/* Test Log
 * Joshua Silva
 * ECE 2230
 * Spring 2024
 * MP3
 *
 *
 * Purpose: Walk through test cases output to prove program functionality. Will only show
 * relevant data to reader.
 *
 * Assumptions: That the reader has read Test Plan, and understands the aim of each test case.
 *
 */

Describing Sorting Trends:

1.
    a. For initial random lists, sorting is generally longer for all methods. Except for Recursive Selection Sort, whose longest sort is for lists who are originally descending and are sorted into ascending order.
    b. My Iterative and Recursive Selection Sort algorithms have similar run times, with Recursive Selection having slightly faster run times.
    c. Lastly, Merge Sort is drastically faster due to its time complexity, O(nlogn) compared to the other sorting methods with O(n^2). This is due to merge sort splitting the last in half, instead of running through the whole list.

2. Insertion Sort shows DRASTICALLY faster results when converting from descending order into ascending order. Taking less than 1 ms when converting descending lists of up to 20000 nodes to ascending order. This is due to Insertion Sort removing the biggest node (LISTPOS_HEAD) from the descending list over and over, causing only 1 run with list_remove. Then using list_insert_sorted, which in this case will sort from smallest to largest for Ascending List, with comp_proc knowing on the 1st run that the next node is larger, continuously causing an insert at LISTPOS_HEAD. In other words, we're constantly removing the beginning of the input list and inserting it at the beginning of our new list, effectively just reversing the input list and avoiding any traveling of either list (as we only need the start of both).

Notes:

- Example Inputs show below directly using the ./geninput size data_method sort_type sort_order being piped into ./lab3. All data in tables and graphs is from individually using this command for each test case.

    - Example inputs show the starting size in the table, but with different initial data methods. All commands will sort initial lists into ascending order as per the Machine Problem instructions.
- testoutput_X files are included to show outputs with a modified "runner.sh" file which is much more effective for testing each sorting method. Data there is similar to data shown below. Decided to put this data into separate .txt files to not clutter this test log.

Bubble Sort:

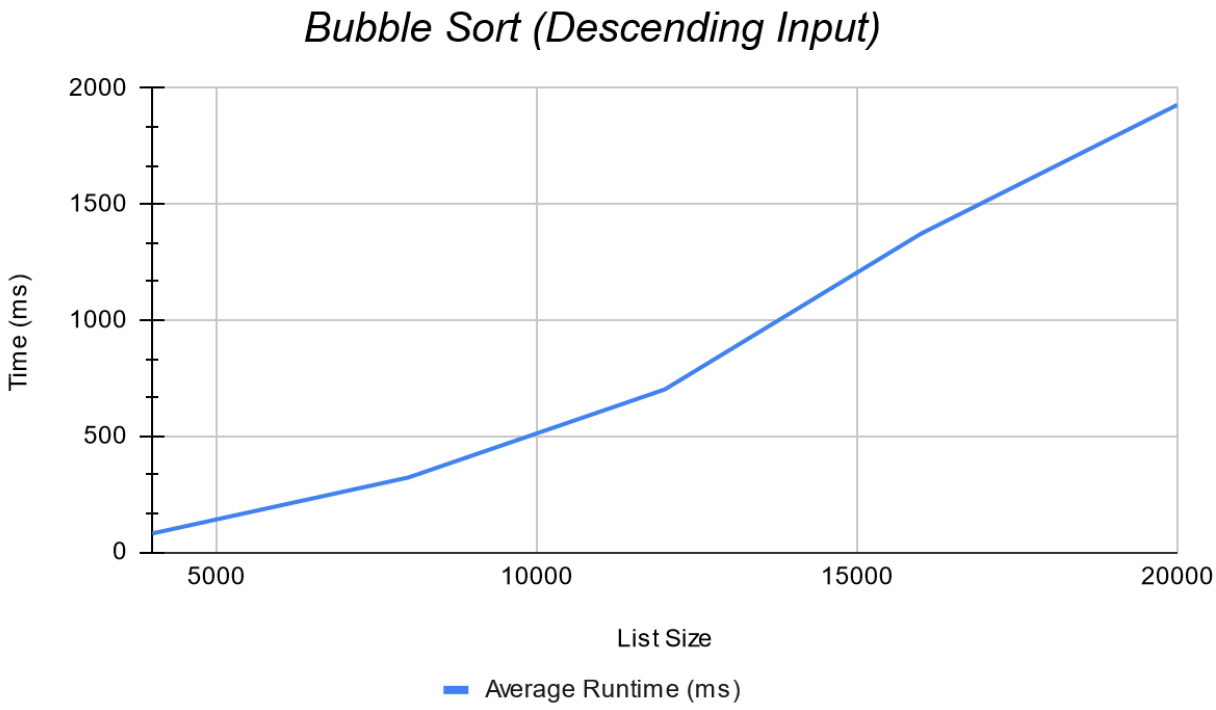
Example Inputs:
./geninput 4000 1 1 2 | ./lab3 10
./geninput 4000 2 1 2 | ./lab3 10
./geninput 4000 3 1 2 | ./lab3 10

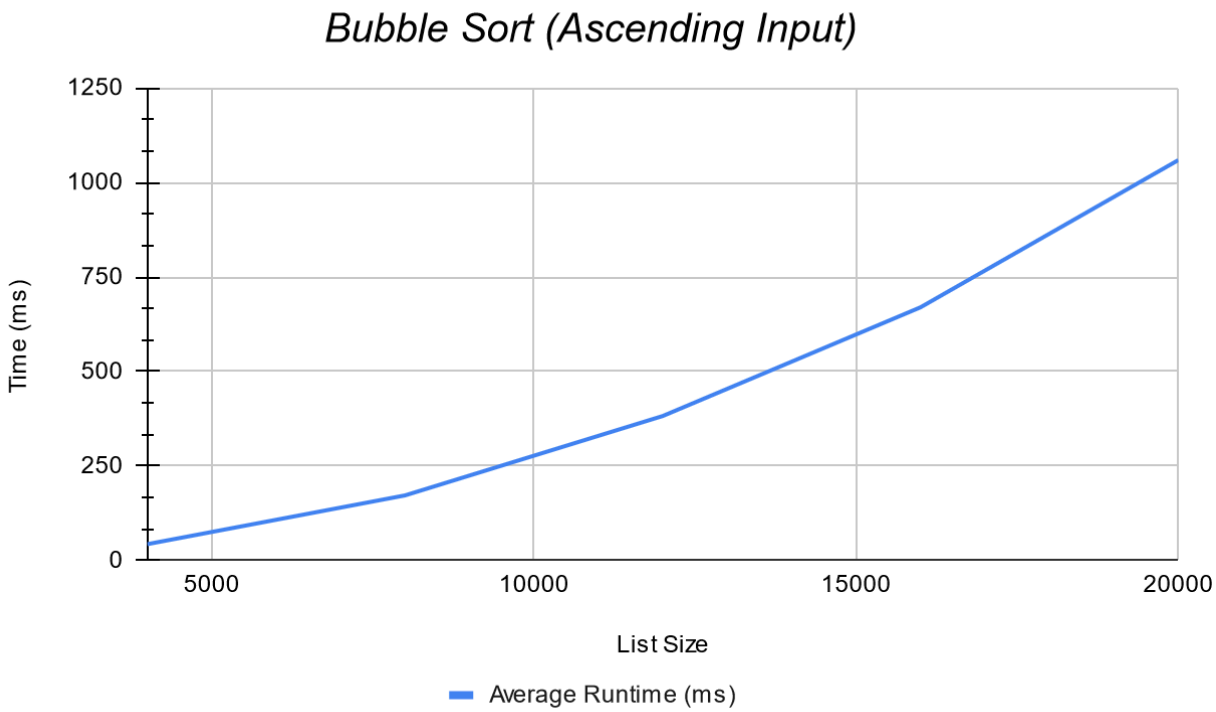
Example Outputs:
…Sorting: n = 4000 time = 89.723000 type = 1 order = 2…
…Sorting: n = 4000 time = 41.907000 type = 1 order = 2…
…Sorting: n = 4000 time = 94.567000 type = 1 order = 2…

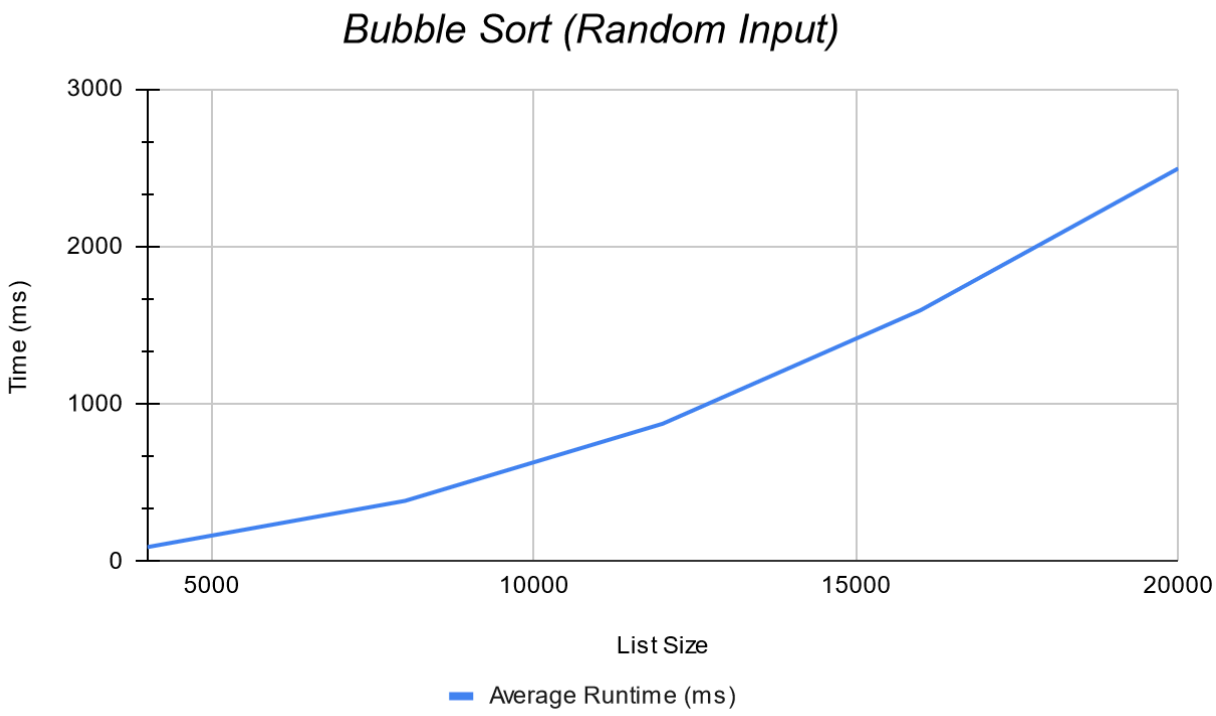
Table:

Bubble Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	4000	89.723	76.575	73.87	80.056
Descending	8000	355.773	312.076	297.349	321.7326667
Descending	12000	669.633	758.735	675.489	701.2856667
Descending	16000	1197.979	1451.14	1470.417	1373.178667
Descending	20000	2014.762	1900.38	1870.767	1928.636333
Ascending	4000	41.907	45.666	42.972	43.515
Ascending	8000	171.692	175.754	169.928	172.458
Ascending	12000	382.067	384.577	380.76	382.468
Ascending	16000	669.255	669.931	670.466	669.884
Ascending	20000	1059.673	1062.221	1055.401	1059.098333
Random	4000	94.567	90.656	90.865	92.02933333
Random	8000	382.494	399.807	374.412	385.571
Random	12000	884.524	862.903	878.599	875.342
Random	16000	1565.874	1585.019	1635.071	1595.321333
Random	20000	2482.351	2523.439	2480.308	2495.366

Graphs:


Bubble Sort (Descending Input)
2000
1500
1000
500
0
Time (ms)
5000
10000
15000
20000
List Size
Average Runtime (ms)


Bubble Sort (Ascending Input)
1250
1000
750
500
250
0
Time (ms)
5000
10000
15000
20000
List Size
Average Runtime (ms)



Insertion Sort:

Example Inputs:

./geninput 4000 1 2 2 | ./lab3 10
./geninput 4000 2 2 2 | ./lab3 10
./geninput 4000 3 2 2 | ./lab3 10

Example Outputs:

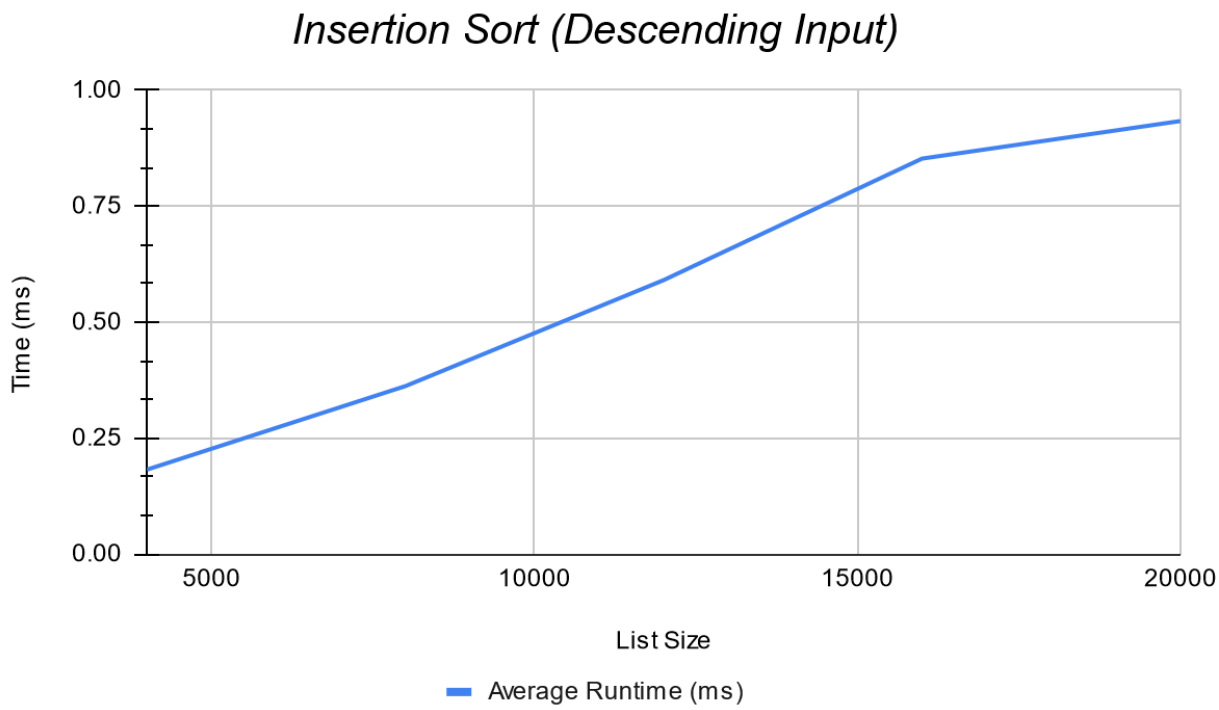
…Sorting: n = 4000 time = 0.179000 type = 1 order = 2…
…Sorting: n = 4000 time = 32.750000 type = 1 order = 2…
…Sorting: n = 4000 time = 52.558000 type = 1 order = 2…

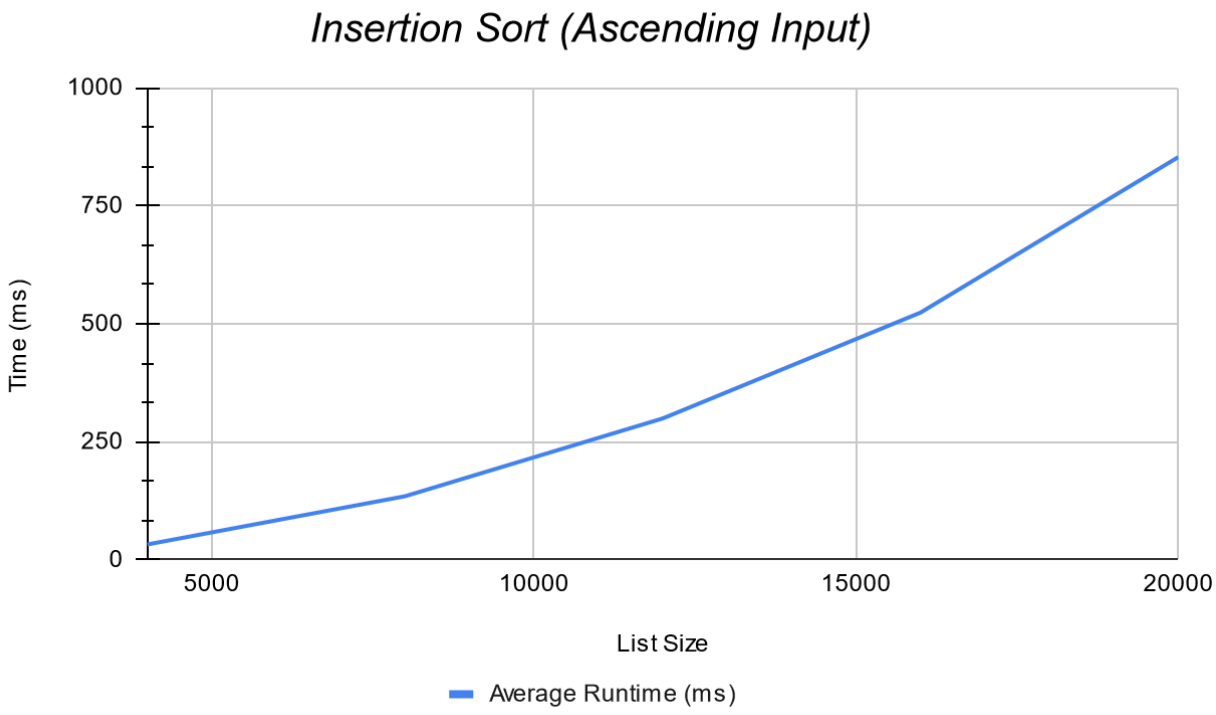
Table:

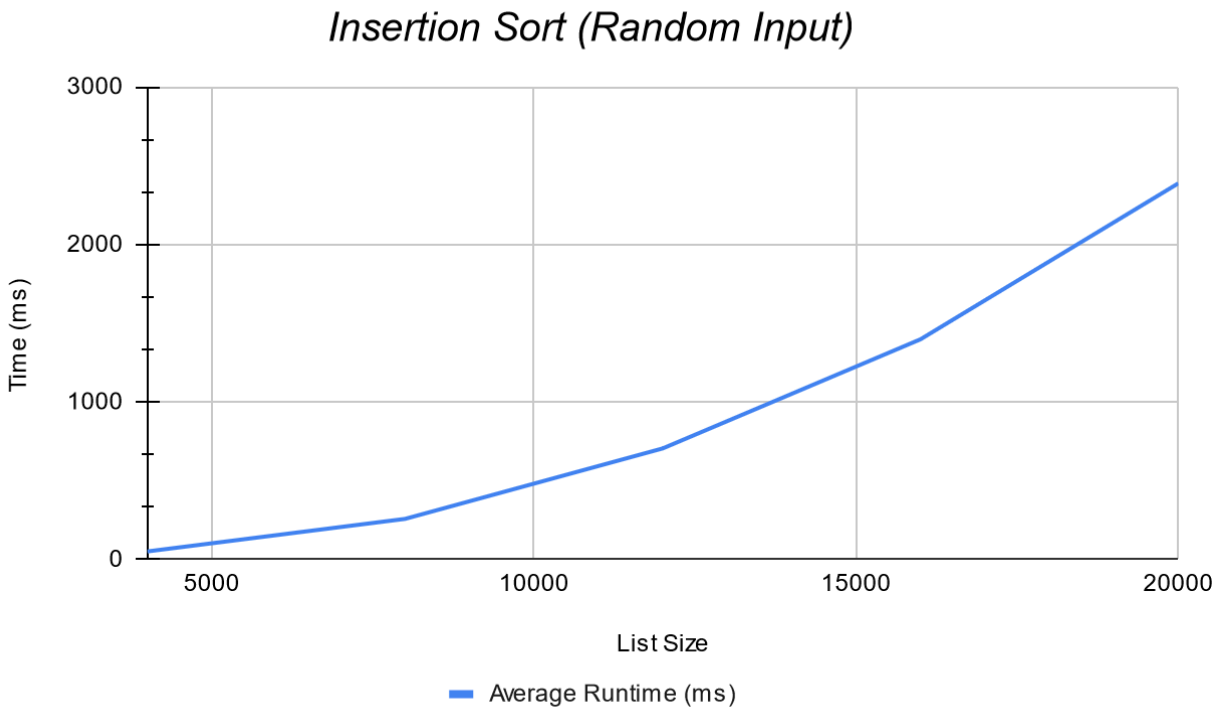
Insertion Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	4000	0.187	0.179	0.18	0.182
Descending	8000	0.356	0.354	0.376	0.362
Descending	12000	0.529	0.534	0.709	0.5906666667

Descending	16000	0.988	0.736	0.834	0.8526666667
Descending	20000	0.917	0.906	0.978	0.9336666667
Ascending	4000	32.75	32.617	32.51	32.62566667
Ascending	8000	136.64	134.895	132.747	134.7606667
Ascending	12000	300.736	303.688	294.389	299.6043333
Ascending	16000	521.258	530.205	520.095	523.8526667
Ascending	20000	819.852	911.067	828.753	853.224
Random	4000	52.588	53.056	50.323	51.989
Random	8000	247.776	257.403	270.388	258.5223333
Random	12000	729.679	698.633	689.284	705.8653333
Random	16000	1405.846	1386.548	1402.955	1398.449667
Random	20000	2345.054	2407.55	2411.63	2388.078

Graphs:







Recursive Selection Sort:

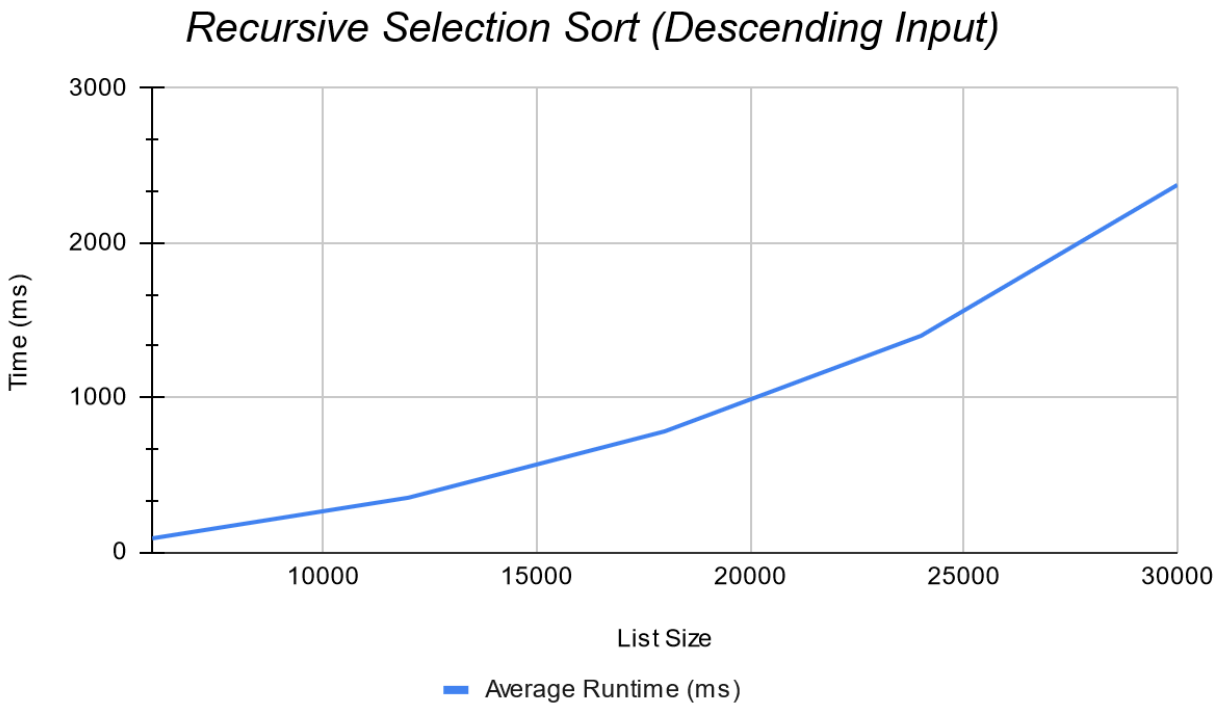
Example Inputs:
./geninput 6000 1 3 2 | ./lab3 10
./geninput 6000 2 3 2 | ./lab3 10
./geninput 6000 3 3 2 | ./lab3 10

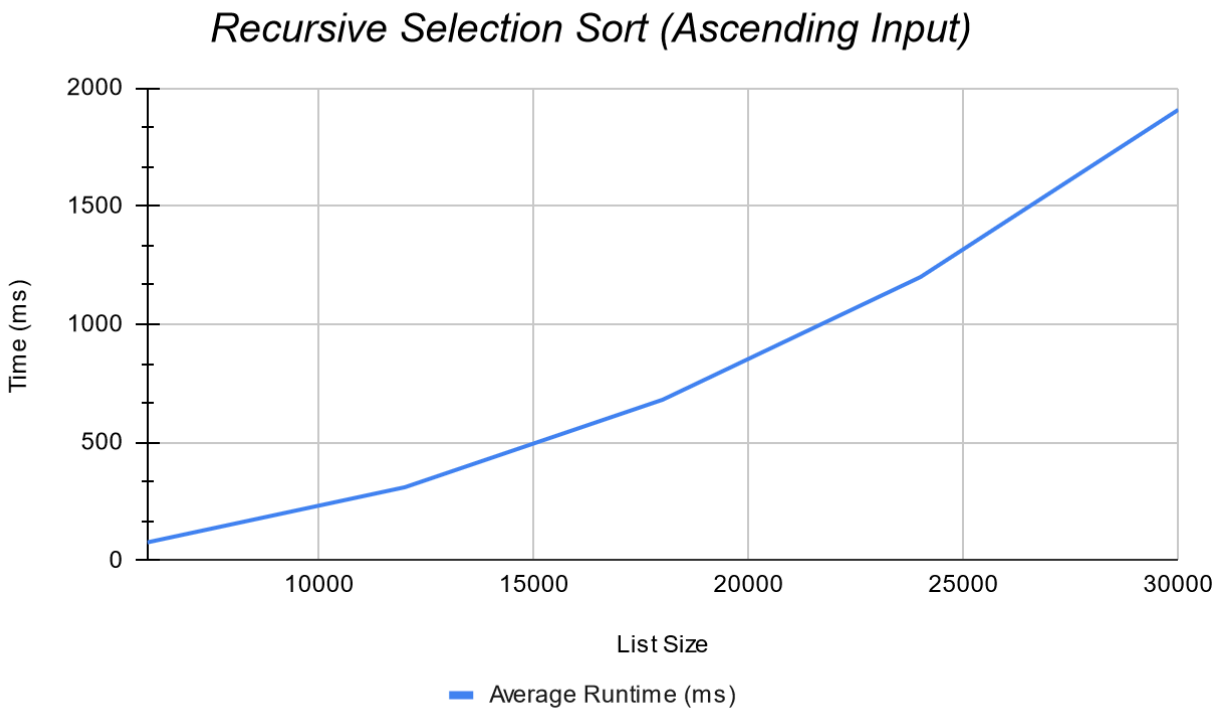
Example Outputs:
…Sorting: n = 6000 time = 88.339000 type = 1 order = 2…
…Sorting: n = 6000 time = 77.738000 type = 1 order = 2…
…Sorting: n = 6000 time = 79.924000 type = 1 order = 2…

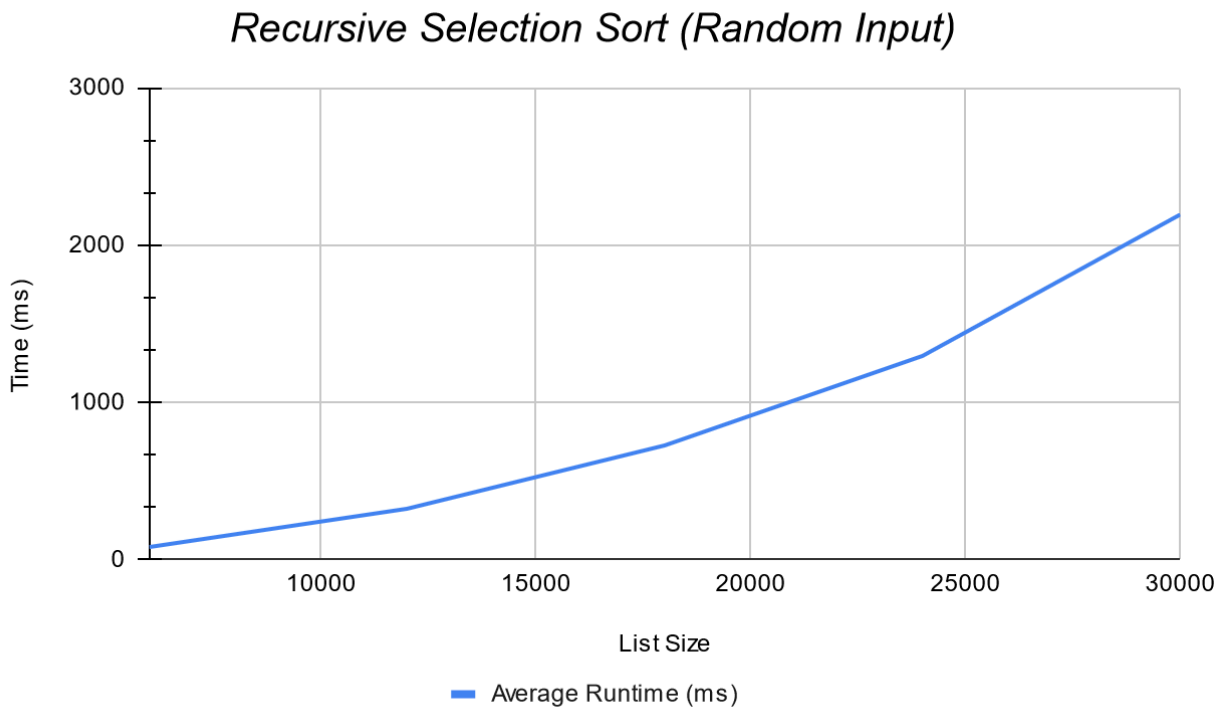
Table:

Recursive Selection Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	6000	88.339	89.447	87.622	88.46933333
Descending	12000	355.818	351.049	348.325	351.7306667
Descending	18000	782.017	779.481	781.186	780.8946667
Descending	24000	1393.584	1404.088	1399.518	1399.063333
Descending	30000	2458.228	2378.477	2287.745	2374.816667
Ascending	6000	77.738	76.984	76.818	77.18
Ascending	12000	307.601	314.991	309.321	310.6376667
Ascending	18000	677.711	682.978	682.472	681.0536667
Ascending	24000	1201.752	1191.448	1206.911	1200.037
Ascending	30000	1909.683	1871.836	1941.53	1907.683
Random	6000	79.924	83.593	80.992	81.503
Random	12000	328.994	320.62	324.752	324.7886667
Random	18000	718.964	743.267	719.775	727.3353333
Random	24000	1294.376	1287.104	1306.764	1296.081333
Random	30000	2232.217	2149.336	2202.132	2194.561667

Graphs:


Recursive Selection Sort (Descending Input)
3000
2000
1000
0
Time (ms)
10000
15000
20000
25000
30000
List Size
Average Runtime (ms)


Recursive Selection Sort (Ascending Input)
2000
1500
1000
500
0
Time (ms)
10000
15000
20000
25000
30000
List Size
Average Runtime (ms)



Iterative Selection Sort:

Example Inputs:

./geninput 6000 1 4 2 | ./lab3 10
./geninput 6000 2 4 2 | ./lab3 10
./geninput 6000 3 4 2 | ./lab3 10

Example Outputs:

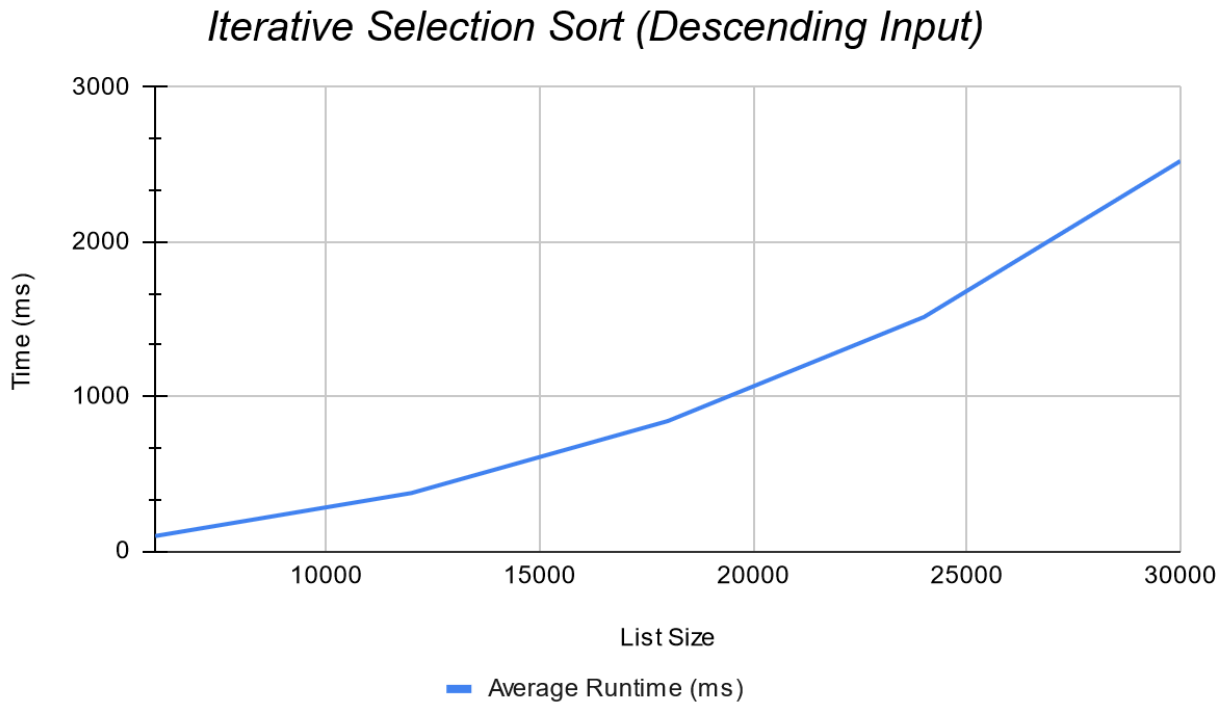
…Sorting: n = 6000 time = 97.392000 type = 1 order = 2…
…Sorting: n = 6000 time = 82.383000 type = 1 order = 2…
…Sorting: n = 6000 time = 85.046000 type = 1 order = 2…

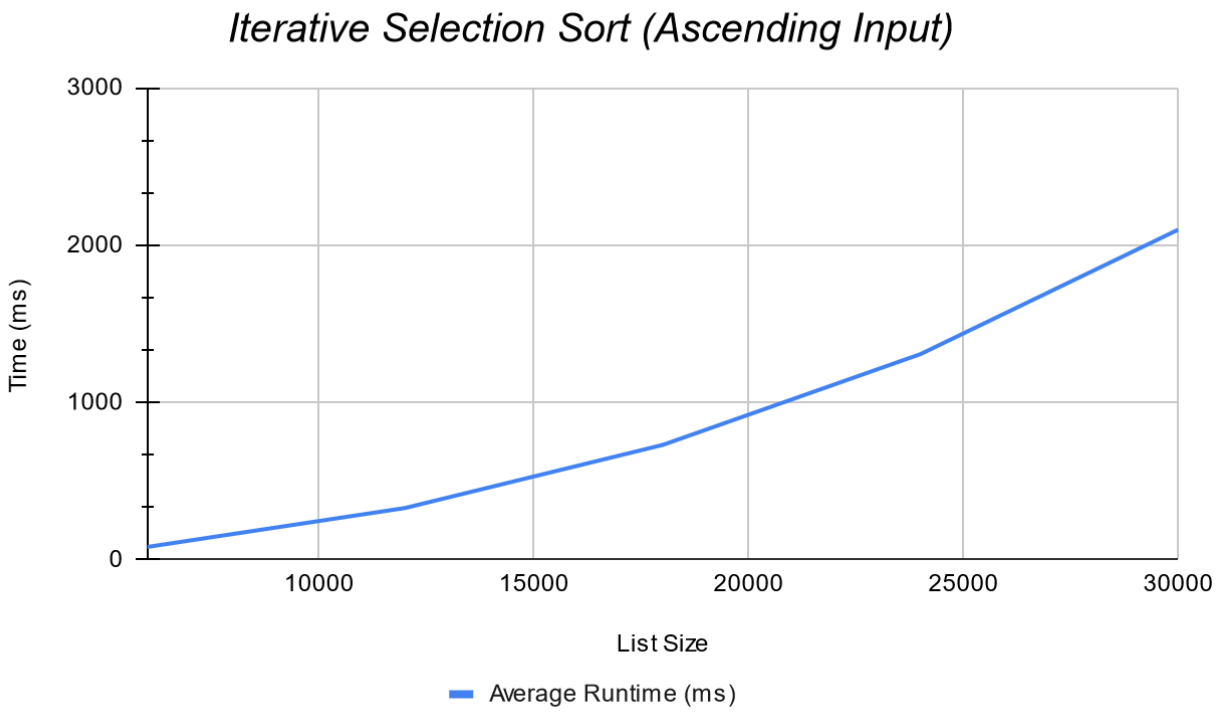
Table:

Iterative Selection Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	6000	97.392	97.397	96.097	96.962
Descending	12000	376.367	379.346	370.389	375.3673333
Descending	18000	843.12	839.907	839.64	840.889
Descending	24000	1533.723	1515.065	1493.717	1514.168333

Descending	30000	2573.328	2477.096	2518.346	2522.923333
Ascending	6000	82.383	82.602	81.378	82.121
Ascending	12000	331.872	326.389	330.14	329.467
Ascending	18000	732.935	726.967	731.664	730.522
Ascending	24000	1307.063	1305.247	1310.659	1307.656333
Ascending	30000	2077.77	2090.832	2126.968	2098.523333
Random	6000	85.046	83.899	84.35	84.43166667
Random	12000	347.383	345.193	352.954	348.51
Random	18000	794.467	805.432	782.466	794.1216667
Random	24000	1465.938	1439.157	1446.176	1450.423667
Random	30000	2360.367	2349.402	2362.61	2357.459667

Graphs:




Iterative Selection Sort (Ascending Input)
3000
2000
1000
0
Time (ms)
10000
15000
20000
25000
30000
List Size
Average Runtime (ms)


Iterative Selection Sort (Random Input)
3000
2000
1000
0
Time (ms)
10000
15000
20000
25000
30000
List Size
Average Runtime (ms)

Merge Sort:

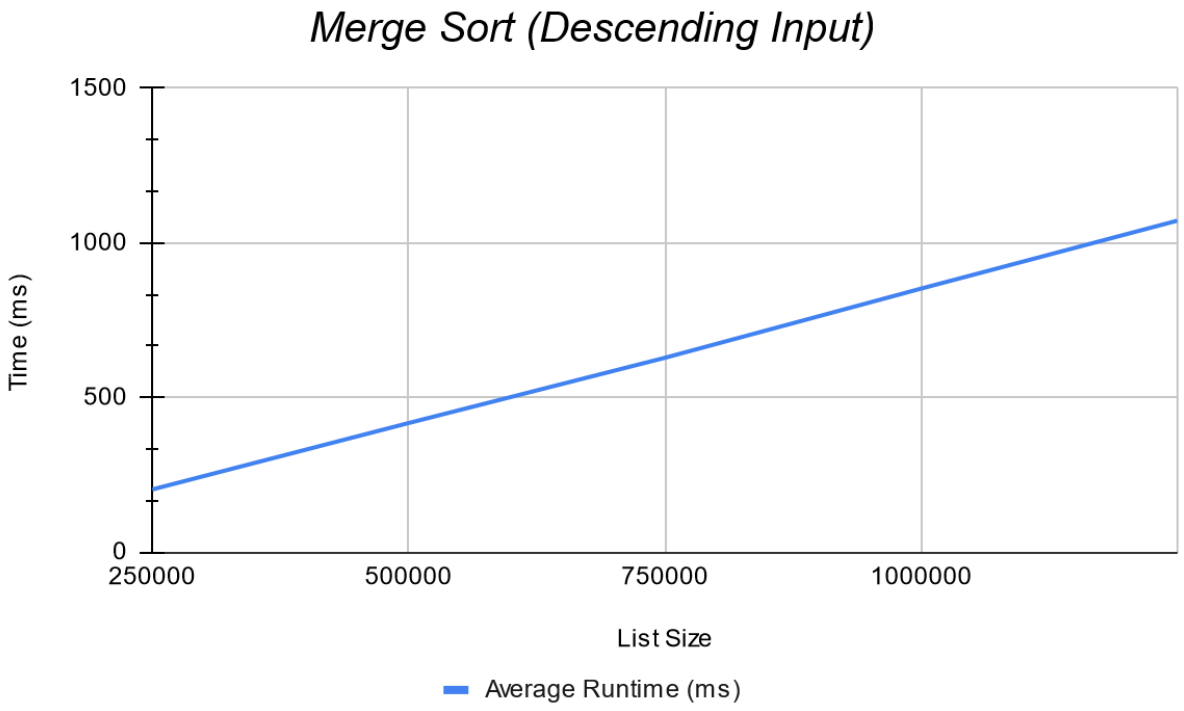
Example Inputs:
./geninput 250000 1 5 2 | ./lab3 10
./geninput 250000 2 5 2 | ./lab3 10
./geninput 250000 3 5 2 | ./lab3 10

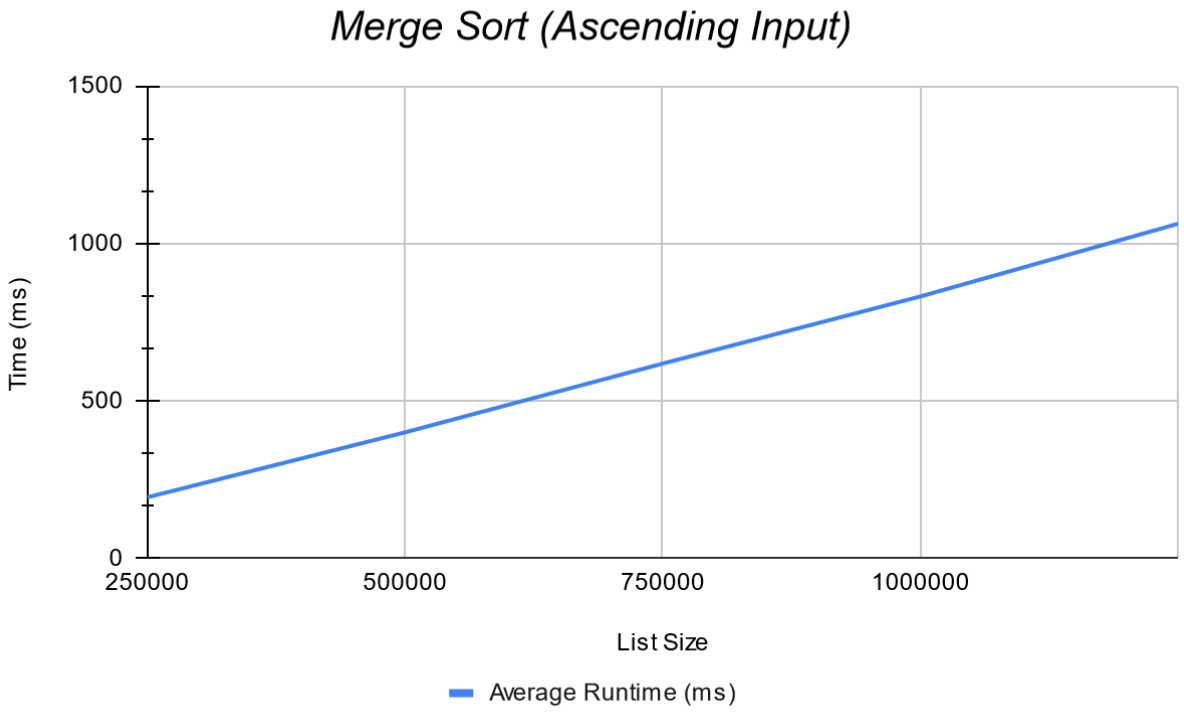
Example Outputs:
…Sorting: n = 250000 time = 202.714000 type = 1 order = 2…
…Sorting: n = 250000 time = 196.292000 type = 1 order = 2…
…Sorting: n = 250000 time = 422.747000 type = 1 order = 2…

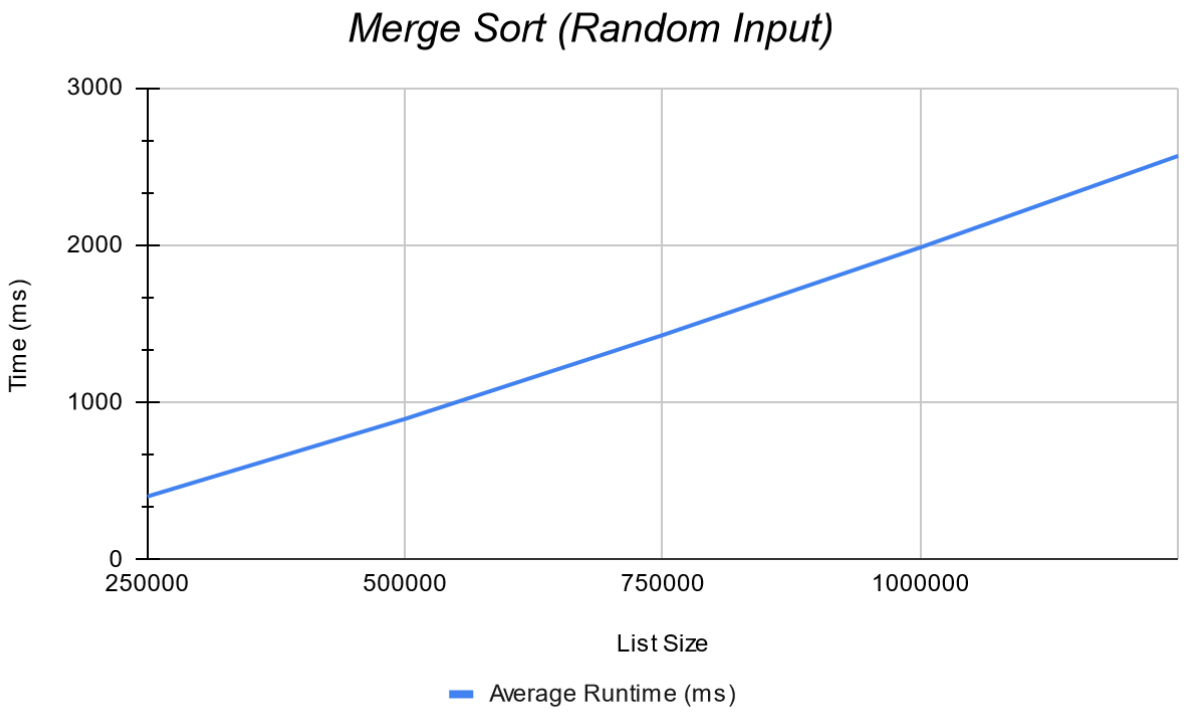
Table:

Merge Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	250000	202.714	200.351	204.24	202.435
Descending	500000	406.054	434.422	411.04	417.172
Descending	750000	623.04	634.972	626.609	628.207
Descending	1000000	869.363	848.7	838.22	852.0943333
Descending	1250000	1071.951	1067.675	1076.034	1071.886667
Ascending	250000	196.292	193.17	194.031	194.4976667
Ascending	500000	401.309	402.877	397.813	400.6663333
Ascending	750000	621.561	615.335	620.209	619.035
Ascending	1000000	830.775	832.439	832.857	832.0236667
Ascending	1250000	1066.366	1060.707	1061.449	1062.840667
Random	250000	422.747	388.756	395.921	402.4746667
Random	500000	881.378	910.74	894.475	895.531
Random	750000	1440.351	1415.318	1429.79	1428.486333
Random	1000000	1975.789	1981.292	2000.315	1985.798667
Random	1250000	2562.251	2563.206	2577.162	2567.539667

Graphs:


Merge Sort (Descending Input)
1500
1000
500
0
Time (ms)
250000
500000
750000
1000000
List Size
Average Runtime (ms)


Merge Sort (Ascending Input)
1500
1000
500
0
Time (ms)
250000
500000
750000
1000000
List Size
Average Runtime (ms)



BONUS:

Change in Time Complexity?:

- No, despite the sorts now running faster. The time complexity of each sort remains the same as the code isn't being fundamentally changed, being reflected in the graphs below. The optimization seemed to have less effect on Merge Sort than the other sort functions.

Notes:

- Used -O3 in the compiler

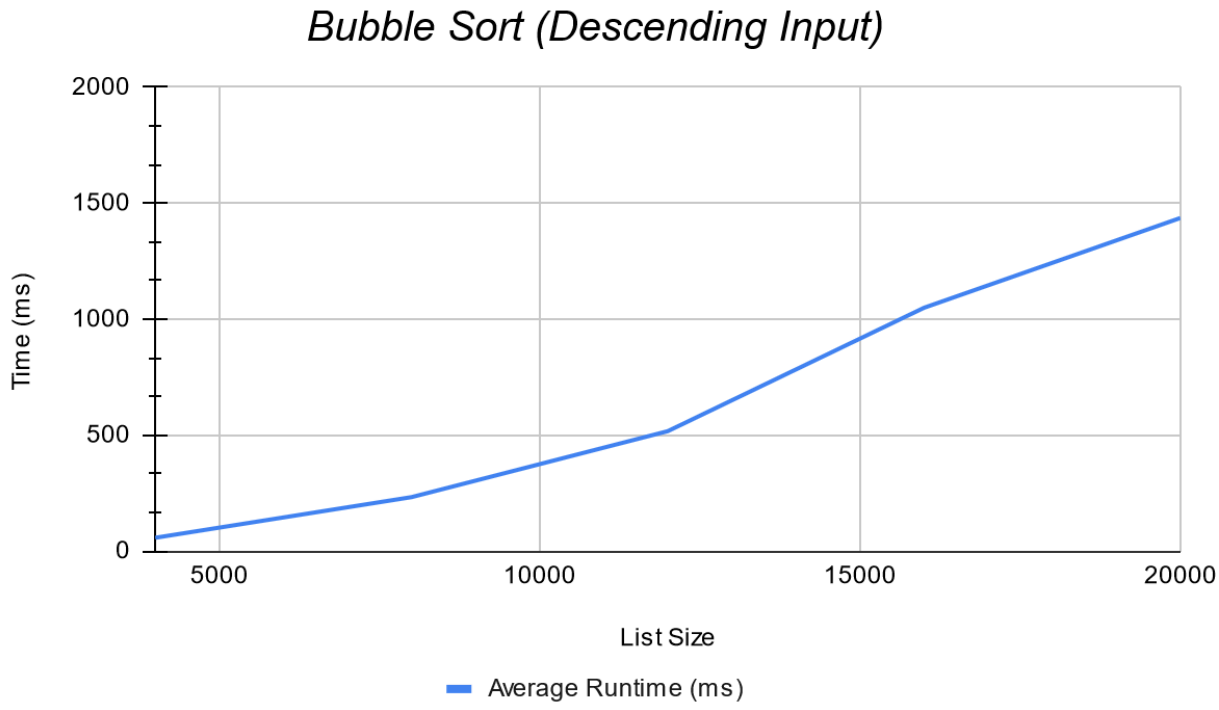
Bubble Sort:

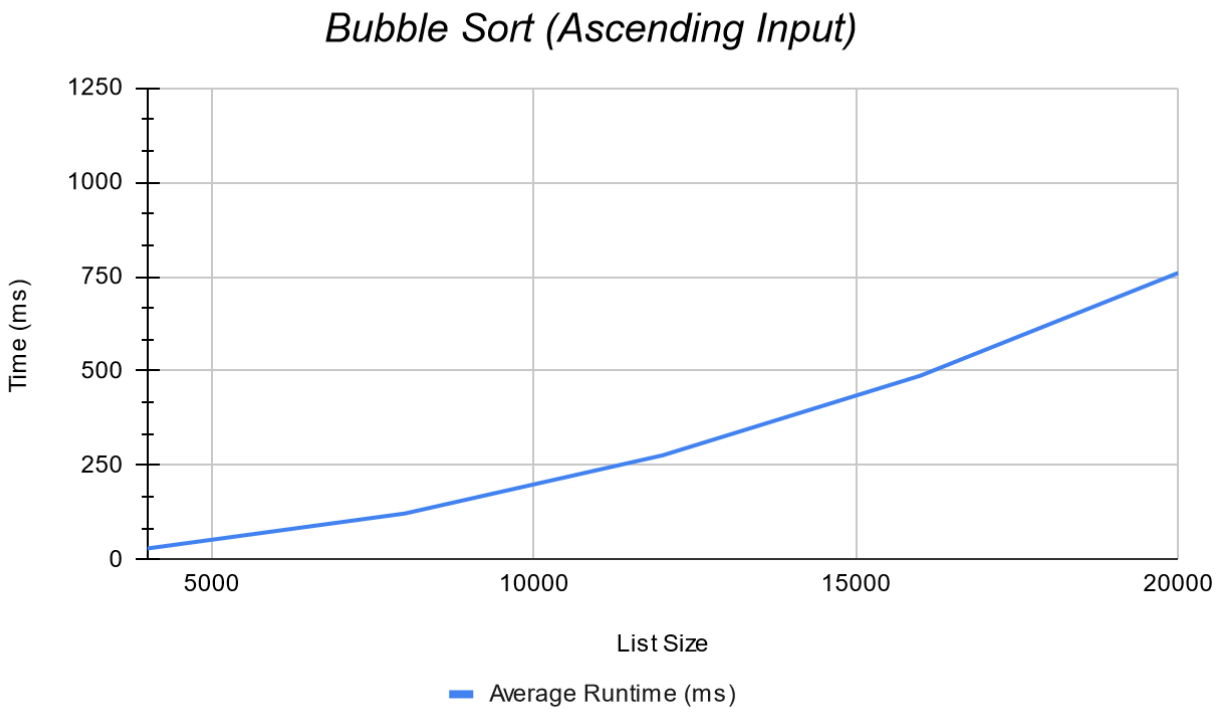
Table:

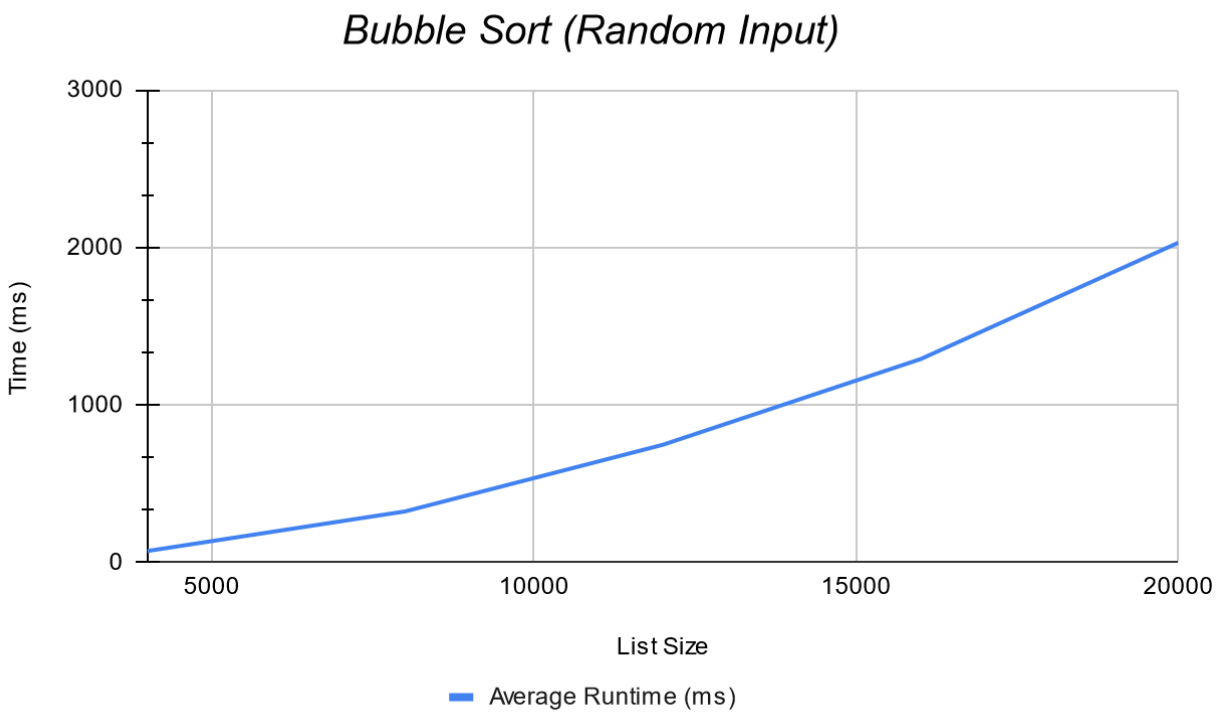
Bubble Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	4000	56.333	57.169	58.251	57.251
Descending	8000	231.994	234.937	230.585	232.5053333
Descending	12000	517.392	514.049	519.461	516.9673333
Descending	16000	1319.308	915.755	912.72	1049.261

Descending	20000	1428.932	1437.429	1442.74	1436.367
Ascending	4000	30.226	30.005	30.016	30.08233333
Ascending	8000	122.558	123.294	122.633	122.8283333
Ascending	12000	277.414	279.987	273.322	276.9076667
Ascending	16000	484.583	490.868	487.988	487.813
Ascending	20000	757.981	757.485	764.194	759.8866667
Random	4000	73.718	73.01	73.913	73.547
Random	8000	354.469	311.092	309.213	324.9246667
Random	12000	826.271	708.383	708.76	747.8046667
Random	16000	1295.594	1273.881	1305.147	1291.540667
Random	20000	2023.988	2024.652	2042.35	2030.33

Graphs:




Bubble Sort (Ascending Input)
1250
1000
750
500
250
0
Time (ms)
5000
10000
15000
20000
List Size
Average Runtime (ms)

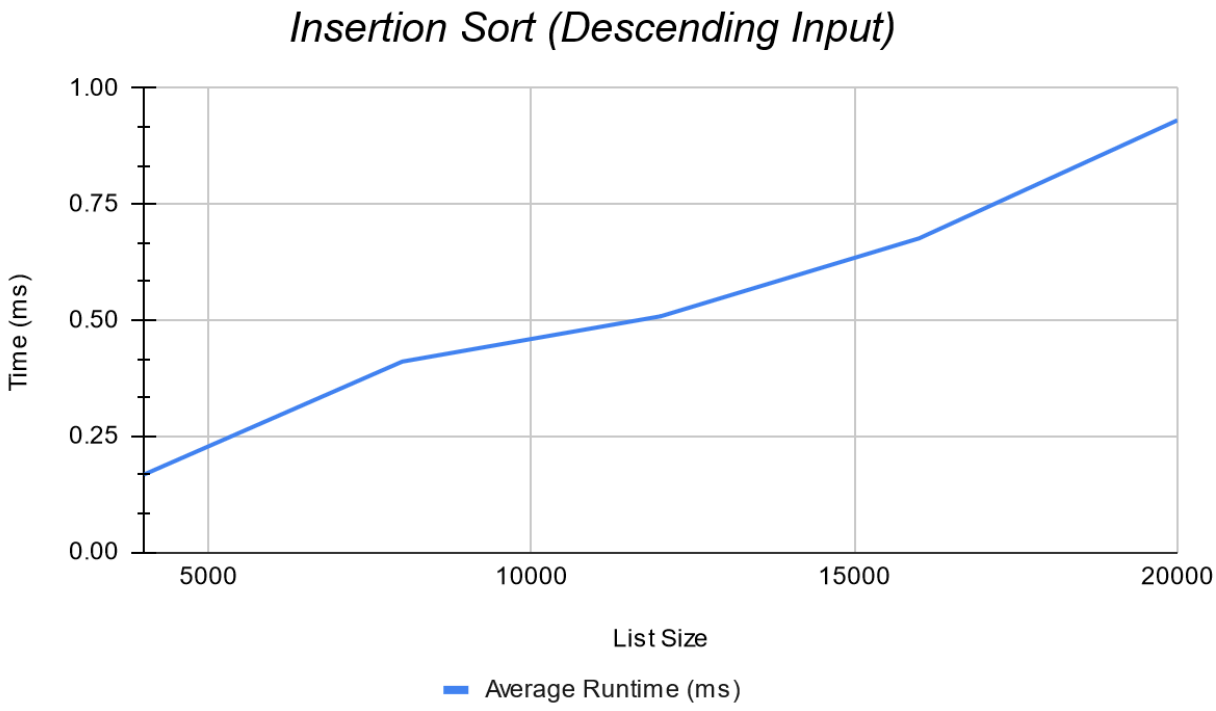

Bubble Sort (Random Input)
3000
2000
1000
0
Time (ms)
5000
10000
15000
20000
List Size
Average Runtime (ms)

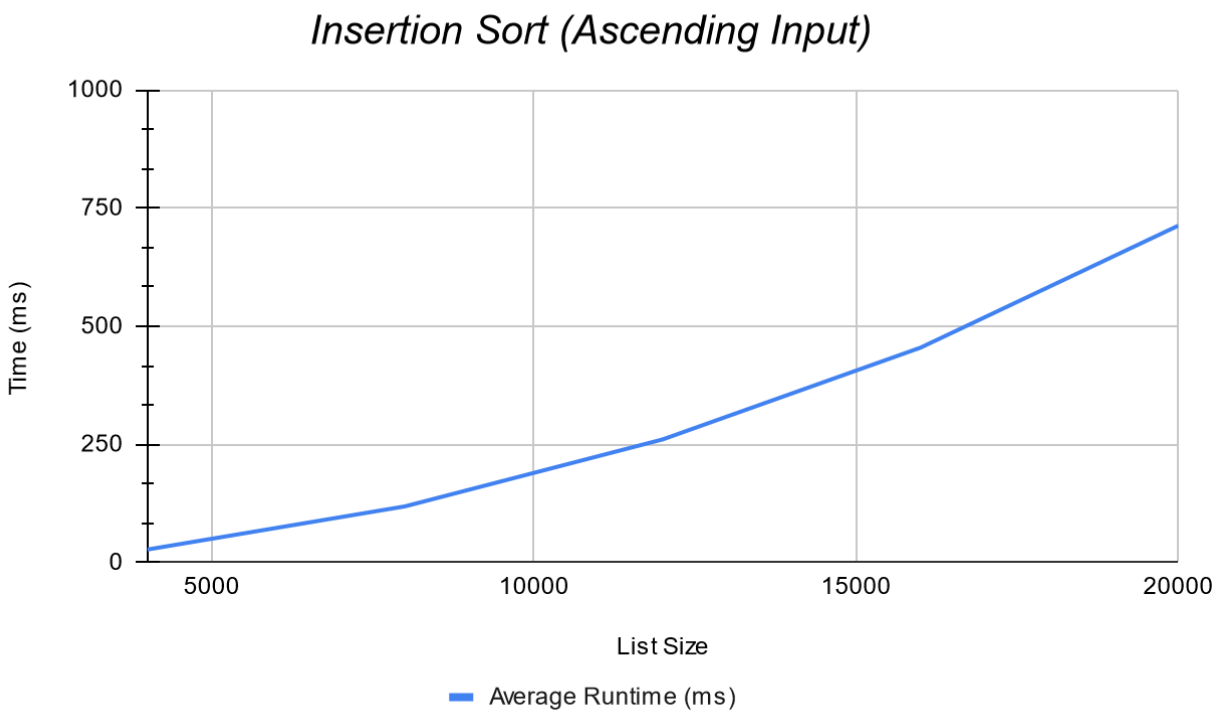
Insertion Sort:

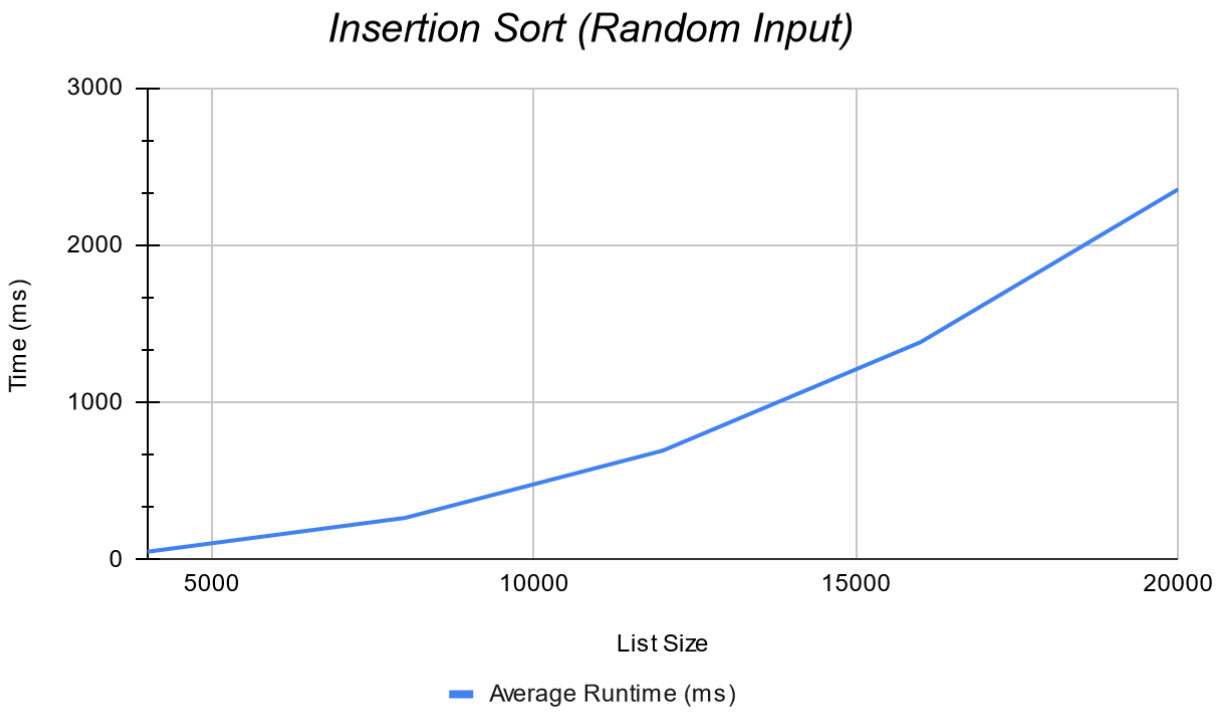
Table:

Insertion Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	4000	0.167	0.169	0.165	0.167
Descending	8000	0.404	0.501	0.327	0.4106666667
Descending	12000	0.52	0.512	0.493	0.5083333333
Descending	16000	0.709	0.667	0.653	0.6763333333
Descending	20000	0.838	1.124	0.831	0.931
Ascending	4000	27.629	27.829	27.508	27.65533333
Ascending	8000	121.518	117.155	118.439	119.0373333
Ascending	12000	258.551	262.032	261.54	260.7076667
Ascending	16000	452.678	452.059	459.692	454.8096667
Ascending	20000	719.524	708.339	710.812	712.8916667
Random	4000	52.427	51.635	50.512	51.52466667
Random	8000	268.853	258.401	272.815	266.6896667
Random	12000	694.398	700.103	688.641	694.3806667
Random	16000	1393.052	1377.58	1380.045	1383.559
Random	20000	2371.642	2345.13	2349.397	2355.389667

Graphs:


Insertion Sort (Descending Input)
1.00
0.75
0.50
0.25
0.00
Time (ms)
5000
10000
15000
20000
List Size
Average Runtime (ms)


Insertion Sort (Ascending Input)
1000
750
500
250
0
Time (ms)
5000
10000
15000
20000
List Size
Average Runtime (ms)



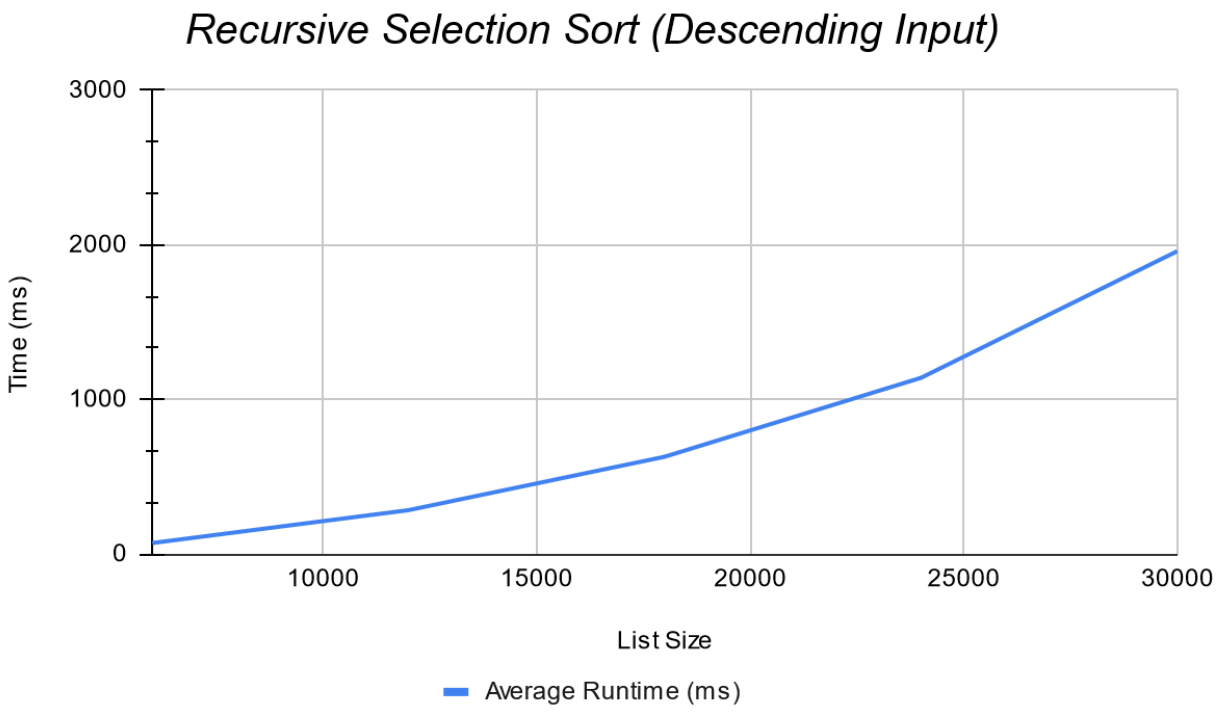
Recursive Selection Sort:

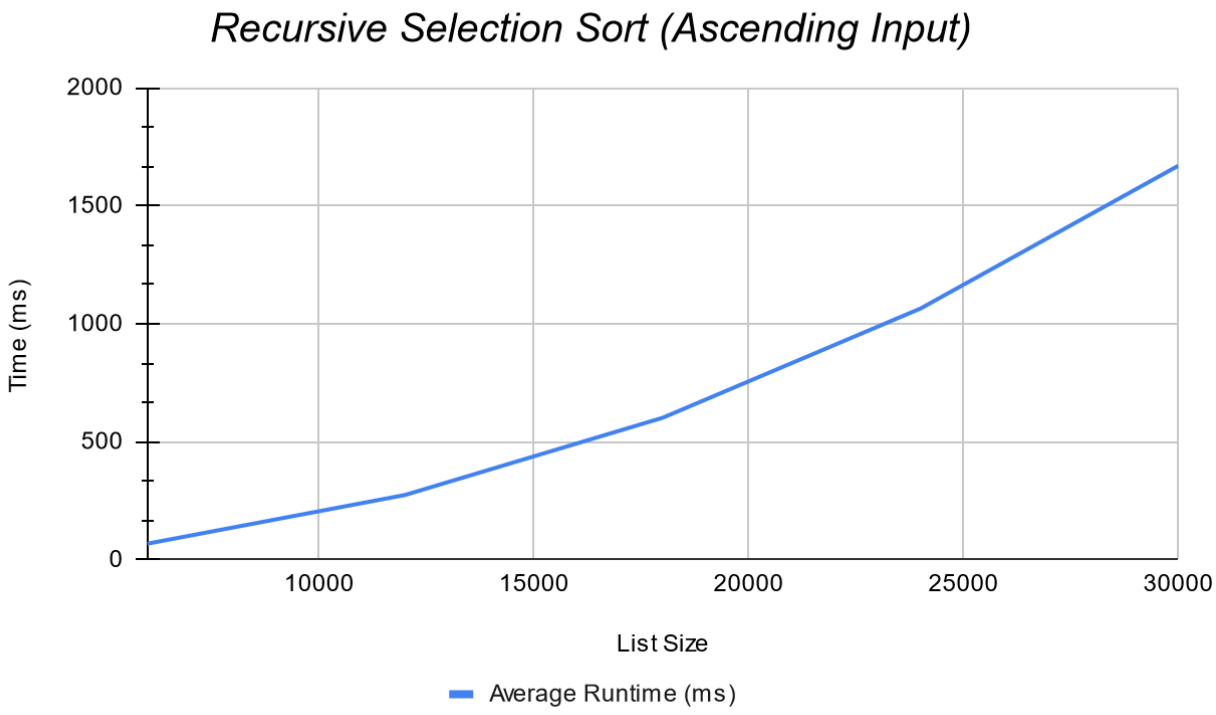
Table:

Recursive Selection Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	6000	69.876	73.572	70.467	71.305
Descending	12000	278.298	286.278	287.137	283.9043333
Descending	18000	637.675	629.77	621.598	629.681
Descending	24000	1180.203	1123.152	1117.037	1140.130667
Descending	30000	1915.768	2018.107	1944.864	1959.579667
Ascending	6000	68.51	68.143	68.654	68.43566667
Ascending	12000	275.812	272.515	276.32	274.8823333
Ascending	18000	598.115	602.154	606.299	602.1893333
Ascending	24000	1075.867	1061.969	1056.804	1064.88
Ascending	30000	1652.623	1680.769	1676.002	1669.798
Random	6000	72.749	70.873	73.282	72.30133333
Random	12000	293.271	288.145	283.233	288.2163333
Random	18000	636.476	620.535	626.634	627.8816667
Random	24000	1129.482	1101.159	1104.155	1111.598667

Random	30000	1797.952	1784.093	1917.437	1833.160667

Graphs:




Recursive Selection Sort (Ascending Input)
2000
1500
1000
500
0
Time (ms)
10000
15000
20000
25000
30000
List Size
Average Runtime (ms)

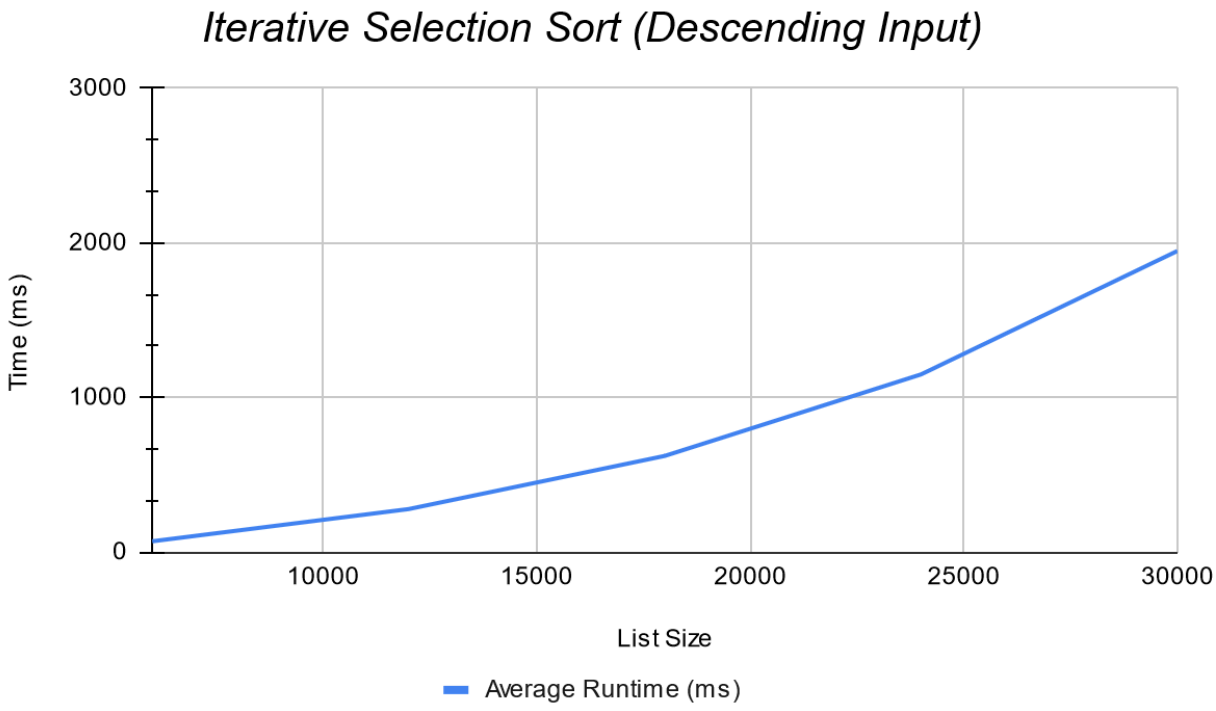

Recursive Selection Sort (Random Input)
3000
2000
1000
0
Time (ms)
10000
15000
20000
25000
30000
List Size
Average Runtime (ms)

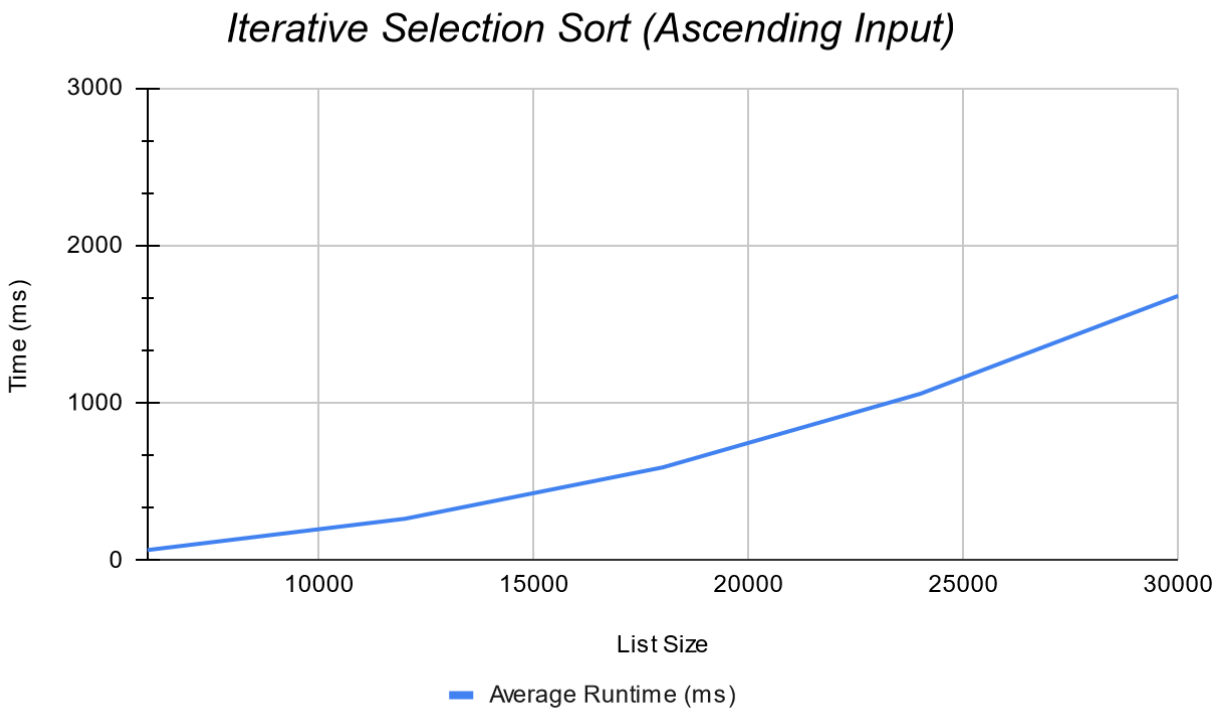
Iterative Selection Sort:

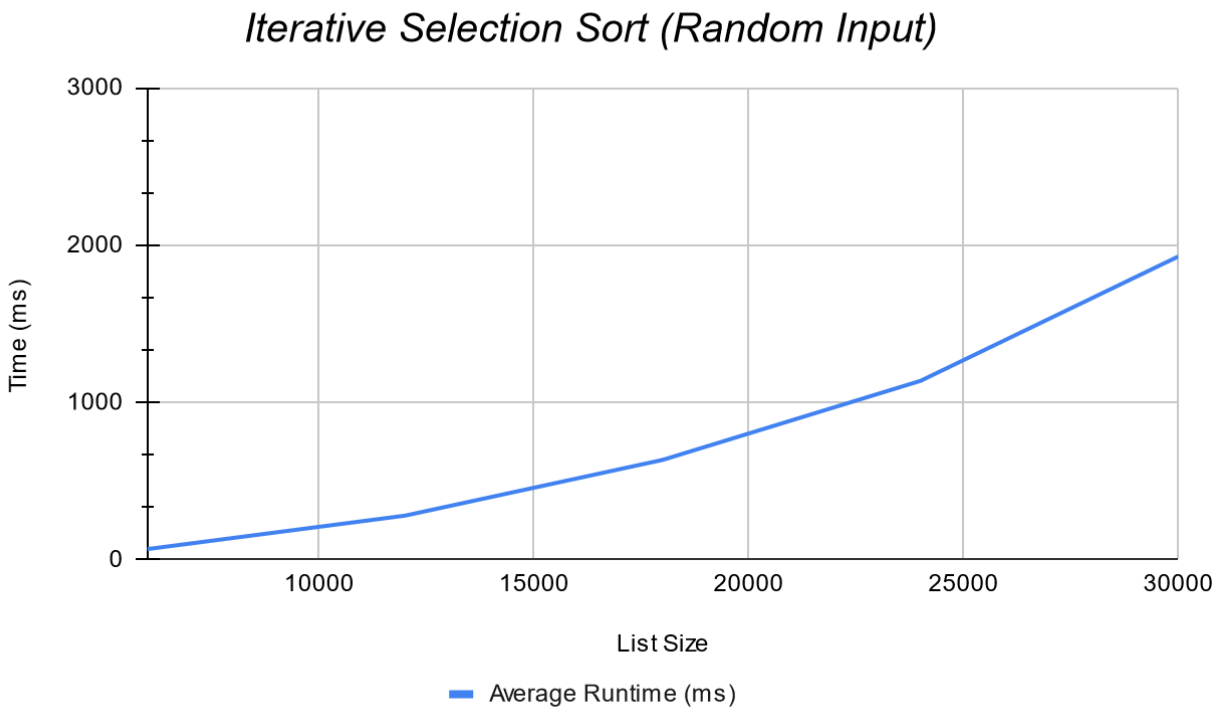
Table:

Iterative Selection Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	6000	69.411	69.65	69.816	69.62566667
Descending	12000	275.051	275.903	282.522	277.8253333
Descending	18000	627.651	617.7	620.557	621.9693333
Descending	24000	1169.631	1128.871	1149.225	1149.242333
Descending	30000	1900.783	1974.561	1967.772	1947.705333
Ascending	6000	69.277	66.219	65.338	66.94466667
Ascending	12000	266.319	265.574	265.54	265.811
Ascending	18000	590.97	586.916	597.517	591.801
Ascending	24000	1050.899	1070.204	1057.249	1059.450667
Ascending	30000	1688.19	1675.405	1677.926	1680.507
Random	6000	66.951	68.761	69.5	68.404
Random	12000	279.034	282.226	281.759	281.0063333
Random	18000	633.309	635.18	638.194	635.561
Random	24000	1145.459	1125.42	1142.166	1137.681667
Random	30000	1935.472	1961.025	1887.616	1928.037667

Graphs:


Iterative Selection Sort (Descending Input)
3000
2000
1000
0
Time (ms)
10000
15000
20000
25000
30000
List Size
Average Runtime (ms)


Iterative Selection Sort (Ascending Input)
3000
2000
1000
0
Time (ms)
10000
15000
20000
25000
30000
List Size
Average Runtime (ms)



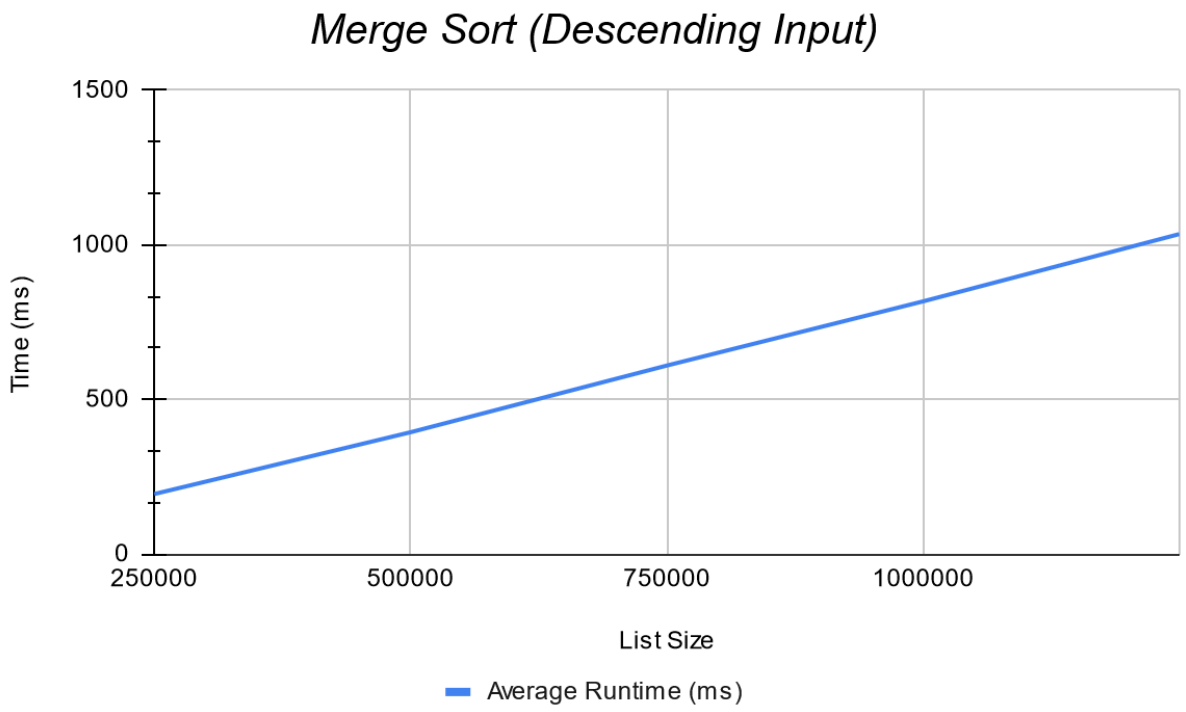
Merge Sort:

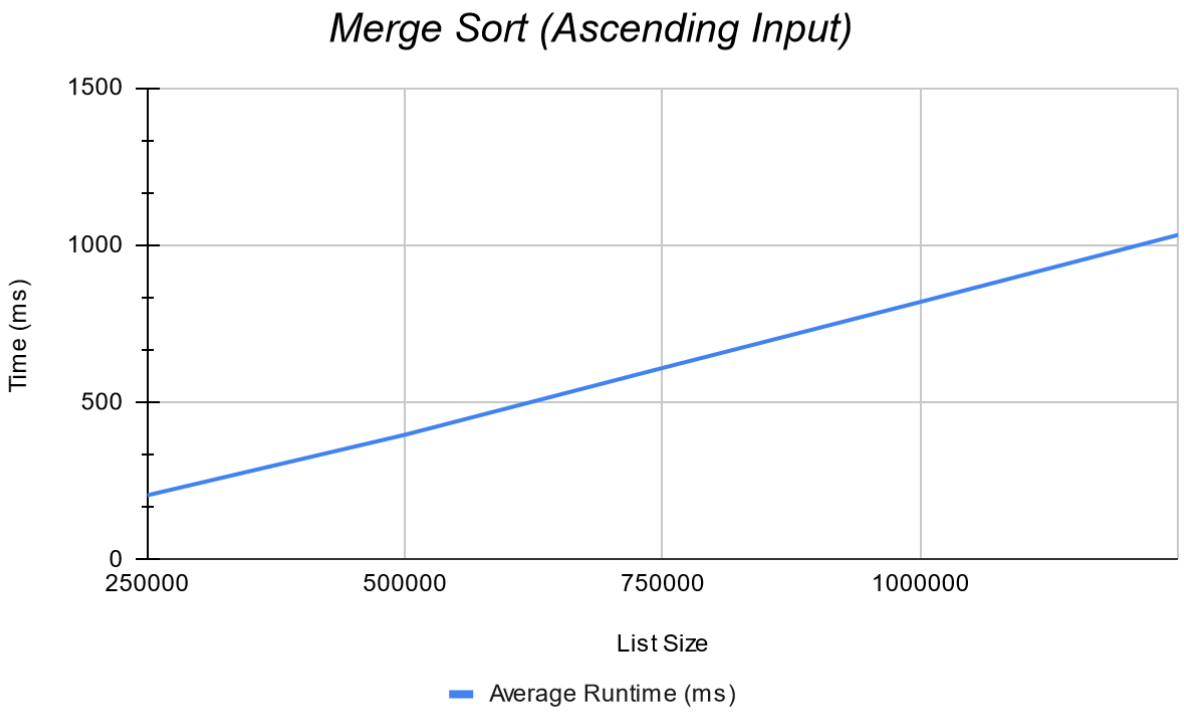
Table:

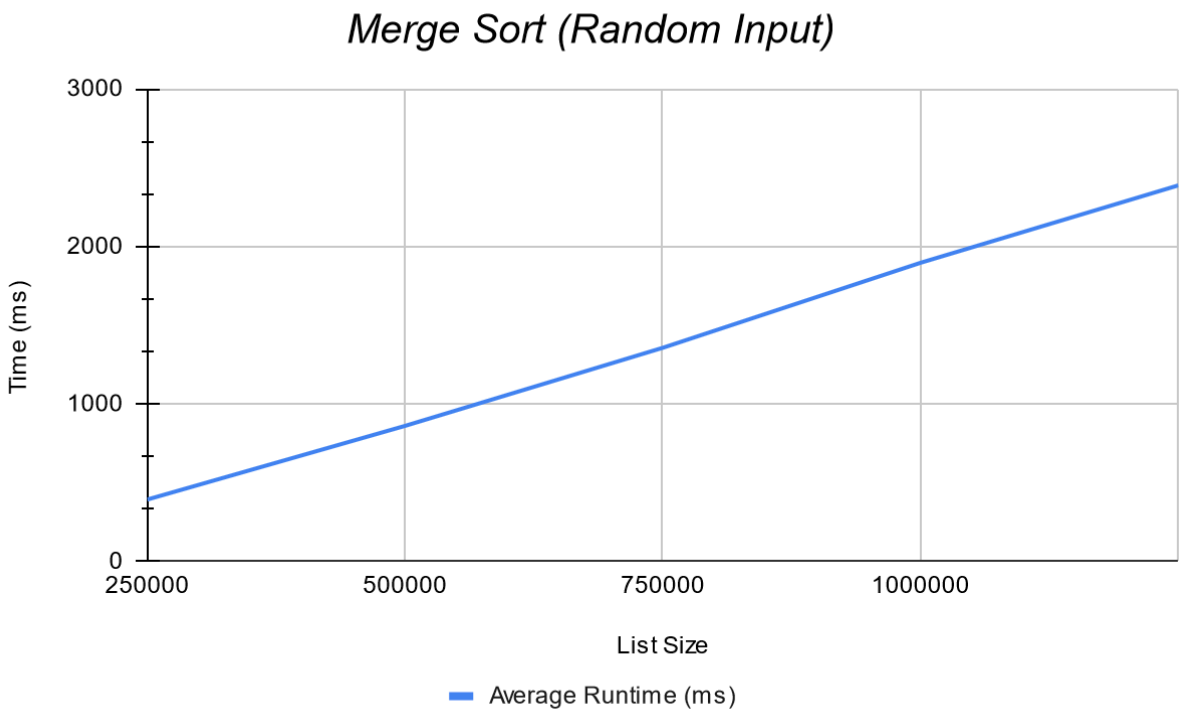
Merge Sort					
Initial Data Method	Size	1st Runtime (ms)	2nd Runtime (ms)	3rd Runtime (ms)	Average Runtime (ms)
Descending	250000	196.666	190.226	195.417	194.103
Descending	500000	392.851	396.315	392.701	393.9556667
Descending	750000	613.364	606.113	609.449	609.642
Descending	1000000	813.894	816.675	821.294	817.2876667
Descending	1250000	1036.523	1035.834	1032.408	1034.921667
Ascending	250000	203.423	198.421	213.476	205.1066667
Ascending	500000	398.232	398.303	395.967	397.5006667
Ascending	750000	606.895	614.439	608.36	609.898
Ascending	1000000	819.198	819.01	821.009	819.739
Ascending	1250000	1028.55	1037.05	1032.337	1032.645667
Random	250000	387.808	404.723	389.222	393.9176667
Random	500000	869.861	859.815	854.598	861.4246667
Random	750000	1357.173	1351.355	1362.662	1357.063333
Random	1000000	1916.941	1886.674	1886.27	1896.628333

Random	1250000	2390.214	2361.381	2411.911	2387.835333

Graphs:




Merge Sort (Ascending Input)
1500
1000
500
0
Time (ms)
250000
500000
750000
1000000
List Size
Average Runtime (ms)


Merge Sort (Random Input)
3000
2000
1000
0
Time (ms)
250000
500000
750000
1000000
List Size
Average Runtime (ms)