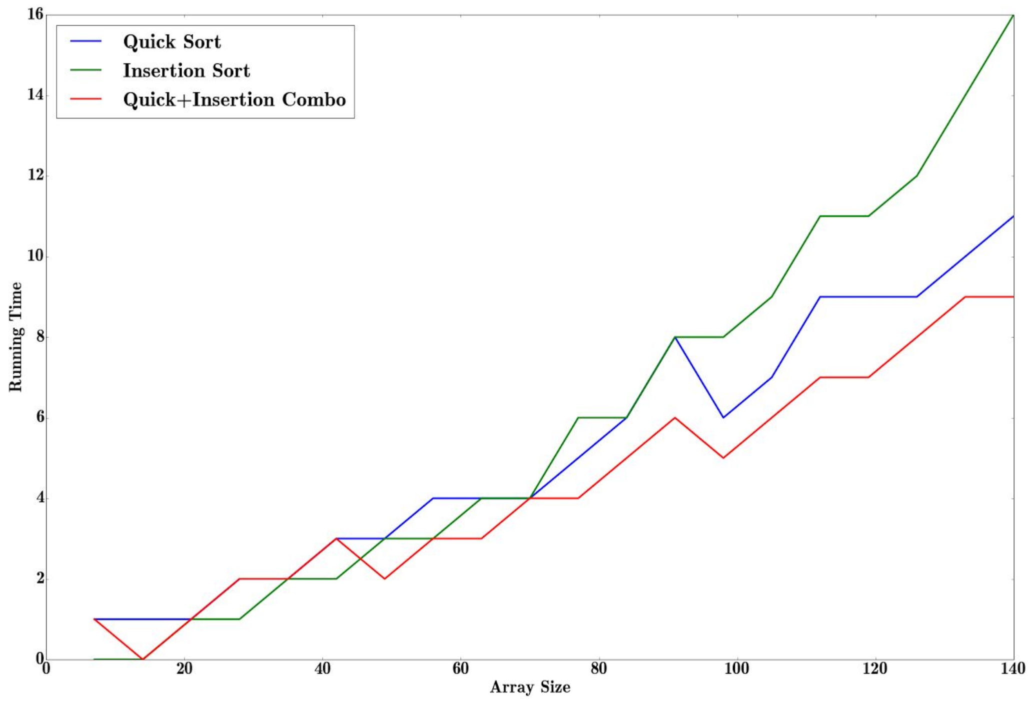
Analysis of Algorithms

Introduction

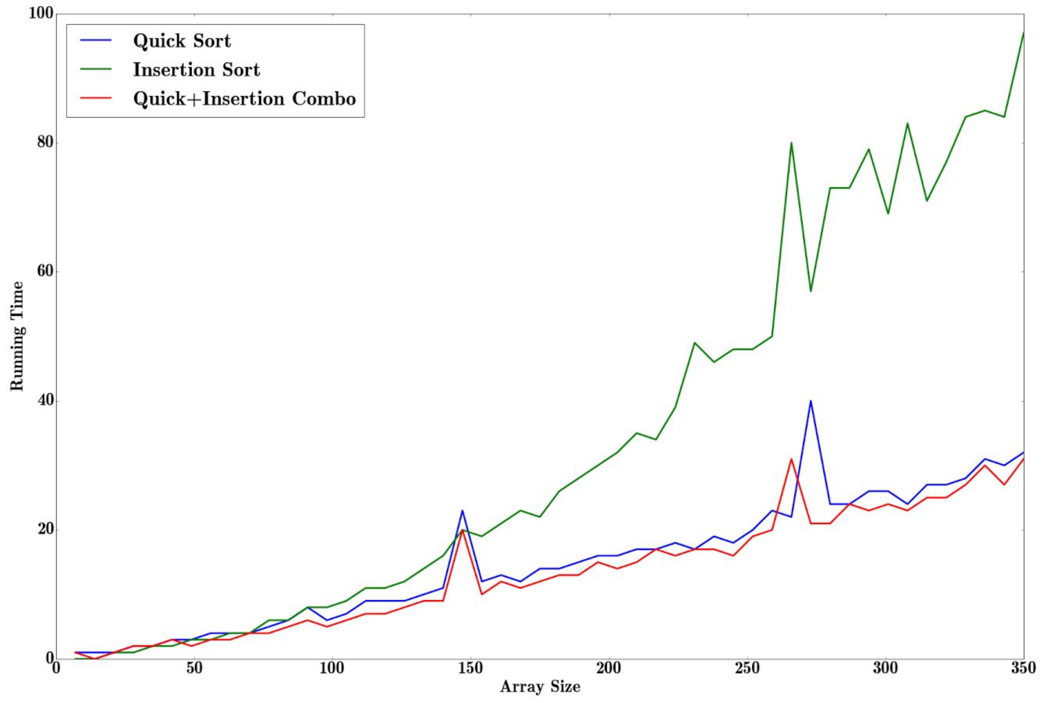
In this assignment, I chose to perform analysis on execution time of insertion sort and quick sort. I also tried the combination of both algorithms with certain threshold obtained from comparison of execution time of insertion sort and quick sort for various sized arrays.

Analysis

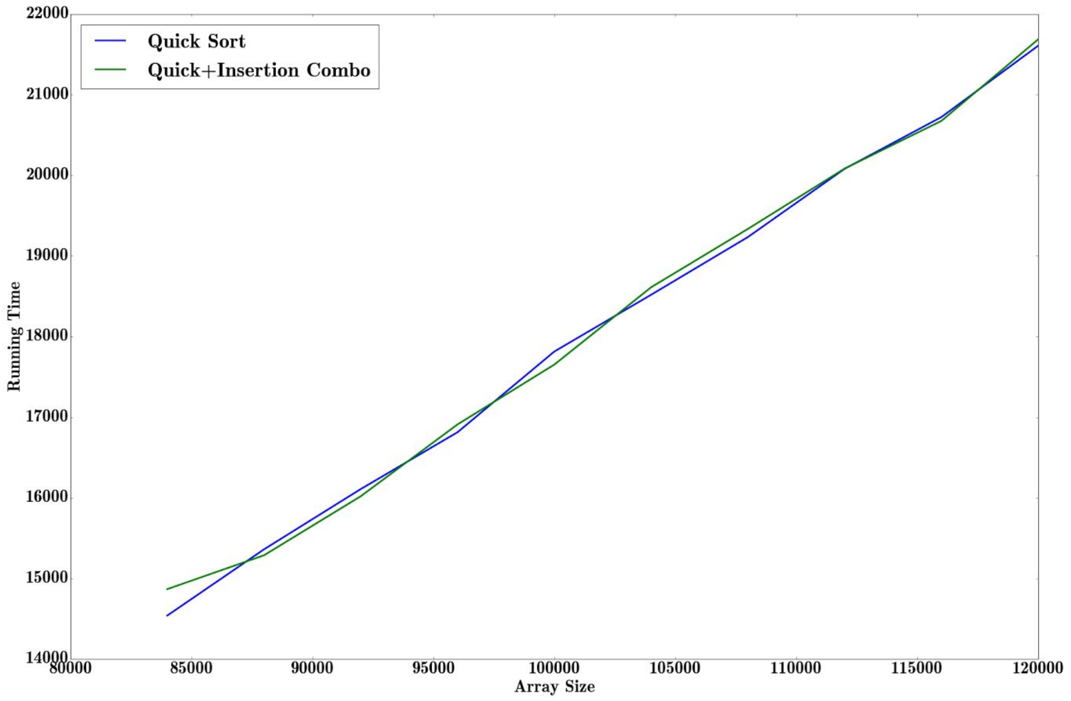
I ran random sized array for both algorithms starting from small size like 10 to very large size of 50,000. Insertion sort seemed to perform well compared to quick sort till the array size is around of 60 to 70. The difference in execution time for insertion sort and quick sort for smaller arrays is not seemed to be notably different though. But, after array size reaches about 100, the execution time of insertion sort increases drastically compared to quick sort. There are few spikes in the graph below which shows drop of execution time even with increasing array size. This is because of the fact that both sorting algorithm highly depend on how they are ordered initially before sorting.



Considering the threshold of about 70, I implemented a sort algorithm which is the combination of insertion sort and quick sort. As quick sort is a divide and conquer algorithm, it recursively divides arrays into smaller subarrays to perform sorting. Taking note of this fact, I applied insertion sort to smaller subarrays within the quick sort to see if performance improves with quick sort.



With the combination, the execution time seemed to be slightly improved, but not drastically like it was between quick-sort and insertion sort. I also run quick-sort and this combinatory-sort for even larger arrays of about size 50,000. The execution time of the combinatory-sort is better most of the time compared to quicksort, but not always, and the difference is not so bigger. The reason may be because of smaller difference in execution time between insertion sort and quick-sort for smaller arrays is not comparable to the difference it makes too the sorting algorithm by initial position of array elements of arrays when array goes bigger.



Conclusion

Though insertion sort performed better for smaller arrays with less than 70 elements, it performs worse compared to quick-sort for larger datasets. Due to the effect of initial position of the array elements in an array and the small difference in execution time between quick-sort and insertion sort for smaller arrays, there is not much improvement by making a combinatory sorting algorithm of these two.