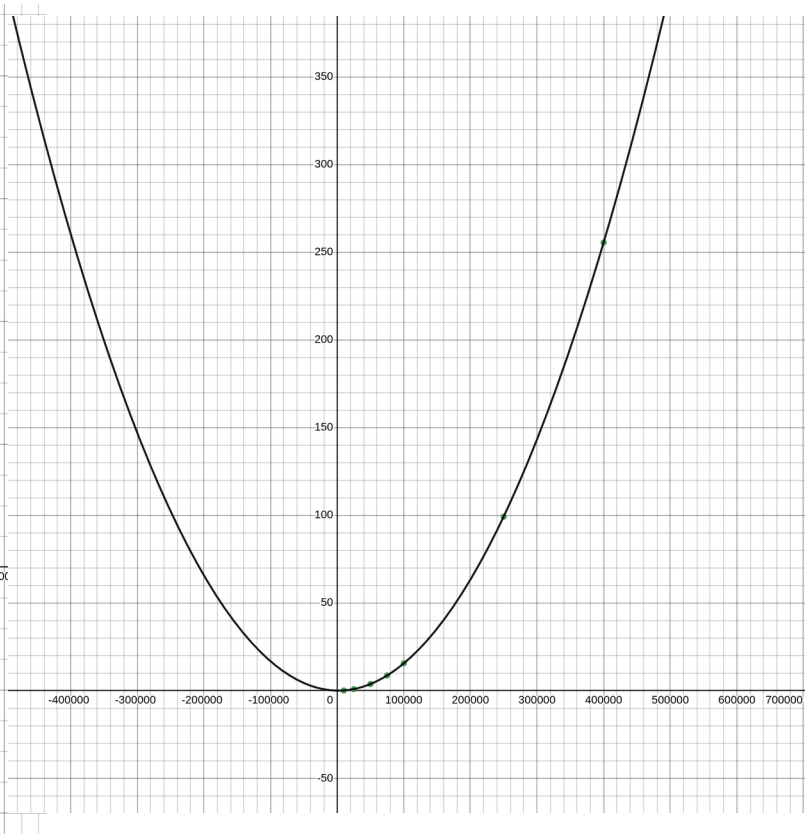
Average Case Insertion Sort:



Average Case Selection Sort:

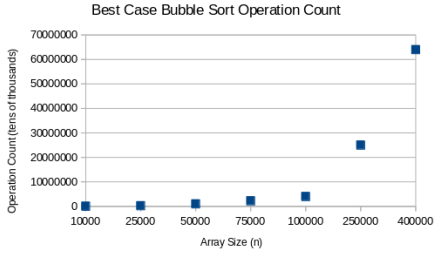


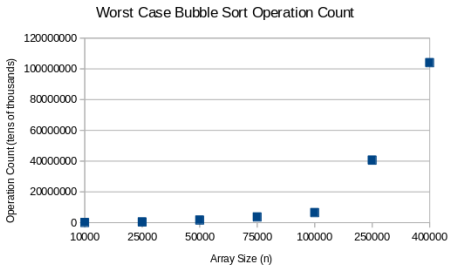
Average Case Bubble Sort 2:

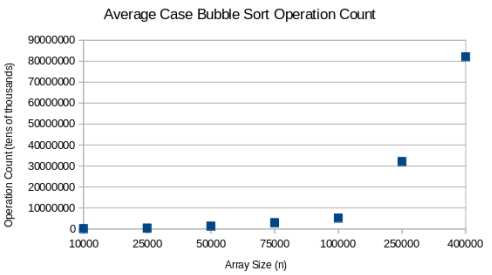


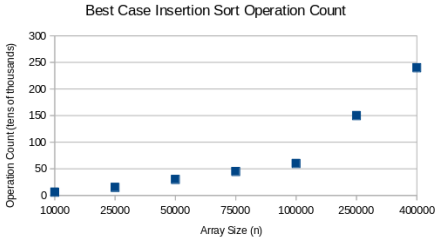
Counting Spreadsheet and Graphs

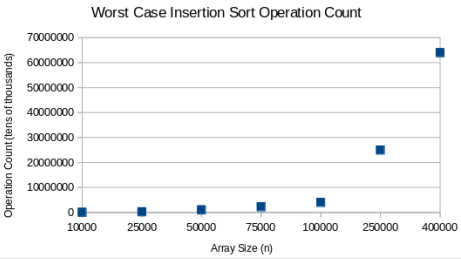
Array Size (n)	BC Bubble O	BC Insertion	BC Selection	BC Bubble 2	WC Bubble O	WC Insertion	WC Selection	WC Bubble 2	AC Bubble O	AC Insertion	AC Selection	AC Bubble 2 Operation Count (tens of thousands)
10000	39999	6	20009	40001	64997	40002	22509	69998	51381	18216	20012	53660
25000	249999	15	125022	250004	406243	250002	140647	437497	320013	112037	125031	334021
50000	999999	30	500045	1000009	1624987	1000010	562545	1749994	1282214	451573	500063	1338667
75000	2249999	45	1125067	2250014	3656231	2250015	1265692	3937492	2888077	1020969	1125095	3015708
100000	3999999	60	2000090	4000019	6499974	4000020	2250090	6999989	5128191	1805166	2000126	5353850
250000	24999999	150	12500225	25000049	40624937	25000050	14062725	43749974	32056623	11290746	12500315	33467997
400000	63999999	240	32000360	64000079	103999899	64000080	36000360	111999959	82108242	28973427	32000506	85729970

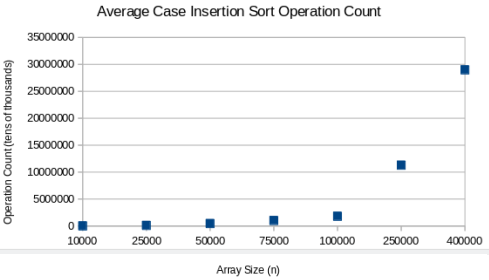


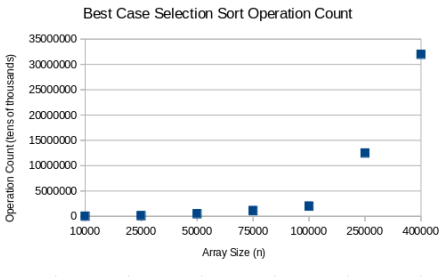


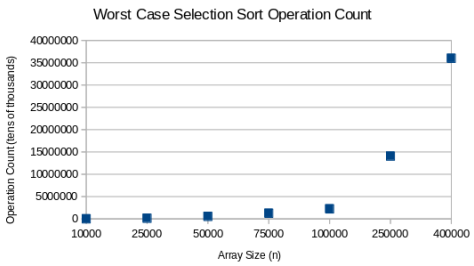


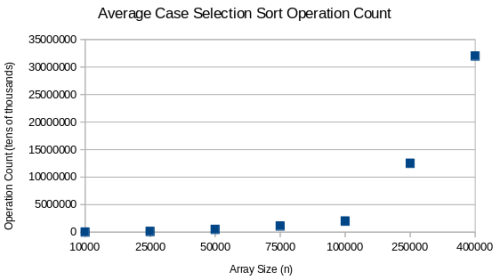


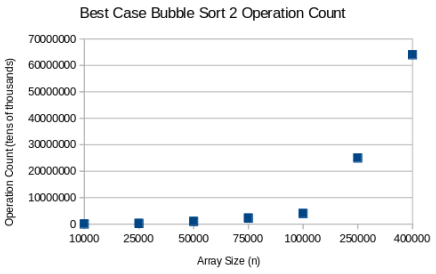


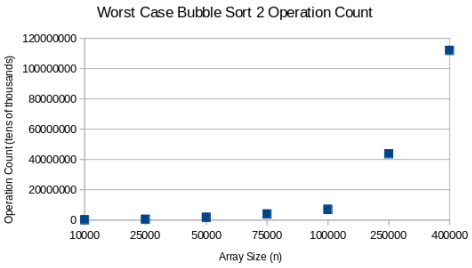


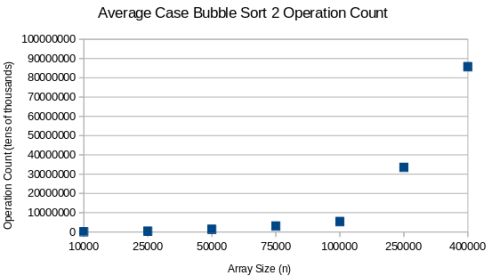






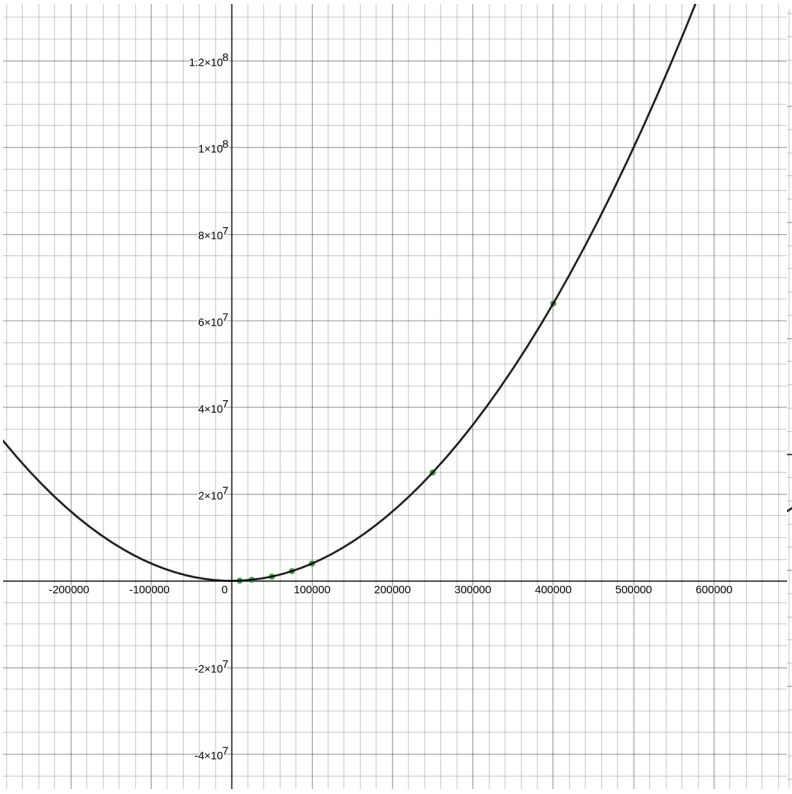




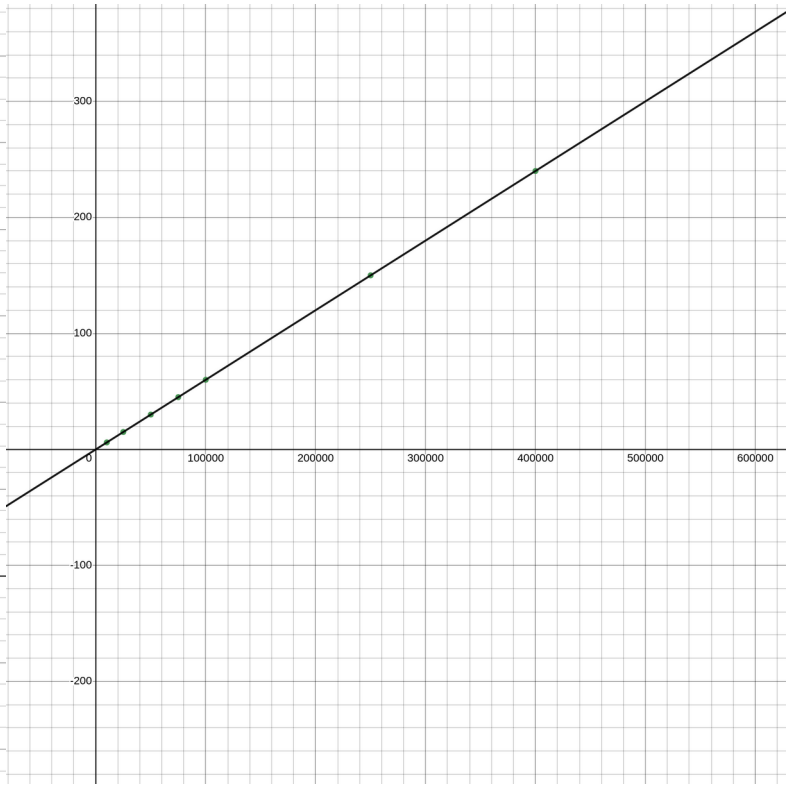


Curve Fitting Graphs for Counting

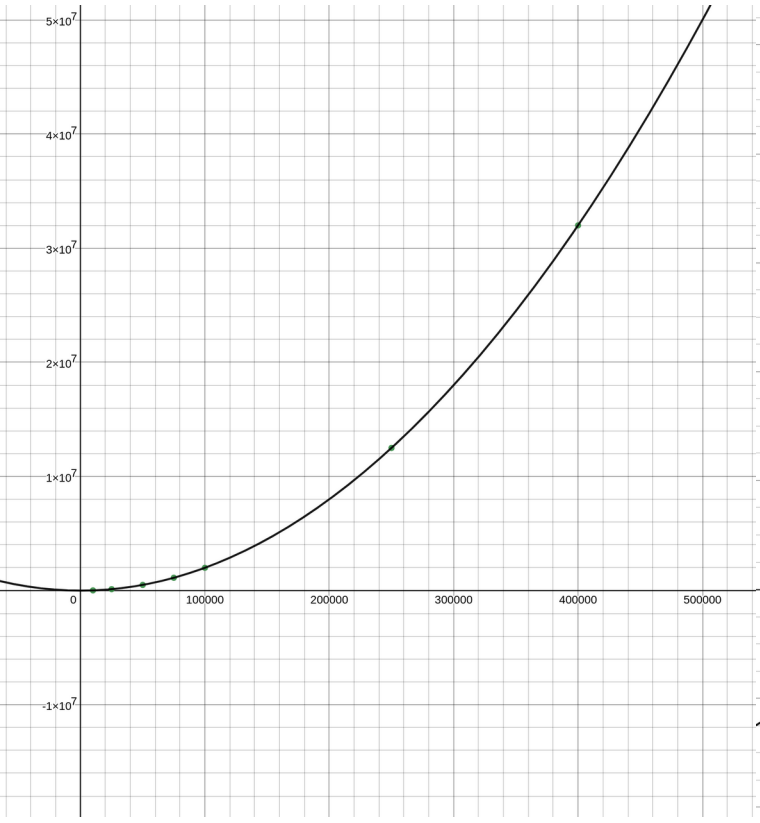
Best Case Bubble Sort:



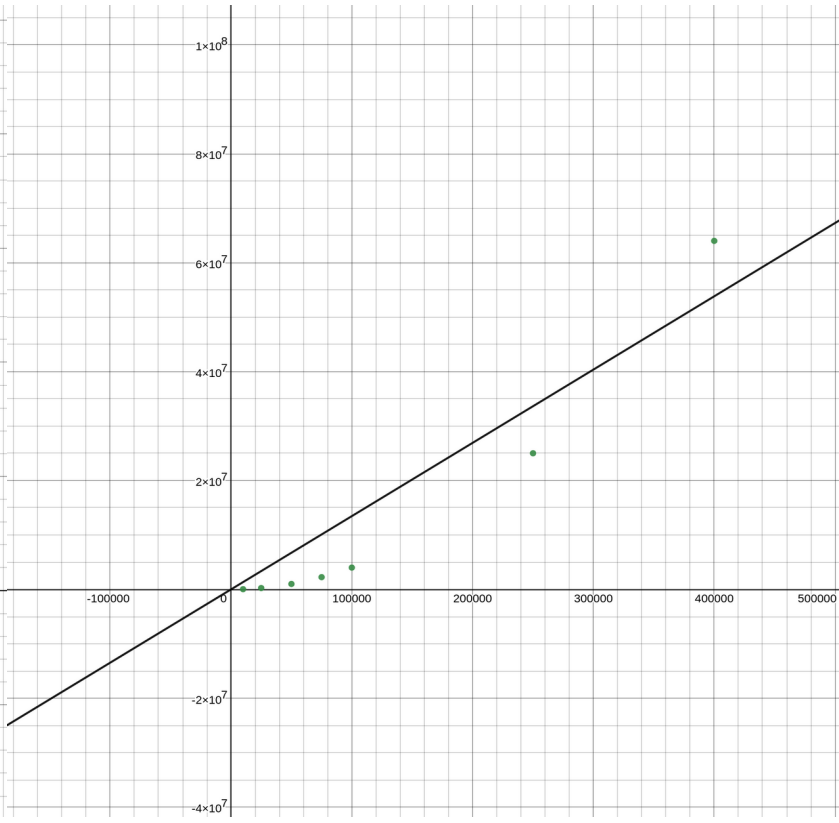
Best Case Insertion Sort:



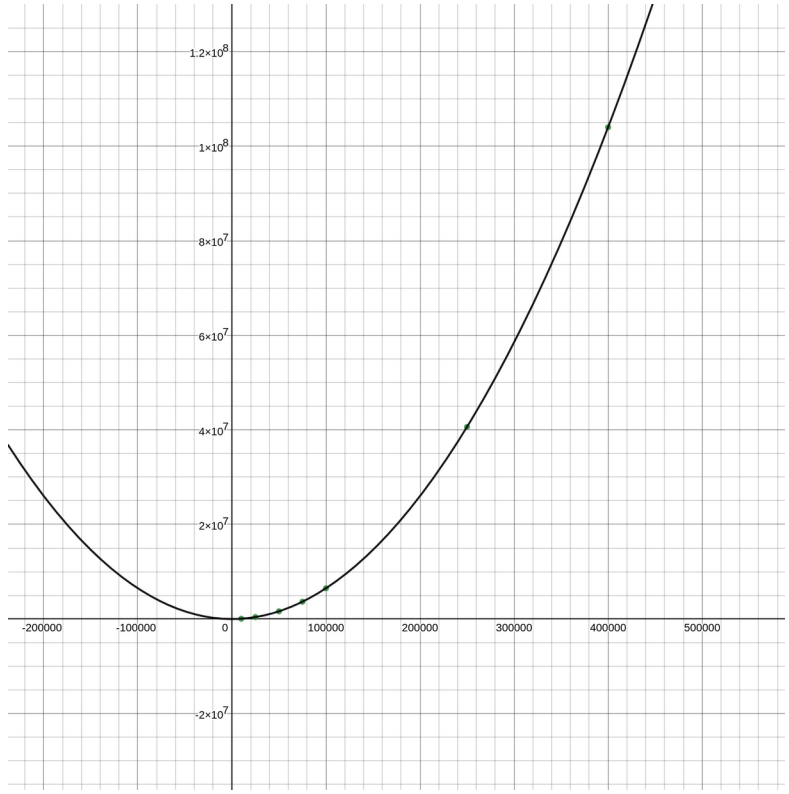
Best Case Selection Sort:



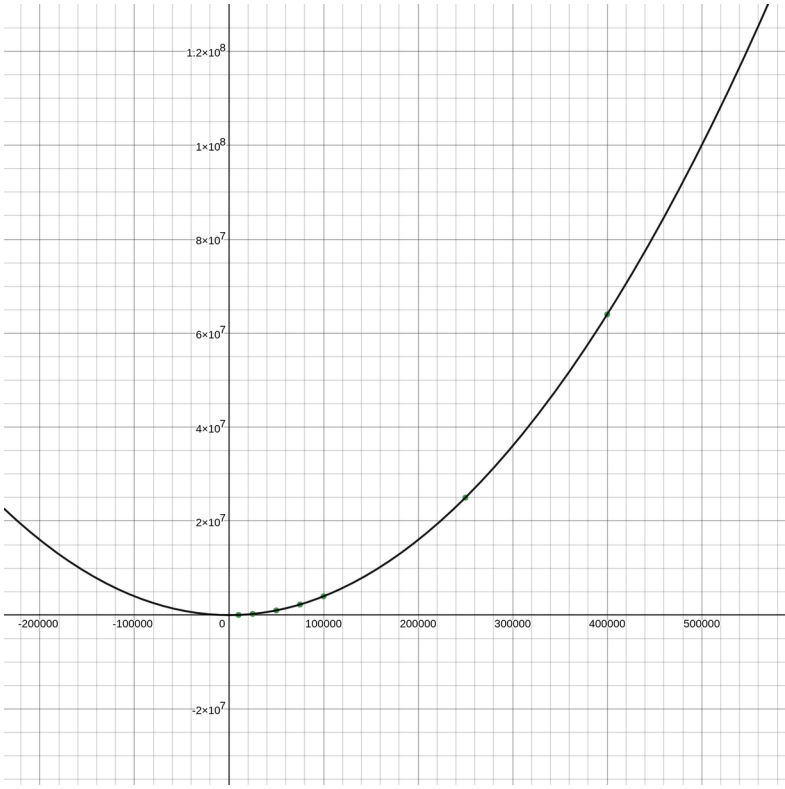
Best Case Bubble Sort 2:



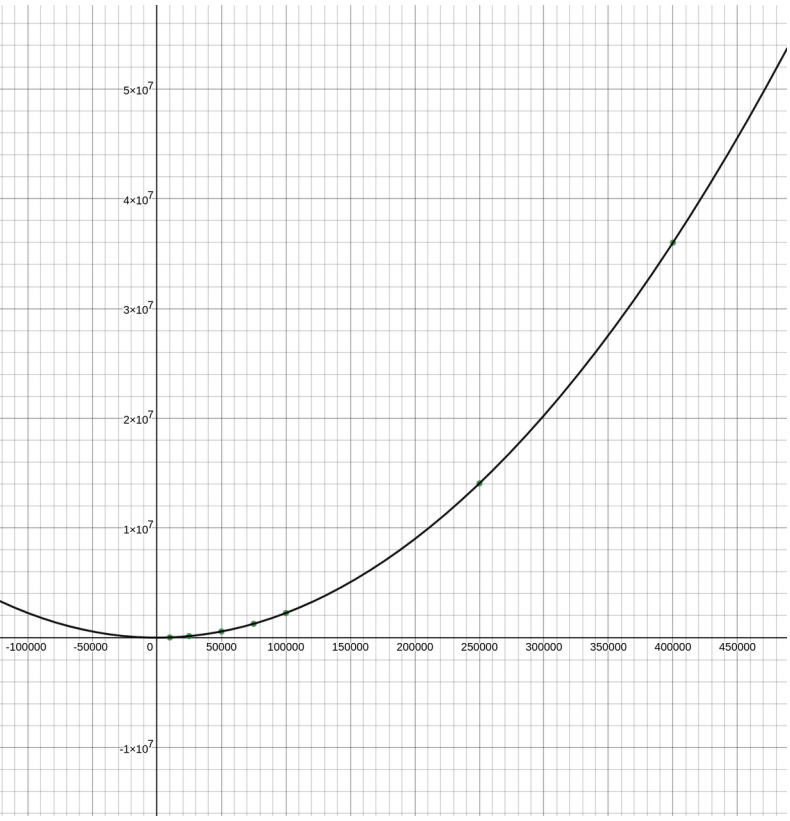
Worst Case Bubble Sort:



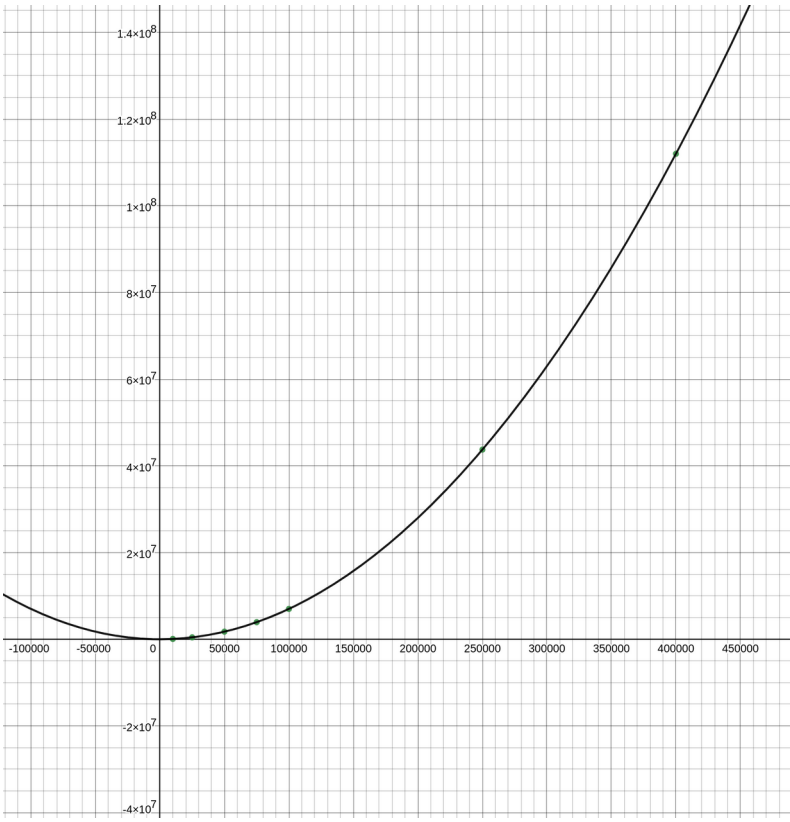
Worst Case Insertion Sort:



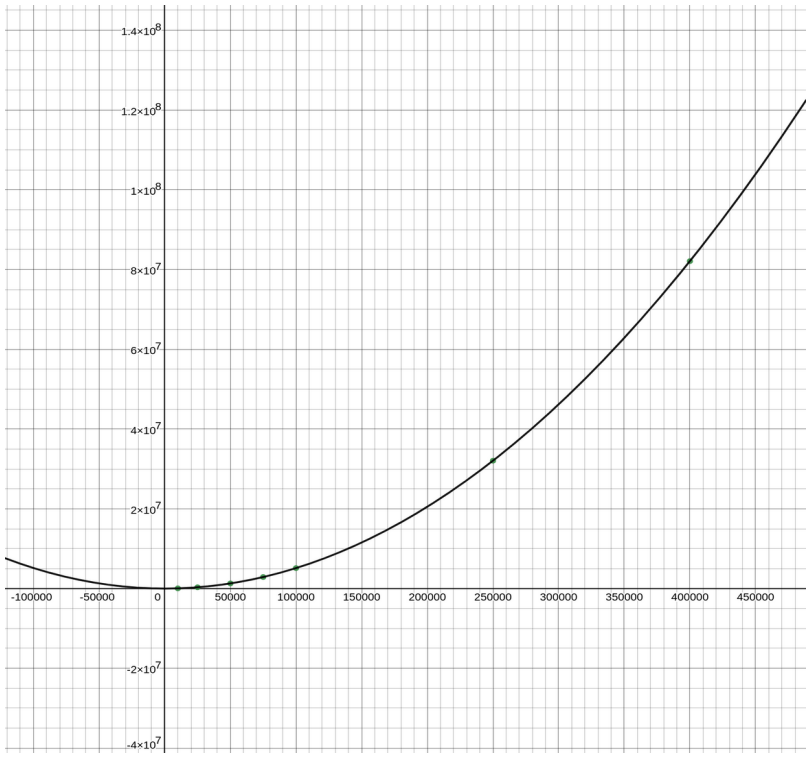
Worst Case Selection Sort:



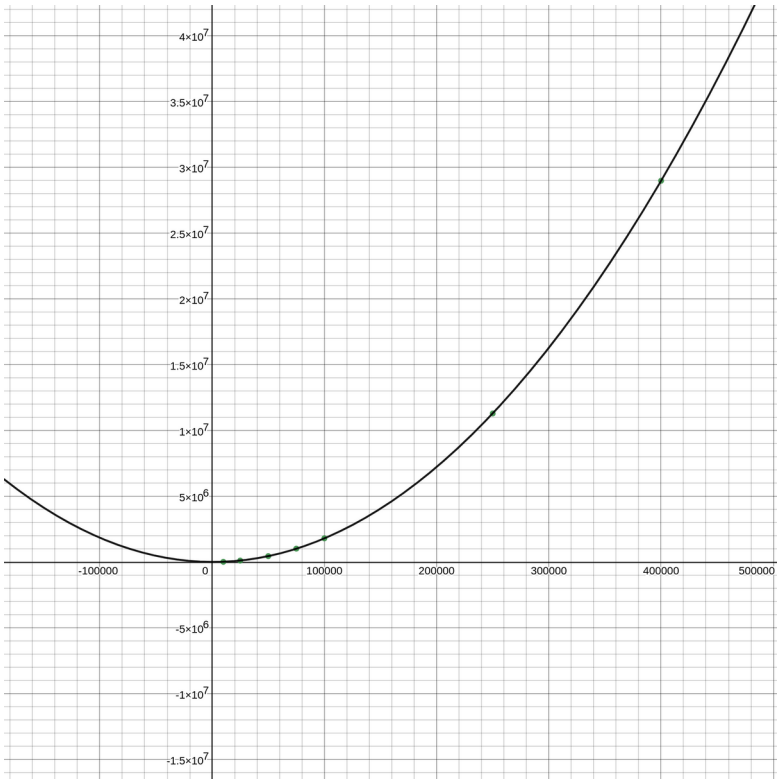
Worst Case Bubble Sort 2:



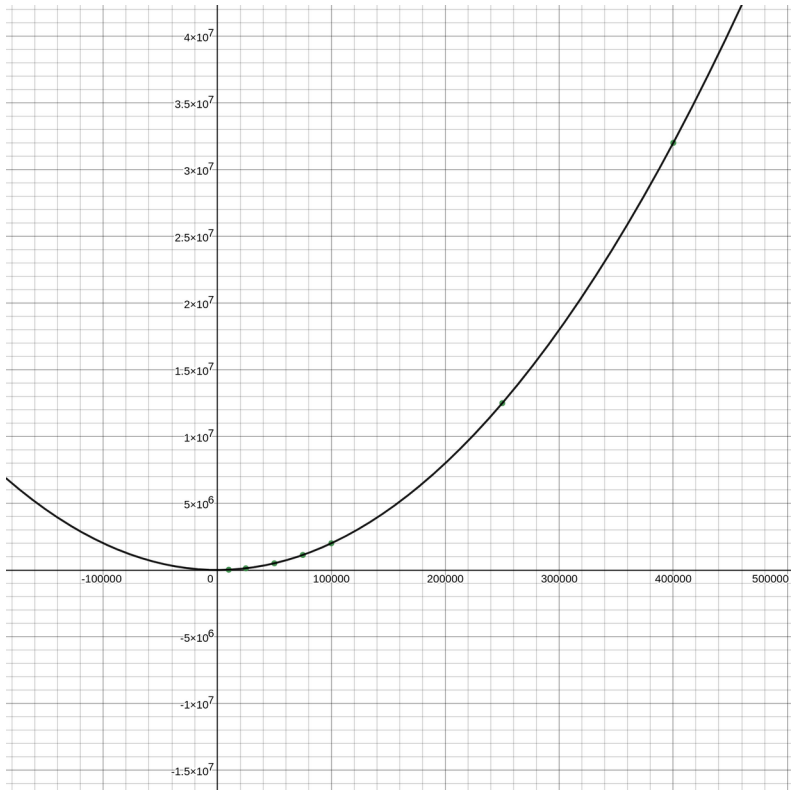
Average Case Bubble Sort:



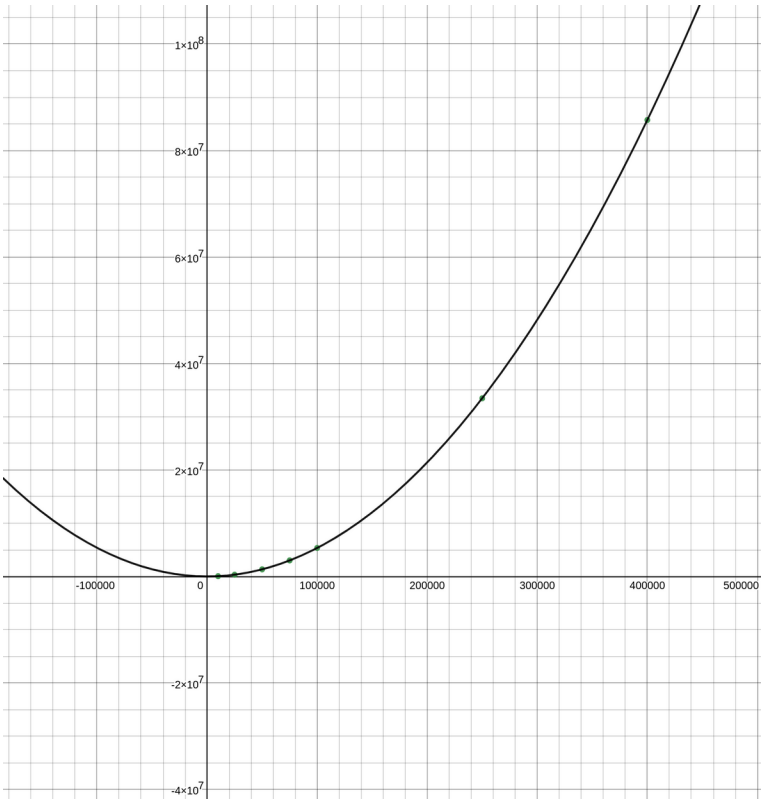
Average Case Insertion Sort:



Average Case Selection Sort:



Average Case Bubble Sort 2:



Responses and Conclusions

1. Bubble Sort and Selection Sort were found to be tightly bound to n^2 complexity for every scenario, while Insertion and Bubble 2 were only tightly bound to n^2 complexity for the worst and average cases. Insertion Sort and Bubble Sort 2 were found to be tightly bound to linear complexity, or n, for their best cases.

Of all the cases with the same complexity, the equation found to have the largest coefficient was the worst case of Bubble Sort 2, with a coefficient of 7. Bubble Sort's worst case was closely behind with a coefficient of 6.5.

Best Cases (fastest to slowest):
Bubble Sort 2, Insertion Sort, Selection Sort, Bubble Sort
Worst Cases (fastest to slowest):
Selection Sort, Insertion Sort, Bubble Sort, Bubble Sort 2
Average Cases (fastest to slowest):
All tied

2. The empirical coefficients were vastly different from the theoretical coefficients. The empirical coefficients were usually something to the order of n*10^-3 or maybe n*10^-4, while the theoretical coefficients were plain integers. Reasons for this discrepancy could be the fact that the array sizes are in the magnitude of tens of thousands, and so the values of the counts are much different from counting by hand. Another reason could also be that each operation was counted as equivalent when counting by hand and the computer does not count different types of operations (logic, arithmetic, assignment) as equivalent.

3. Best Cases (fastest to slowest):
Insertion Sort, Selection Sort, Bubble Sort, Bubble Sort 2
Worst Cases (fastest to slowest):
Selection Sort, Insertion Sort, Bubble Sort, Bubble Sort 2
Average Cases (fastest to slowest):
Insertion Sort, Selection Sort, Bubble Sort, Bubble Sort 2

These cases are not the same as the cases derived in part one, but they are similar. The main error comes from the best case assumption of Bubble Sort 2. It was assumed that it would run the fastest at its best case because of the boolean statement that would know the array was sorted after one pass over, but this turned out to not be the case. Either the boolean statement did not catch, or the given function does not work as originally assumed and the calculation was wrong. The best and worst case was predicted completely accurately besides that error with Bubble Sort 2. The average cases were not able to be put into a cost function, so their results for part one were essentially all tied, but as the empirical testing showed, this is not the case. Although they all do run with n^2 complexity, the average case has Insertion Sort as it's fastest, and then selection, bubble, and bubble

4. The results for the empirical coefficients of the lines of best fit for the count functions were similar to the empirical coefficients for timing, in that they were vastly different from the empirical calculations. The count function displays the sum of all the arithmetic, logic, and assignment operations returned from each sorting algorithm. Because the data is being summed for arrays in the order of tens of thousands, and even hundreds of thousands, and the running time of the functions is exponential, the outputs are massive, and the coefficient of a line to fit that data must be incredibly small to compensate.

Best Cases (fastest to slowest):
Insertion Sort, Selection Sort, Bubble Sort, Bubble Sort 2
Worst Cases (fastest to slowest):

Selection Sort, Insertion Sort, Bubble Sort, Bubble Sort 2
Average Cases (fastest to slowest):
Insertion Sort, Selection Sort, Bubble Sort, Bubble Sort 2

These cases are the same for counting as they are for timing, and therefore different from the predicted values from part one for the same reasons.

5. Overall, the two of the four algorithms are valid to use for sorting large quantities of data. The Insertion Sort and the Selection Sort both have running times under or close to a minute for arrays with elements up to 400,000. The Bubble Sorts take a few minutes or more for arrays of that size. Comparing Insertion Sort and Selection Sort, it can be concluded that the insertion sort has the benefits of being better in the best case and the average case, but worse in the worst case. The Selection Sort is better in the worst case, and this is because for every scenario it runs about the same time, with a negligible margin of difference. Due to these reasons, it can be said that Selection Sort is the most consistent sorting algorithm out of the four.