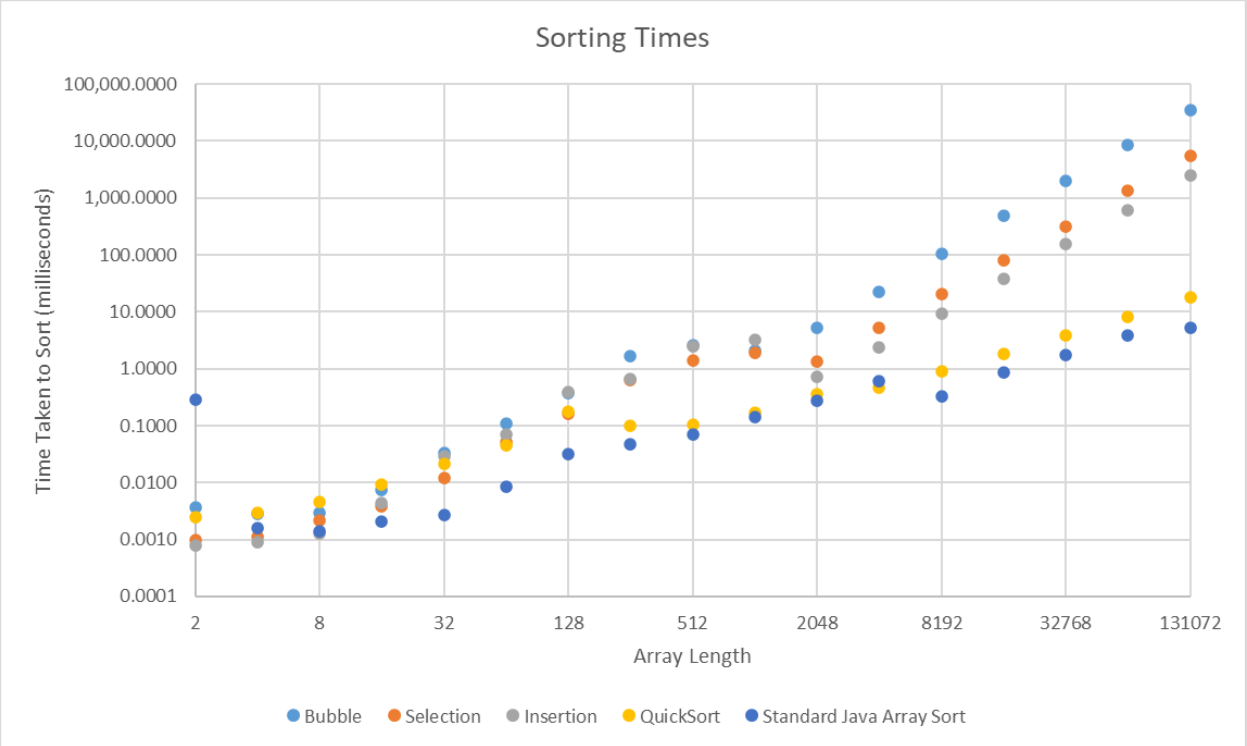
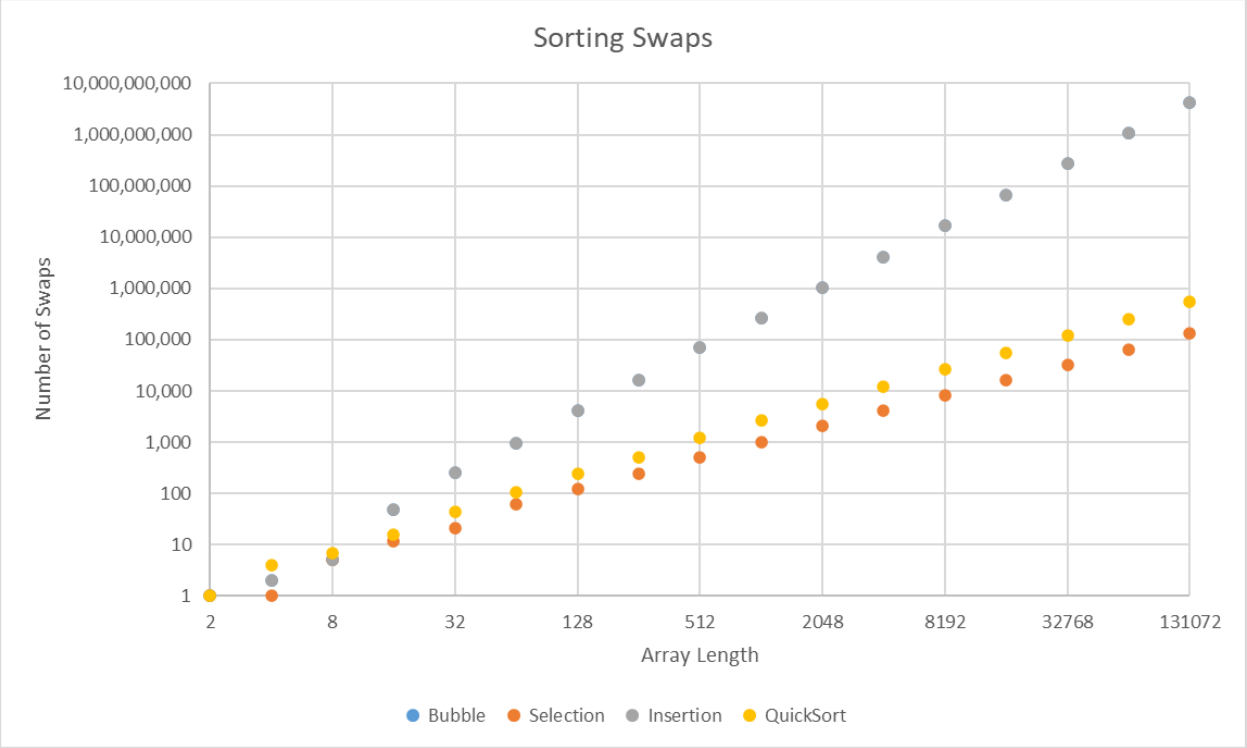
Brent Orlina

I implemented four different sorting algorithms, bubble sort, selection sort, insertion sort, and quick sort. The standard sort function that Java uses was also included for testing. Each algorithm was tested using an array with a randomized number in each element with the maximum value as the length of the array. They were tested from an array length of 2 to 2^17. Going any further than 2^17 takes more than a minute for bubble sort to complete. They were benchmarked in the time it took to sort the array and the number of swaps and comparisons when sorting. The standard sort function was not included in benchmarking for the number of swaps and comparisons.

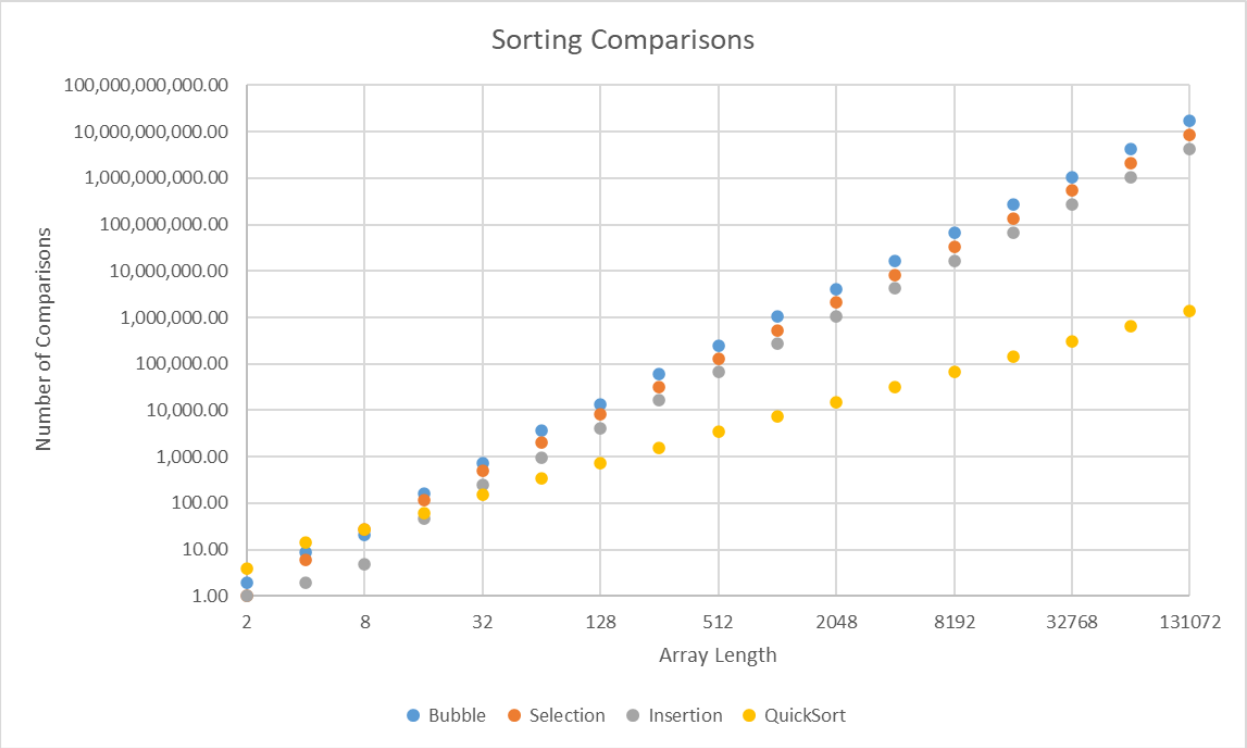


The results in benchmarking for time show that the standard sort function is faster on average when used with bigger arrays. In the smaller array sizes, greatly seen in length 2, it has similar run times, if not worse, than the other algorithms. Bubble sort takes the most time with most of the bigger arrays. Quicksort has a similar run time as the standard sort function as the

standard function primarily uses quicksort when handling bigger array sizes. It can also be seen that quicksort and the standard sort function separates from bubble, selection, and insertion sorts. This supports the fact that the average time complexity of quicksort and the standard sort function is O(nlog(n)) while bubble, selection, and insertion sort is O(n^2).



The results of benchmarking for the number of swaps taken to sort the array show that selection and quicksort take less swaps than insertion and bubble sort. It also reveals that insertion and bubble sort take the same number of swaps, with insertion sort overlapping bubble sort. I believe this happens because both algorithms swap elements in pairs, while quicksort and selection sort does not. It also seems like bubble and insertion sort have, on average, a complexity of O(n^2) in terms of the number of swaps while quicksort has O(nlog(n)) and selection sort has O(n).



Benchmarking for comparisons however shows that bubble, selection, and insertion sort grow with a complexity of O(n^2), with insertion sort on average having the least amount of comparisons out of the three algorithms. Quicksort, on the other hand, grows with a complexity of O(nlog(n)), taking the least amount of comparisons compared to all of the algorithms.

In conclusion, using the standard sort function in Java is certainly the fastest way to sort an array. However, it uses a mixture of different algorithms to do so, including quicksort. Quicksort is the fastest algorithm to sort an array out of the four tested. It takes the least comparisons and the second least in swaps. However, if swapping elements is costly, selection sort can be a great replacement as it takes the least amount of swaps, growing at a complexity of O(n). Bubble and insertion sort are the most inefficient. However, insertion sort is an online sorting algorithm, meaning that it sorts as data comes in. This can be useful when a list being sorted has data that is continuously being added to it.

Replit: https://replit.com/@BrentOrlina/SortingBenchmark#BenchmarkResults.java