240 P3 Data and Analysis- Jordan Martin

Input Size (N)	merge	merge1	quicksort
1000	0.00014958	0.00013436	0.00010833
2000	0.00026997	0.00025647	0.00021004
3000	0.00038966	0.0003824	0.00030517
4000	0.00054148	0.00051013	0.00042454
5000	0.00073691	0.00070822	0.00056198
6000	0.0009294	0.0008614	0.00071982
7000	0.00104083	0.00100639	0.00080525
8000	0.00128618	0.00123211	0.00102312
9000	0.00140595	0.00134626	0.00109032
10000	0.00198506	0.00194869	0.00153527

Figure 1.1 Improving Merges (Ordered) Table

This table represents the runtimes in seconds for each of the three methods at their respective input sizes for an ordered input.

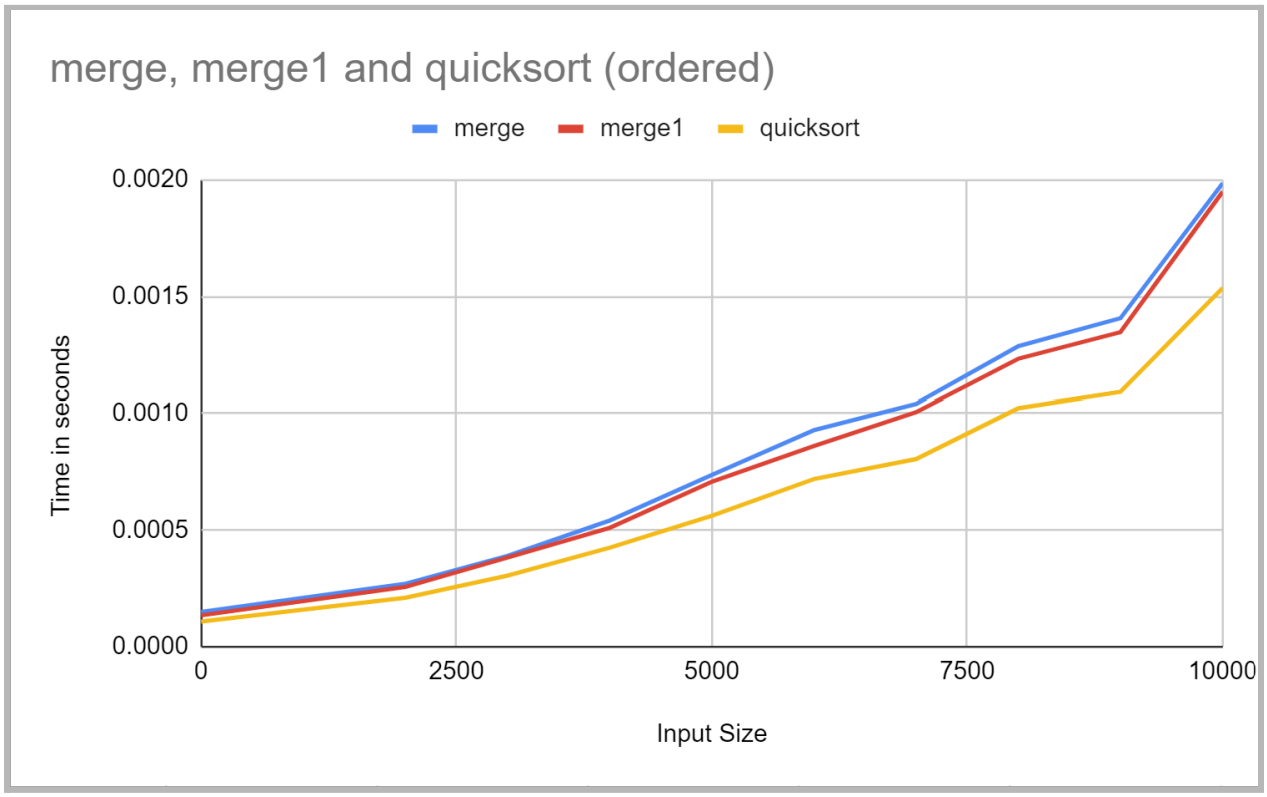

Figure 1.2 Improved Merges (Ordered) Chart

In this chart, the quicksort method has a faster runtime overall than both the merge and merge1 methods. Additionally, the merge1 method has overall a faster runtime than merge, but not by an exceptional amount. The merge method requires a temporary array of the size of the entire

array while merge1 only requires a temporary array of *half* the size of the entire array, which allows it to run slightly faster. Quicksort requires little space compared to the former two methods, which is why it has a substantially faster runtime than them.

Figure 1.3 Improved Merges (Random) Table
This table represents the runtimes in seconds for each of the three methods at their respective input sizes for a random input.

Input Size (N)	merge	merge1	quicksort
1000	0.00023561	0.00020329	0.000159
2000	0.00051735	0.00043296	0.00034405
3000	0.00077017	0.00067893	0.00052833
4000	0.00105558	0.00092006	0.00074375
5000	0.00136984	0.00118181	0.00091136
6000	0.00181872	0.0016649	0.0012878
7000	0.00189573	0.00166455	0.00133562
8000	0.00224175	0.00192862	0.00160422
9000	0.00255369	0.0022478	0.00177976
10000	0.00283093	0.00245125	0.00196355

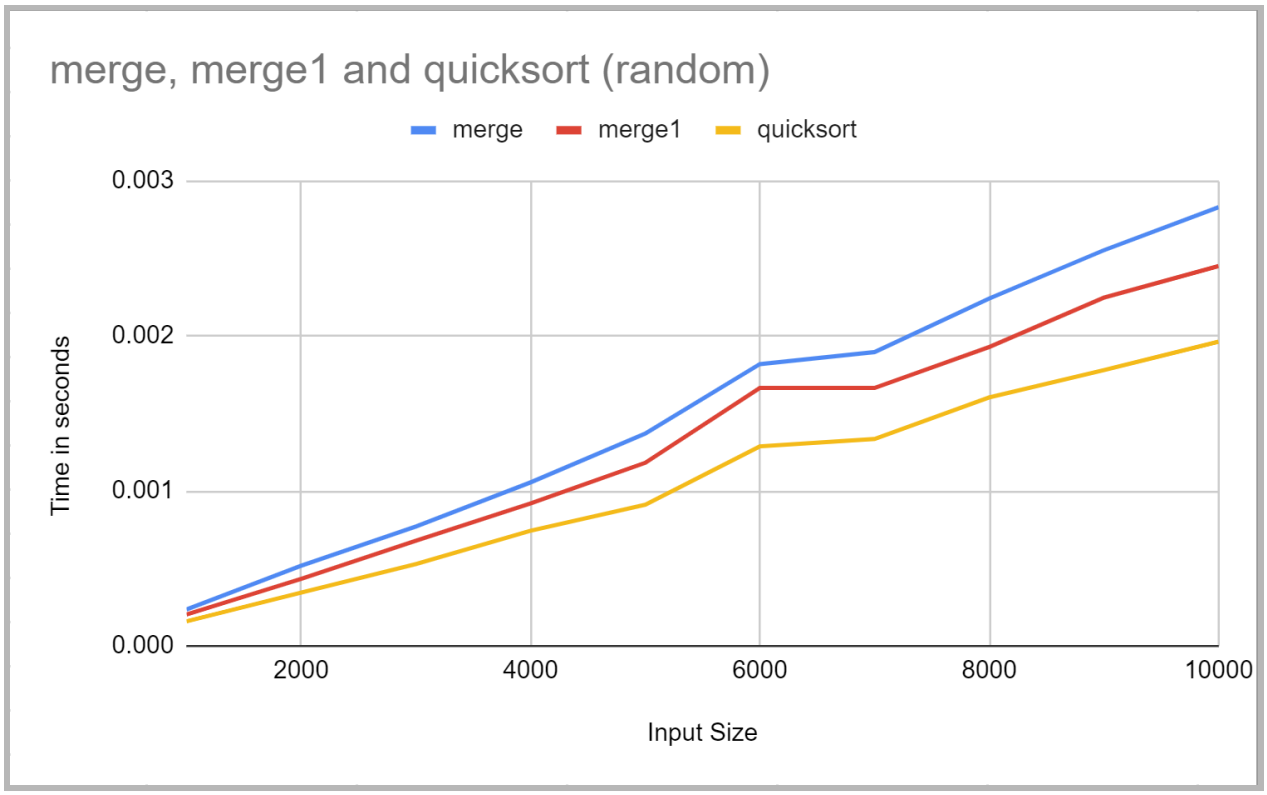

Figure 1.4 Improved Merges (Random) Chart
Contrary to the ordered chart in figure 1.2, the runtimes are far more separated from each other. Merge1 stands about midway between quicksort and merge with the former maintaining the title of fastest runtime. I believe I achieved a “lucky” run for this one, as I expect merge1 to still be closer to the line for merge than shown in this graph.

Figure 2.1 Threshold Selection Table
This table represents the runtimes in seconds for each of the different thresholds at their respective input sizes for a random input.

	insertion	merge2(155)	merge2(120)	merge(100)
20	0.0000025	0.00000077	0.00000065	0.00000073
40	0.00000282	0.00000249	0.00000193	0.00000223
60	0.0000055	0.00000536	0.00000369	0.00000431
80	0.00000847	0.00000803	0.00000614	0.00000716
100	0.00001374	0.00001424	0.00000881	0.00001292
120	0.00001607	0.00001595	0.00007667	0.00007949
140	0.00002252	0.00002192	0.00003364	0.00003345

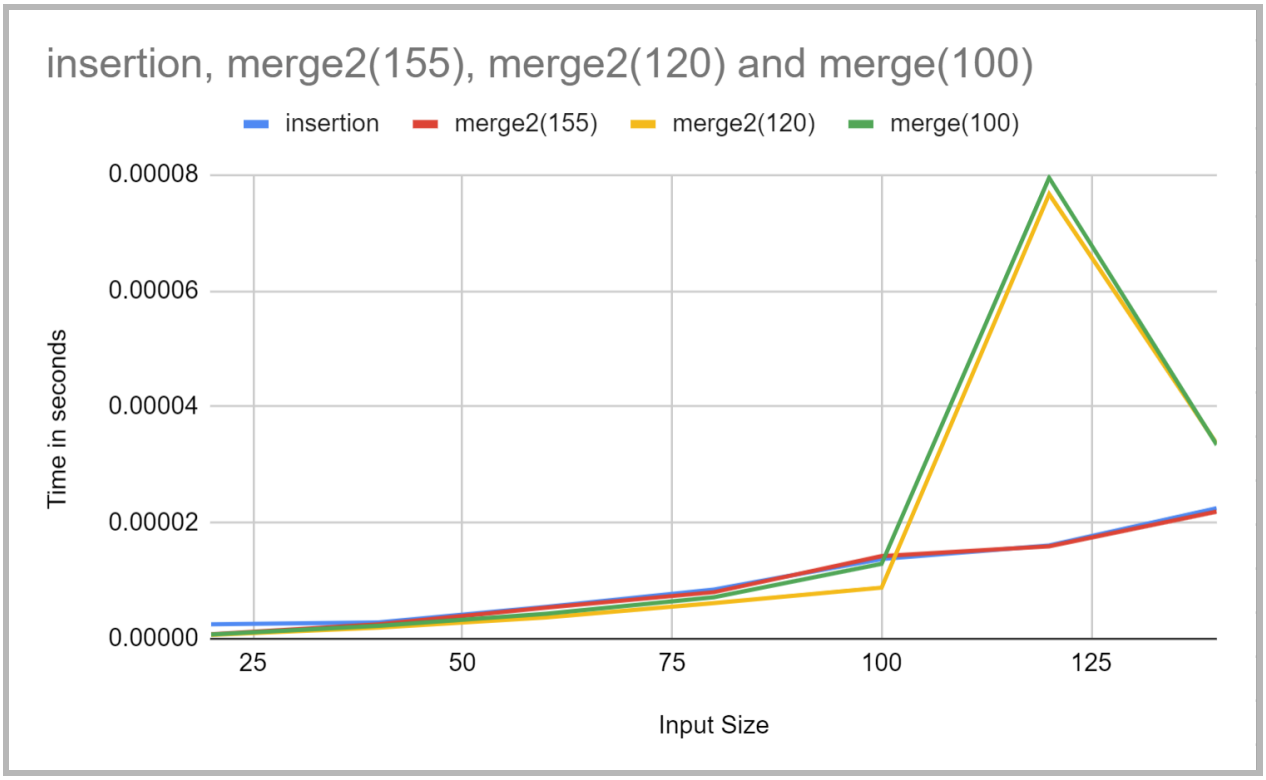

Figure 2.2 Threshold Selection Chart

This graph gives a very obvious answer to what the threshold should be. At both 120 and 100, the time spikes all the way up to 0.00008 seconds from 0.00002 seconds, but at 155, the line stays consistent with insertion.

Figure 3.1 Switching Strategies (Ordered) Table

This table represents the runtimes in seconds for each of the four methods at their respective input sizes for an ordered input.

Input Size (N)	merge	merge1	merge2	quicksort
1000	0.00011803	0.00025377	0.00022342	0.00008456
2000	0.00025212	0.0003388	0.00032302	0.00018548
3000	0.00040002	0.00039357	0.00039506	0.00029738
4000	0.00054211	0.00053717	0.00054375	0.00039506
5000	0.00072558	0.00072039	0.0006961	0.00052157
6000	0.0008986	0.00087788	0.0008739	0.00065816
7000	0.00110806	0.0010939	0.00108577	0.00082398
8000	0.00126768	0.00121344	0.00120958	0.0009105
9000	0.00141938	0.00139399	0.00139665	0.00104701
10000	0.00158752	0.00153474	0.00161863	0.00112937

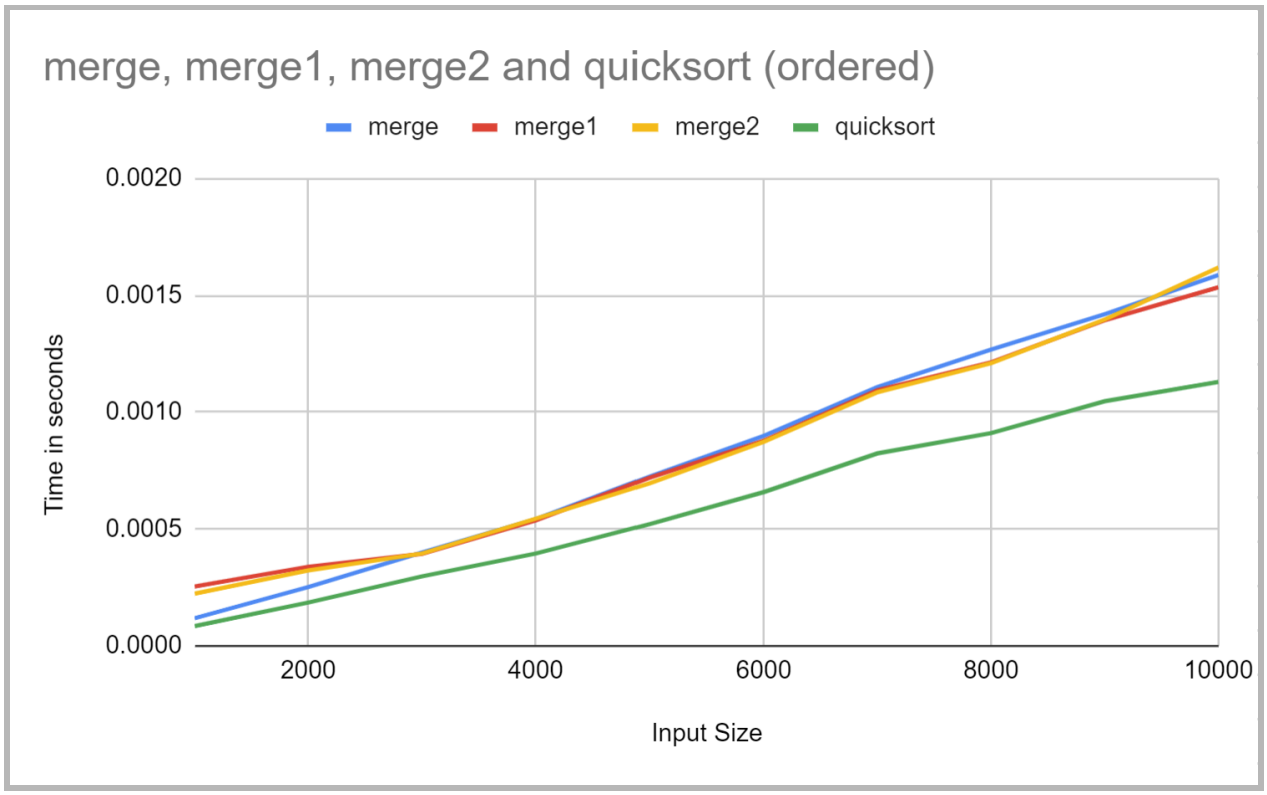

Figure 3.2 Switching Strategies (Ordered) Chart

In this chart, quicksort is substantially faster than any of the three merge methods. Otherwise, all three of the merge methods are relatively the same runtime. Merge1 and merge2 are almost the exact same runtime.

Figure 3.3 Switching Strategies (Random) Table
This table represents the runtimes in seconds for each of the four methods at their respective input sizes for a random input.

Input Size (N)	merge	merge1	merge2	quicksort
1000	0.00011225	0.00027877	0.00022792	0.00008617
2000	0.00024765	0.00052876	0.00049817	0.00018237
3000	0.0003956	0.00080191	0.00080164	0.00028671
4000	0.00053738	0.00077363	0.00080179	0.0003942
5000	0.00071243	0.00068091	0.00068942	0.00052738
6000	0.00086401	0.00081157	0.00081537	0.00065086
7000	0.00102465	0.00097389	0.00096523	0.00073211
8000	0.00119931	0.00116759	0.00115878	0.00088153
9000	0.00133886	0.00129183	0.00126583	0.00100784
10000	0.00151983	0.00145301	0.00142886	0.00110244

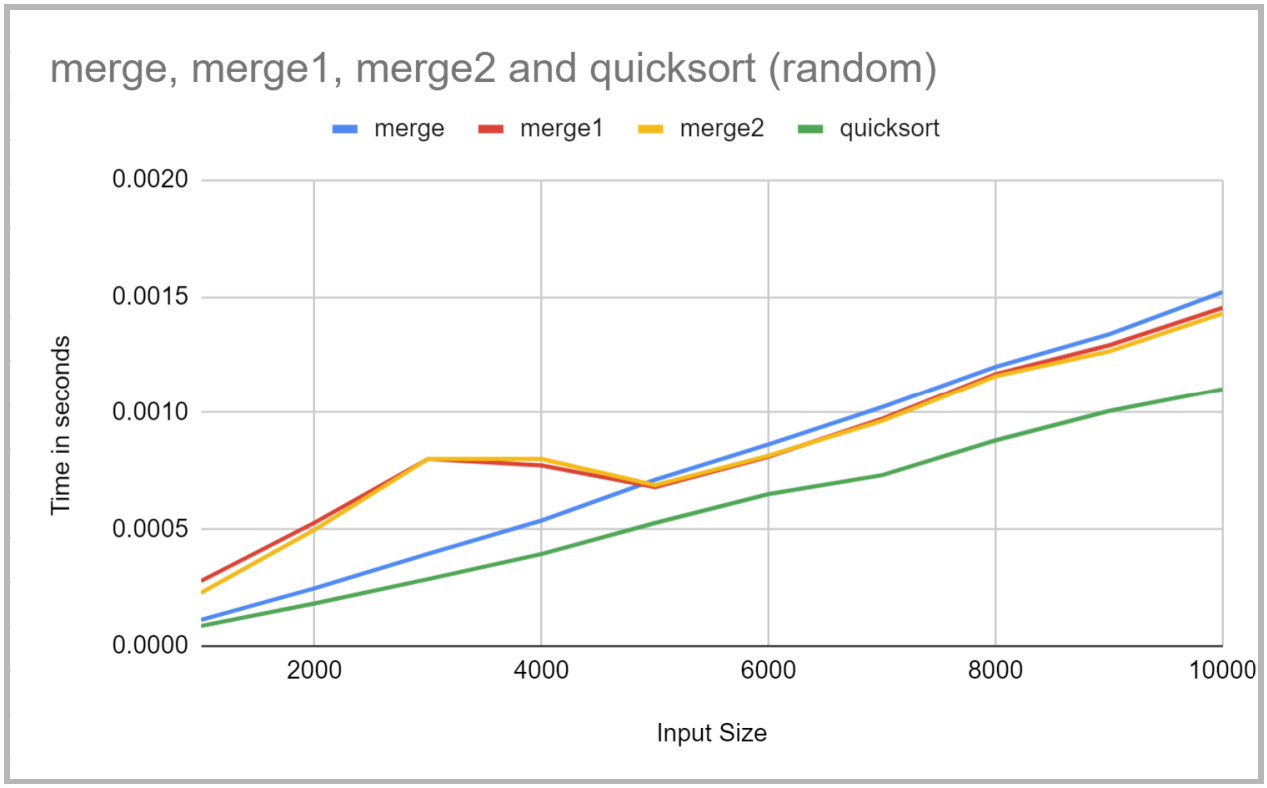

Figure 3.4 Switching Strategies (Random) Chart
Unlike the ordered chart, merge is clearly separated from merge1 and merge2. Like the other graph, though, merge1 and merge2 are nearly identical lines. I think this is because I call almost the same sorting algorithm for both of those methods. Merge, however, is completely separate from both of the methods I created, which is why the line is separate.

Figure 4.1 Introspective Sort (Ordered) Table
This table represents the runtimes in seconds for the two methods at their respective input sizes for an ordered input size.

	introspective,	quick
1000	0.0242922	0.00020781
2000	0.01537089	0.00029869
3000	0.00890715	0.00047969
4000	0.01464083	0.00065598
5000	0.08052096	0.00146293
6000	0.01907055	0.00141394
7000	0.02223806	0.00128994
8000	0.07667904	0.00132544
9000	0.08192113	0.0018617
10000	0.00618568	0.00223178

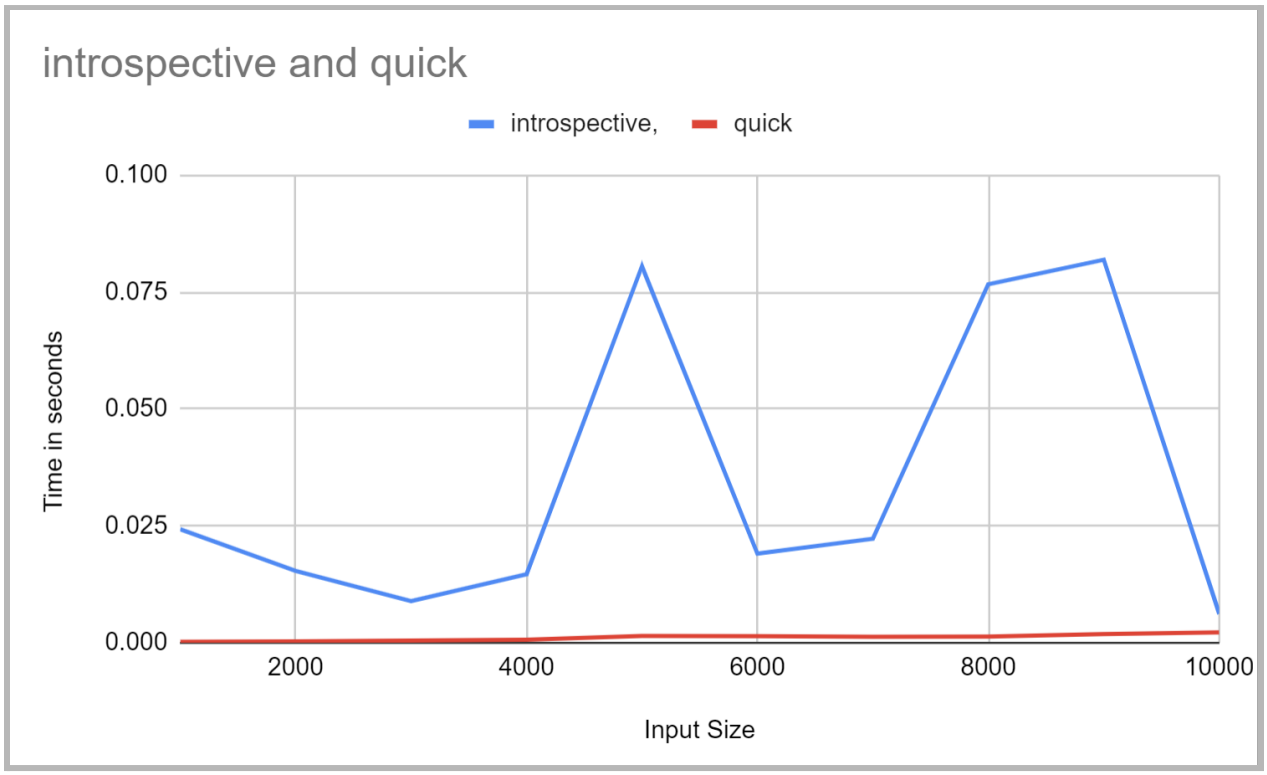

Figure 4.2 Introspective Sort (Ordered) Chart
In this graph, quicksort is very visibly faster than quicksort. Introspective is a much more staggered and has a much slower time than quicksort.

Figure 4.3 Introspective Sort (Random) Table
This table represents the runtimes in seconds for the two methods at their respective input sizes for a random input size.

Input Size (N)	introspective	quick
1000	0.0053623	0.00022071
2000	0.02403888	0.00044714
3000	0.01012293	0.00043638
4000	0.01386786	0.00060824
5000	0.00738587	0.00073583
6000	0.003984	0.00085329
7000	0.0189222	0.00129713
8000	0.00259967	0.00133192
9000	0.01389032	0.00175152
10000	0.00398485	0.00195644

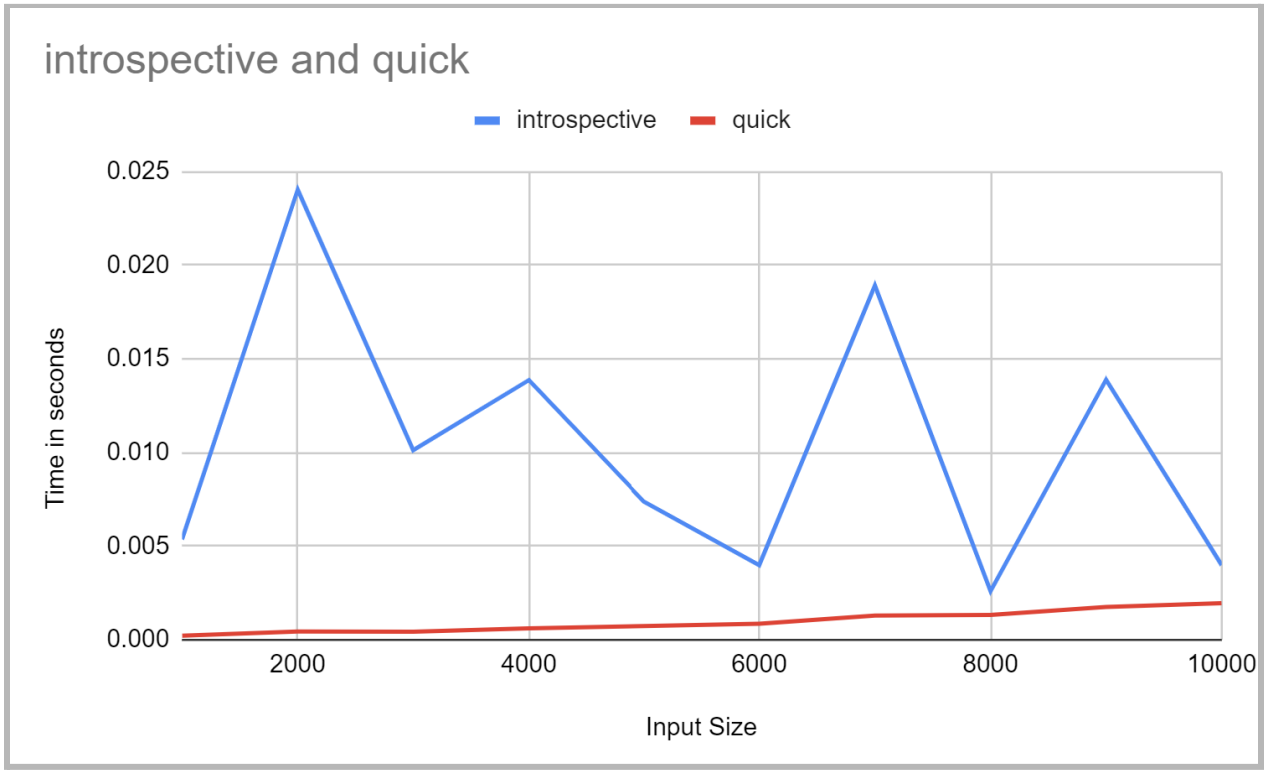

Figure 4.4 Introspective Sort (Random) Chart
In this graph, quicksort is faster than introspective, but the runtime increases faster than it did on the ordered chart. Additionally, introspective fluctuates to a much lower degree than it did in the previous graph.