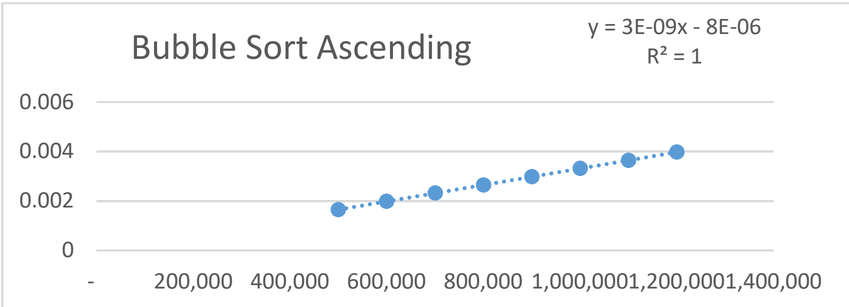
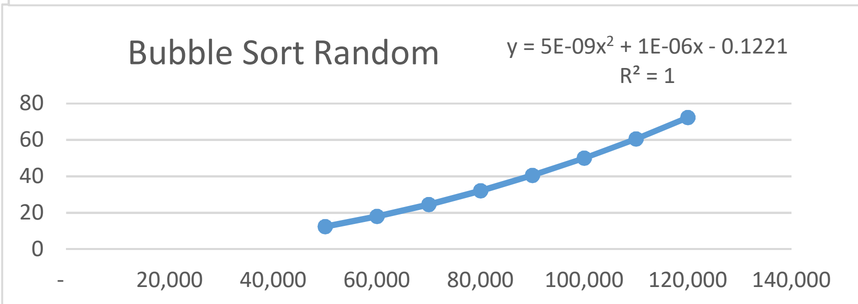
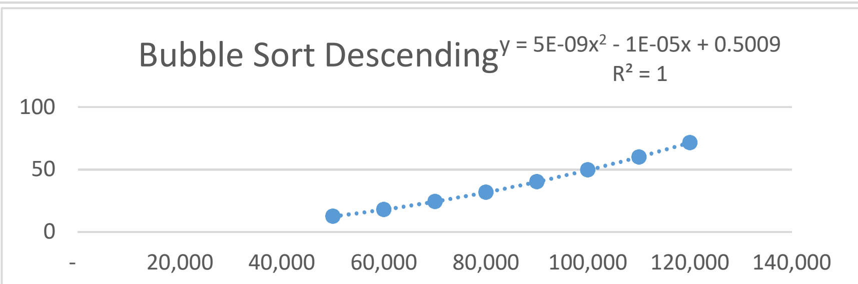
Sorting Type	Order	Test Size	Time	Big-O	Big-O Justification	Estimate for 10,000,000 in Seconds
Bubble	ascending	500,000	0.00165211	O(N)	When I added a trend line to my graph, the R^2 was exactly one. Meaing it was a perfect fit. In additon, it can be seen that the time goes up by .0003 each time	0.029992 I used the best fit line and pluged 10,000,000 in for x
		600,000	0.00198357			
		700,000	0.0023145			
		800,000	0.00264589			
		900,000	0.0029813			
		1,000,000	0.00331666			
		1,100,000	0.00364023			
		1,200,000	0.00397527			
	random	50,000	12.4322	O(N^2)	Looking at the graph the X^2 curve can start to be seen. In addition, the trend line had an R^2 of 1.	500009.8779 I used the best fit line and pluged 10,000,000 in for x
		60,000	17.9781			
		70,000	24.4935			
		80,000	32.1399			
		90,000	40.609			
		100,000	49.9707			
		110,000	60.5437			
		120,000	72.2546			
	descending	50,000	12.4154	O(N^2)	Looking at the graph the X^2 curve can start to be seen. In addition, the trend line had an R^2 of 1.	499900.5009 I used the best fit line and pluged 10,000,000 in for x
		60,000	17.836			
		70,000	24.2723			
		80,000	31.72			
		90,000	40.1407			
		100,000	49.5693			
		110,000	59.9926			
		120,000	71.5542			
Insertion	ascending	500,000	0.00240693	O(N)	When I added a trend line to my graph, the R^2 was exactly one. Meaing it was a perfect fit. In additon, it can be seen that the time goes up by .0005 each time	.04998 I used the best fit line and pluged 10,000,000 in for x
		600,000	0.00289132			
		700,000	0.00336996			
		800,000	0.00385116			
		900,000	0.00433633			
		1,000,000	0.00481821			
		1,100,000	0.00534489			
		1,200,000	0.00577724			
	random	50,000	1.74491	O(N^2)	Looking at the graph the X^2 curve can start to be seen. In addition, the trend line had an R^2 of 1.	69999.9874 I used the best fit line and pluged 10,000,000 in for x
		60,000	2.51427			
		70,000	3.42145			
		80,000	4.44875			
		90,000	5.6254			
		100,000	6.95313			
		110,000	8.39581			
		120,000	9.98897			
	descending	50,000	3.48306	O(N^2)	Looking at the graph the X^2 curve can start	
		60,000	5.02106			
		70,000	6.82521			
		80,000	8.93423			

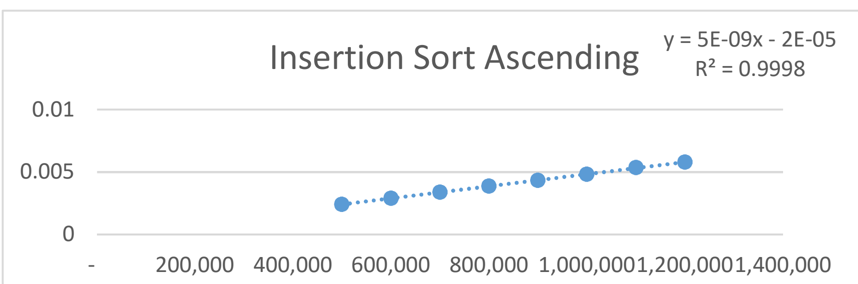
	descending	90,000	11.2782	O(N^2)	to be seen. In addition, the trend line had an R^2 of 1.	99999.4982 I used the best fit line and pluged 10,000,000 in for x
		100,000	13.9617			
		110,000	16.8681			
		120,000	20.0847			
	ascending	50,000	2.63667	O(N^2)	Looking at the graph the X^2 curve can start to be seen. In addition, the trend line had an R^2 of 1.	100029.8995 I used the best fit line and pluged 10,000,000 in for x
		60,000	3.79725			
		70,000	5.15278			
		80,000	6.74552			
		90,000	8.52107			
		100,000	10.5771			
		110,000	12.7256			
		120,000	15.1326			
selection	random	50,000	2.80287	O(N^2)	Looking at the graph the X^2 curve can start to be seen. In addition, the trend line had an R^2 of 1.	1000039.8839 I used the best fit line and pluged 10,000,000 in for x
		60,000	4.03273			
		70,000	5.48163			
		80,000	7.16974			
		90,000	9.07126			
		100,000	11.1925			
		110,000	13.5159			
		120,000	16.0645			
	descending	50,000	2.72311	O(N^2)	Looking at the graph the X^2 curve can start to be seen. In addition, the trend line had an R^2 of 1.	100006.9906 I used the best fit line and pluged 10,000,000 in for x
		60,000	3.91024			
		70,000	5.32237			
		80,000	6.95981			
		90,000	8.7856			
		100,000	10.8341			
		110,000	13.1121			
		120,000	15.6027			
	ascending	500,000	0.0486559	O(NlogN)	This is nlogn because it is very close to being exactly linear, but isn't quite because it is multiplied by log(n). Thus causing not a good linear fit line. Also this makes sense because log(n) won't affect the graph compared to n.	.8054 I used the best fit line and pluged 10,000,000 in for x knowing that the actual time would be slightly larger
		600,000	0.0566355			
		700,000	0.0593768			
		800,000	0.0706971			
		900,000	0.0831534			
		1,000,000	0.0910074			
		1,100,000	0.0979086			
		1,200,000	0.104106			
quick	random	500,000	0.0883772	O(NlogN)	This is nlogn because it is very close to being exactly linear, but isn't quite because it is multiplied by log(n). Thus causing not a good linear fit line. Also this makes sense because log(n) won't affect the graph compared to n.	1.9882 I used the best fit line and pluged 10,000,000 in for x knowing that the actual time would be slightly larger
		600,000	0.107096			
		700,000	0.134573			
		800,000	0.148976			
		900,000	0.171701			
		1,000,000	0.198801			
		1,100,000	0.206702			
		1,200,000	0.230121			
		500,000	0.0411105		This is nlogn because it is very close to being exactly linear, but isn't	.9862 I used the best
		600,000	0.0454013			
		700,000	0.0545364			

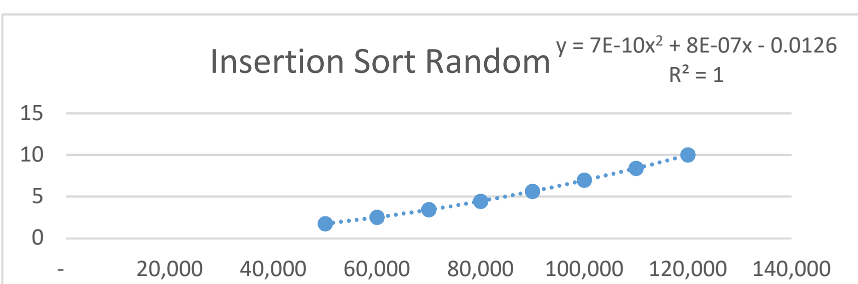
	descending	800,000	0.0606201	O(NlogN)	quite because it is multiplied by log(n). Thus causing not a good linear fit line. Also this makes sense because log(n) won't affect the graph compared to n.	fit line and pluged 10,000,000 in for x knowing that the actual time would be slightly larger
		900,000	0.0819382			
		1,000,000	0.0911331			
		1,100,000	0.098578			
		1,200,000	0.106144			
	ascending	500,000	0.0611842	O(NlogN)	This is nlogn because it is very close to being exactly linear, but isn't quite because it is multiplied by log(n). Thus causing not a good linear fit line. Also this makes sense because log(n) won't affect the graph compared to n.	1.0106 I used the best fit line and pluged 10,000,000 in for x knowing that the actual time would be slightly larger
		600,000	0.0716825			
		700,000	0.083145			
		800,000	0.0927485			
		900,000	0.104099			
		1,000,000	0.111122			
		1,100,000	0.12199			
		1,200,000	0.134331			
merge	random	500,000	0.109171	O(NlogN)	This is nlogn because it is very close to being exactly linear, but isn't quite because it is multiplied by log(n). Thus causing not a good linear fit line. Also this makes sense because log(n) won't affect the graph compared to n.	1.9986 I used the best fit line and pluged 10,000,000 in for x knowing that the actual time would be slightly larger
		600,000	0.1376039			
		700,000	0.157244			
		800,000	0.179712			
		900,000	0.200812			
		1,000,000	0.223564			
		1,100,000	0.247549			
		1,200,000	0.270506			
	descending	500,000	0.059403	O(NlogN)	This is nlogn because it is very close to being exactly linear, but isn't quite because it is multiplied by log(n). Thus causing not a good linear fit line. Also this makes sense because log(n) won't affect the graph compared to n.	1.0074 I used the best fit line and pluged 10,000,000 in for x knowing that the actual time would be slightly larger
		600,000	0.0720787			
		700,000	0.0819603			
		800,000	0.0939546			
		900,000	0.10231			
		1,000,000	0.113991			
		1,100,000	0.123579			
		1,200,000	0.135322			

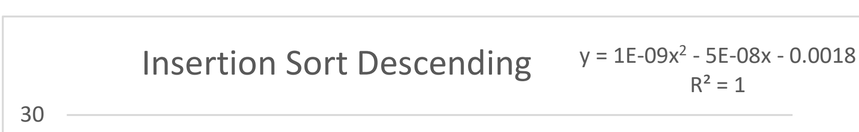

Bubble Sort Ascending
y = 3E-09x - 8E-06
R² = 1
0.006
0.004
0.002
0
-
200,000 400,000 600,000 800,000 1,000,0001,200,0001,400,000

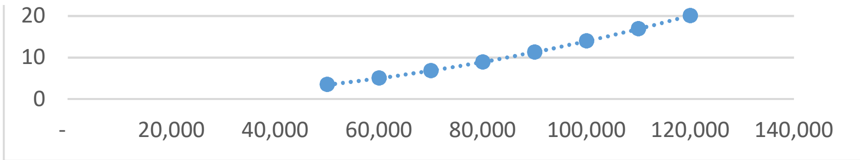

Bubble Sort Random
y = 5E-09x² + 1E-06x - 0.1221
R² = 1
80
60
40
20
0
-
20,000 40,000 60,000 80,000 100,000 120,000 140,000

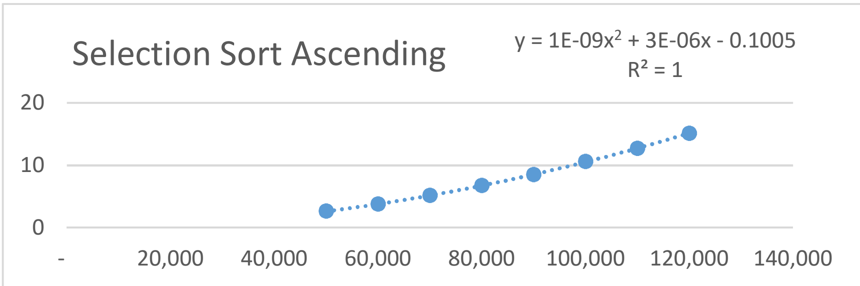

Bubble Sort Descending
y = 5E-09x² - 1E-05x + 0.5009
R² = 1
100
50
0
-
20,000 40,000 60,000 80,000 100,000 120,000 140,000

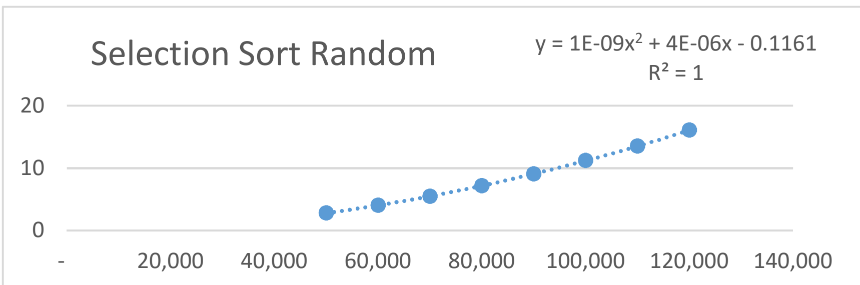

Insertion Sort Ascending
y = 5E-09x - 2E-05
R² = 0.9998
0.01
0.005
0
-
200,000 400,000 600,000 800,000 1,000,0001,200,0001,400,000

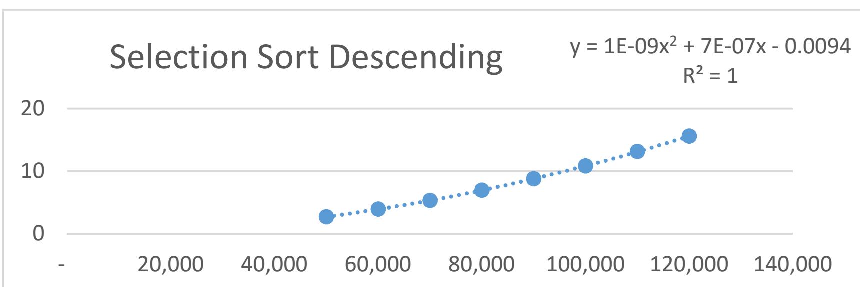

Insertion Sort Random
y = 7E-10x² + 8E-07x - 0.0126
R² = 1
15
10
5
0
-
20,000 40,000 60,000 80,000 100,000 120,000 140,000

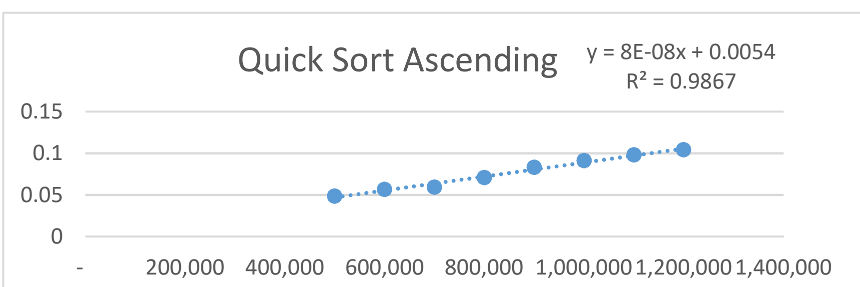

Insertion Sort Descending
y = 1E-09x² - 5E-08x - 0.0018
R² = 1
30

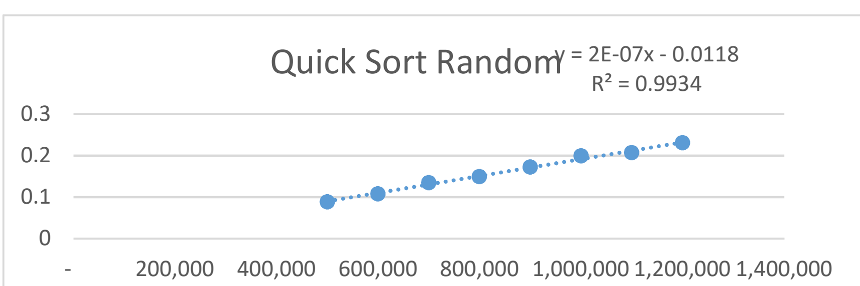

20
10
0
-
20,000
40,000
60,000
80,000
100,000
120,000
140,000

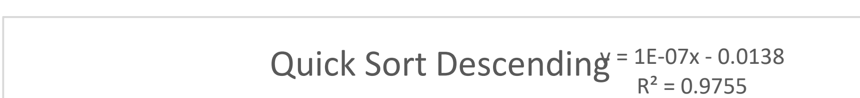

Selection Sort Ascending
$y = 1E-09x^2 + 3E-06x - 0.1005$
$R^2 = 1$
20
10
0
-
20,000
40,000
60,000
80,000
100,000
120,000
140,000

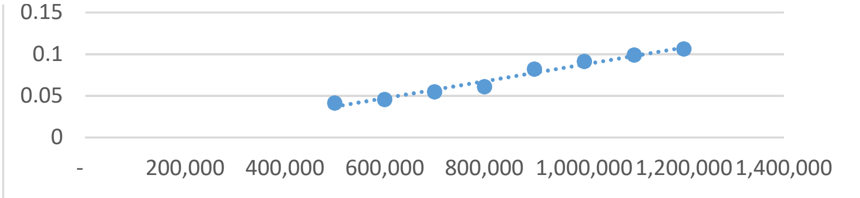

Selection Sort Random
$y = 1E-09x^2 + 4E-06x - 0.1161$
$R^2 = 1$
20
10
0
-
20,000
40,000
60,000
80,000
100,000
120,000
140,000

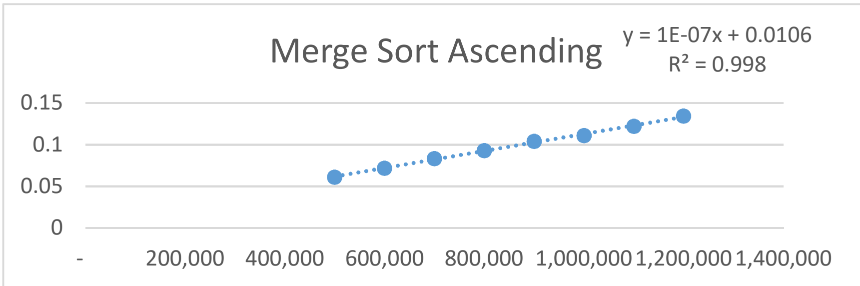

Selection Sort Descending
$y = 1E-09x^2 + 7E-07x - 0.0094$
$R^2 = 1$
20
10
0
-
20,000
40,000
60,000
80,000
100,000
120,000
140,000

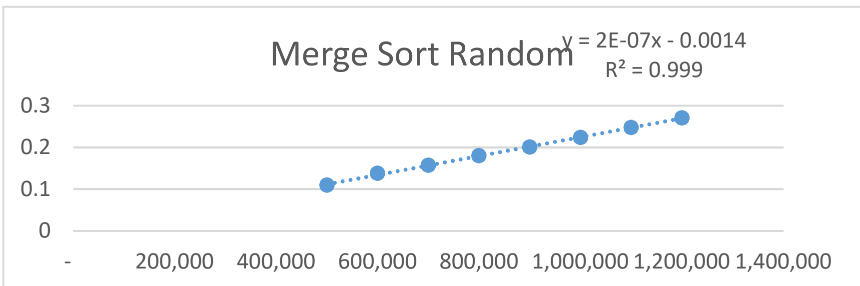

Quick Sort Ascending
$y = 8E-08x + 0.0054$
$R^2 = 0.9867$
0.15
0.1
0.05
0
-
200,000
400,000
600,000
800,000
1,000,000
1,200,000
1,400,000

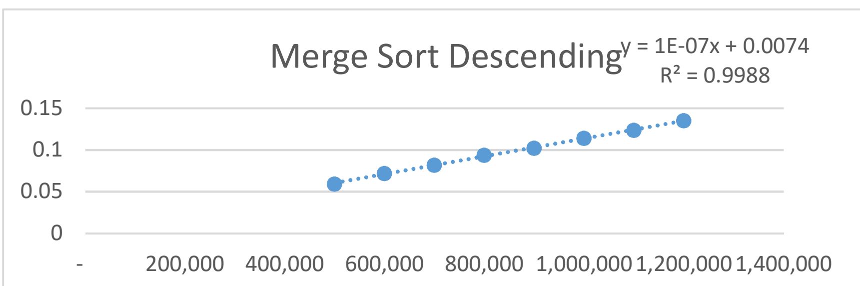

Quick Sort Random
$y = 2E-07x - 0.0118$
$R^2 = 0.9934$
0.3
0.2
0.1
0
-
200,000
400,000
600,000
800,000
1,000,000
1,200,000
1,400,000


Quick Sort Descending
$y = 1E-07x - 0.0138$
$R^2 = 0.9755$


0.15
0.1
0.05
0
-
200,000
400,000
600,000
800,000
1,000,000
1,200,000
1,400,000


Merge Sort Ascending
y = 1E-07x + 0.0106
R² = 0.998
0.15
0.1
0.05
0
-
200,000
400,000
600,000
800,000
1,000,000
1,200,000
1,400,000


Merge Sort Random
y = 2E-07x - 0.0014
R² = 0.999
0.3
0.2
0.1
0
-
200,000
400,000
600,000
800,000
1,000,000
1,200,000
1,400,000


Merge Sort Descending
y = 1E-07x + 0.0074
R² = 0.9988
0.15
0.1
0.05
0
-
200,000
400,000
600,000
800,000
1,000,000
1,200,000
1,400,000