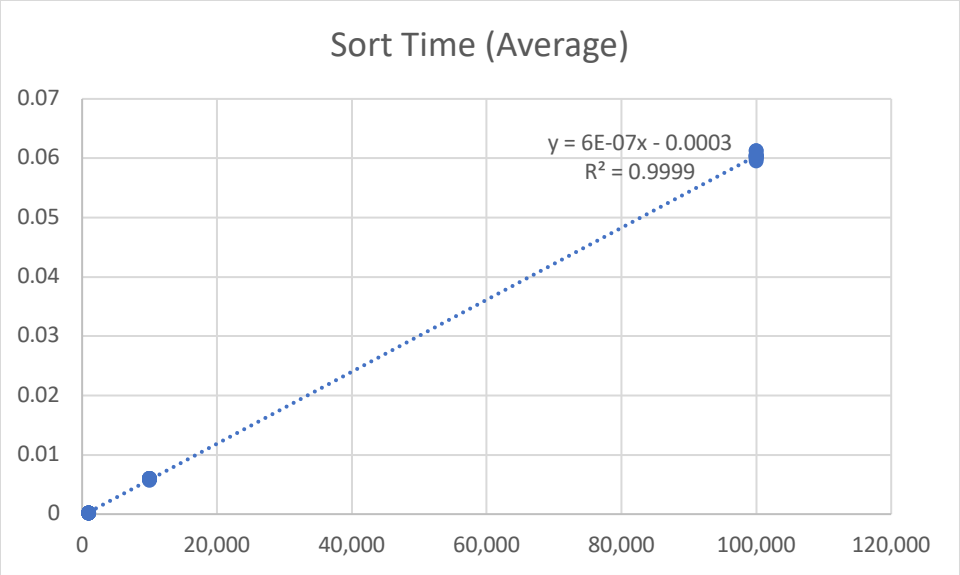
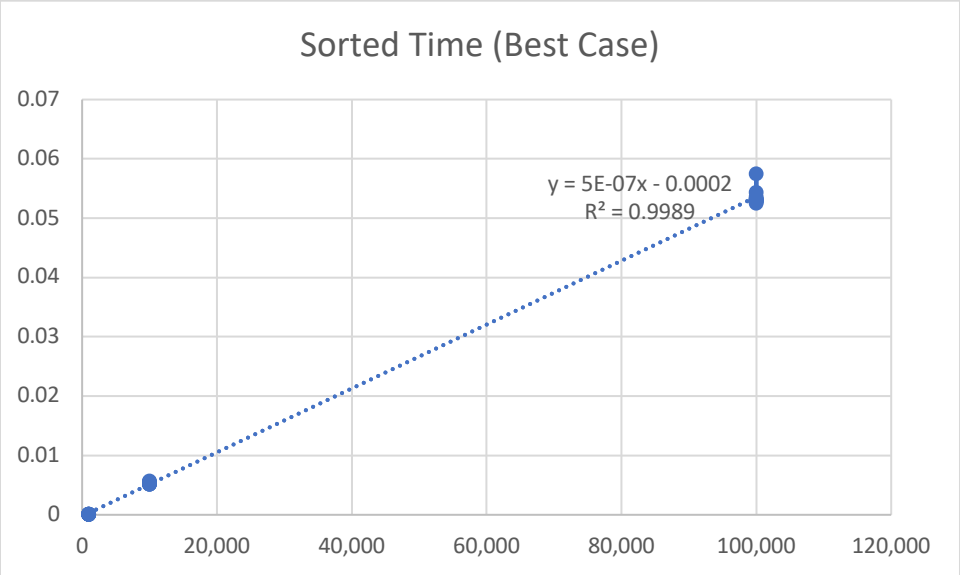
Documentation

“make” compiles all.
Settings.h contains the amount of integers to generate and range of integers
./inputGenerator creates the input textfile
./main runs both merge and insert sort

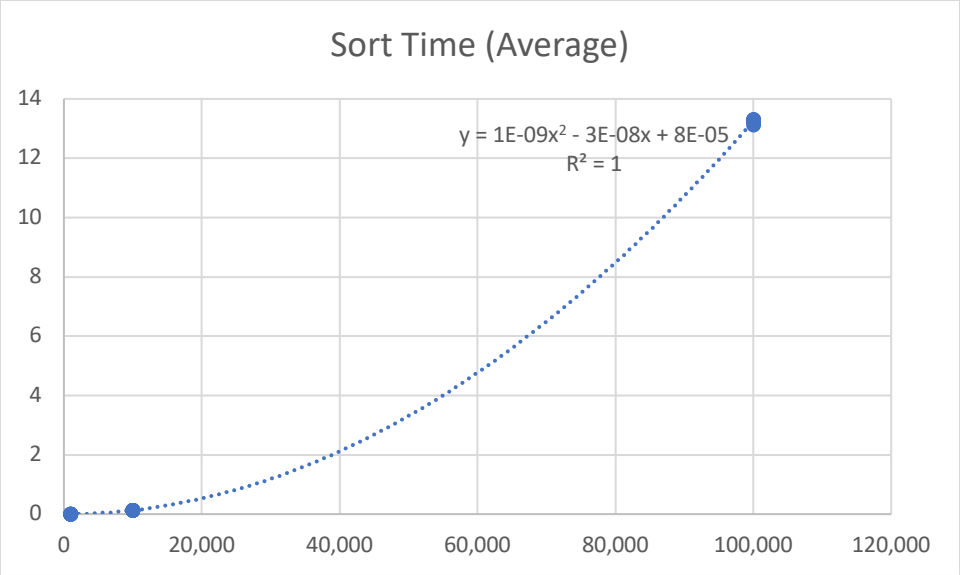
I included both sort algorithms into the same class and the .h file for simplicity.

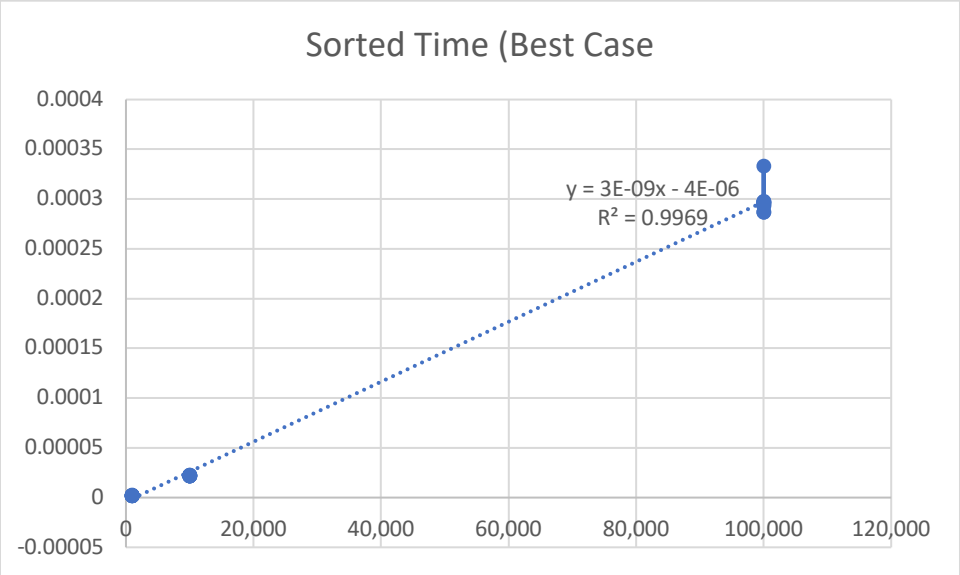
	Input Size	Sort Time	Swaps	Comparisons	Sorted Time	Sorted Swaps	Sorted Compa
Merge Sort	100,000	0.0595827	1668928	1536027	0.0531644	1668928	853904
	100,000	0.060146	1668928	1536030	0.0575571	1668928	853904
	100,000	0.0613014	1668928	1536719	0.0529664	1668928	853904
	100,000	0.060457	1668928	1536491	0.0533493	1668928	853904
	100,000	0.0598632	1668928	1536390	0.0529774	1668928	853904
	100,000	0.0605816	1668928	1536438	0.0532274	1668928	853904
	100,000	0.0612947	1668928	1536082	0.0543677	1668928	853904
	100,000	0.0602523	1668928	1536048	0.0529608	1668928	853904
	100,000	0.0600688	1668928	1536481	0.0525136	1668928	853904
	100,000	0.0603935	1668928	1536702	0.0527554	1668928	853904
Ave:	100,000	0.06039412	1668928	1536340.8	0.05358395	1668928	853904
	10,000	0.00589444	133616	120463	0.00526559	133616	69008
	10,000	0.00596882	133616	120432	0.00512572	133616	69008
	10,000	0.00601882	133616	120481	0.00532571	133616	69008
	10,000	0.00603514	133616	120522	0.00512642	133616	69008
	10,000	0.00572136	133616	120450	0.00526152	133616	69008
	10,000	0.00602418	133616	120431	0.00565264	133616	69008
	10,000	0.00602421	133616	120397	0.00572451	133616	69008
	10,000	0.00596271	133616	120412	0.00512752	133616	69008
	10,000	0.00599126	133616	120498	0.00525679	133616	69008
	10,000	0.00591389	133616	120548	0.00546142	133616	69008
Ave:	10,000	0.00595548	133616	120463.4	0.00533278	133616	69008
	1,000	0.00018701	9976	8698	0.00013581	9976	5044
	1,000	0.00018802	9976	8726	0.00013452	9976	5044
	1,000	0.00018718	9976	8742	0.00013263	9976	5044
	1,000	0.00018692	9976	8737	0.00013503	9976	5044
	1,000	0.00018763	9976	8708	0.00013608	9976	5044
	1,000	0.00018677	9976	8718	0.00013504	9976	5044
	1,000	0.00019581	9976	8705	0.00013475	9976	5044
	1,000	0.00018895	9976	8736	0.00013524	9976	5044
	1,000	0.00018792	9976	8716	0.0001352	9976	5044
	1,000	0.00016377	9976	8717	0.00011737	9976	5044
Ave:	1,000	0.000186	9976	8720.3	0.00013317	9976	5044


Sort Time (Average)
y = 6E-07x - 0.0003
R² = 0.9999
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
20,000
40,000
60,000
80,000
100,000
120,000


Sorted Time (Best Case)
y = 5E-07x - 0.0002
R² = 0.9989
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
20,000
40,000
60,000
80,000
100,000
120,000

	Input Size	Sort Time	Swaps	Comparisons	Sorted Time	Sorted Swap	Sorted Comp
Insert Sort	100,000	13.1183	2497194464	2497294463	0.00028697	4955	104954
	100,000	13.2463	2504646848	2504746847	0.00029515	5027	105026
	100,000	13.2373	2504545037	2504645036	0.00029656	5108	105107
	100,000	13.2423	2489668038	2489768037	0.00033349	5092	105091
	100,000	13.2315	2498468813	2498568812	0.00029705	4953	104952
	100,000	13.2906	2505003782	2505103781	0.00029612	5106	105105
	100,000	13.2356	2495936459	2496036458	0.00029309	4956	104955
	100,000	13.3152	2501577389	2501677388	0.00029341	4939	104938
	100,000	13.2695	2500129349	2500229348	0.00028711	5001	105000
	100,000	13.2964	2500509377	2500609376	0.00029707	5003	105002
	100,000	13.2483	2499767956	2499867955	0.0002976	5014	105013
	10,000	0.132468	25011614	25021613	2.20E-05	40	10039
	10,000	0.133605	25203993	25213992	2.21E-05	46	10045
	10,000	0.133425	25111445	25121444	2.21E-05	49	10048
	10,000	0.131221	24955222	24965221	2.24E-05	51	10050
	10,000	0.131161	24915322	24925321	2.21E-05	47	10046
	10,000	0.131343	24880370	24890369	2.21E-05	50	10049
	10,000	0.131243	24804126	24814125	2.23E-05	42	10041
	10,000	0.131563	24942355	24952354	2.22E-05	40	10039
	10,000	0.133736	25222302	25232301	2.23E-05	45	10044
	10,000	0.132789	25151048	25161047	2.22E-05	51	10050
	10,000	0.1322554	25019779.7	25029778.7	2.22E-05	46.1	10045.1
	1,000	0.001516	260945	261944	2.23E-06	1	1000
	1,000	0.0014248	243933	244932	2.15E-06	0	999
	1,000	0.00134266	255322	256321	2.26E-06	1	1000
	1,000	0.0013364	248333	249332	2.31E-06	1	1000
	1,000	0.0013666	248307	249306	2.26E-06	1	1000
	1,000	0.00150211	248998	249997	2.53E-06	0	999
	1,000	0.00130255	238103	239102	2.26E-06	1	1000
	1,000	0.00138944	252121	253120	2.19E-06	0	999
	1,000	0.00128643	237659	238658	2.22E-06	1	1000
	1,000	0.00126314	239832	240831	2.23E-06	1	1000
	1,000	0.00137301	247355.3	248354.3	2.26E-06	0.7	999.7


Sort Time (Average)
y = 1E-09x² - 3E-08x + 8E-05
R² = 1
14
12
10
8
6
4
2
0
0
20,000
40,000
60,000
80,000
100,000
120,000


Sorted Time (Best Case
y = 3E-09x - 4E-06
R² = 0.9969
0.0004
0.00035
0.0003
0.00025
0.0002
0.00015
0.0001
0.00005
0
-0.00005
0
20,000
40,000
60,000
80,000
100,000
120,000

Ethical Discussion: Through this project I learned that sorting takes a lot more time than I expected. Fast sorting algorithims are extremely important for data analysis. For example, stock analysis would likely require fast data analysis. In addition, much of our lives revolve around algorithims. In the future, AI and Machine learning algorithims will become more influential in our life, which will raise the need for efficient coding.