Kris Elsrod

Table of timing results, in seconds:

Selection	Insertion	Bubble	Quick	Merge	Built In
0.120514	0.0804276	0.53789	0.0014107	0.00306212	0.00139811
0.235481	0.133428	0.590012	0.00175544	0.00433777	0.0017017
0.0934997	0.108513	0.470152	0.00150236	0.0157227	0.0018336

Selection and Insertion sort appear to take a very similar amount of time on average. Quick sort and the built in sort also have very similar times, likely due to the built in sort being a variation of quick sort. Merge sort is better than selection and insertion but worse than quick sort and the built in sort. Bubble sort is the worst of them all by a significant amount.

These results make sense if you compare them to their big O notations. Quick sort, the built in sort, and merge sort are all nlogn complexity which is why their times are similar to each other. Insertion, selection, and bubble are all n^2 complexity which is why they are slower than the previously mentioned sorts. Surprisingly, bubble sort took more than twice as much time than insertion and selection sort, even though they have the same complexity.

According to the results, quick sort and the built in sort are the best for large data sets with merge sort close behind. This is because of their divide and conquer strategy, which gives them a large advantage. Insertion and selection aren’t terrible, but there are much better options. Bubble sort should never be used for large data sets, because of how badly it did compared to all the other sorting methods. This is likely because bubble sort does more swaps and comparisons than even insertion and selection sort, even when those operations aren’t necessarily needed. In conclusion, you should use divide and conquer sorting methods like quick, merge, and the built in sort when working with large data sets. If you don’t care about efficiency too much, insertion and selection can get the job done albeit a lot slower.