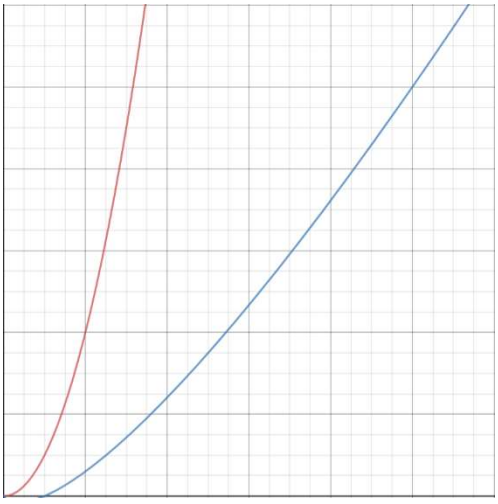
Each of the sorting algorithms were capable of properly sorting all of the values, though with some large differences in the time they took to do so. After running the program a few times with 100,000 randomly generated floating point values, I found that bubble sort took approximately 53 seconds to execute every time. This was far and away the slowest algorithm. Insertion was the next slowest at about 15 seconds per execution and selection was right behind it at roughly 13 seconds. Even these were easily surpassed by both merge sort and quick sort. Merge sort consistently ran at about 0.03 seconds. Quick sort was slightly faster at 0.02 seconds. Going into this, I knew that merge and quick sort would be faster, but it shocked me just how much more efficient the divide and conquer methods were. Looking at task manager, under performance, bubble sort stayed at approximately 30% utilization during these tests, with regular spikes to about 60%. Both insertion and selection sort were also about 30%, though their spikes only averaged 50%. With both merge and quick sort, they executed so fast that they didn't even show up on task manger.

When I ran the program with my file test1 (which only contains 10 values), each of the algorithms ran in about the same amount of time, each only taking hundred-thousandths of a second to run. This shows that there is a great difference in performance when large sets of data need to be sorted. This makes sense, as merge sort and quick sort are both logarithmic functions whereas bubble, insertion, and selection are quadratic. Here is a graph showing portions of the n^2 (in red) and n*logn (in blue). The red line is far steeper than the blue line, but close to 0, they are actually quite close to each other.



Now this method of testing is not perfect; these algorithms will all run differently on different computers. However, it does show that merge and quick sort are far faster than their quadratic counterparts. The slower methods do have their advantages

though. First of all, they are far easier to implement. One can reason through bubble sort without much trouble at all. As said previously, if you know that you will only be sorting a few values, then the quadratic methods will likely suit your needs just fine. Another appeal is that bubble, selection, and insertion sort do not use recursion. As not all languages support recursion, you may need to opt for one of these sorting algorithms. An additional point in favor of insertion sort is that if your data is already partially sorted, insertion sort will perform much faster.

Overall, each of these sorting algorithms have their uses. Choosing the best algorithm for one case may not be the best fit for another. Finding the right method all depends on your particular case and the data with which you are presented.