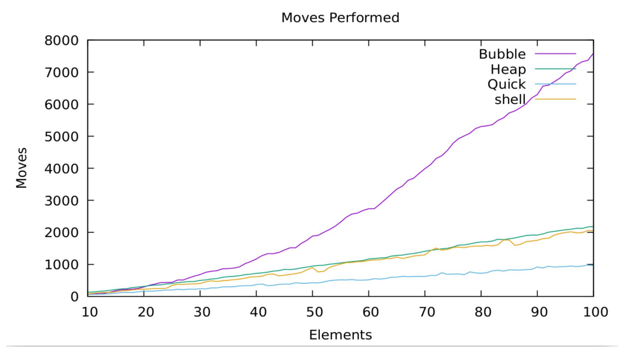
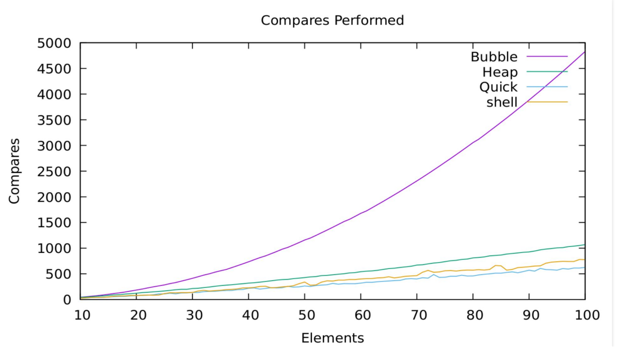
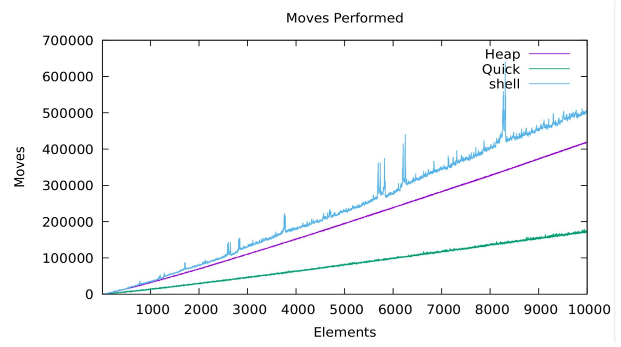
Results of Small Amount of Elements

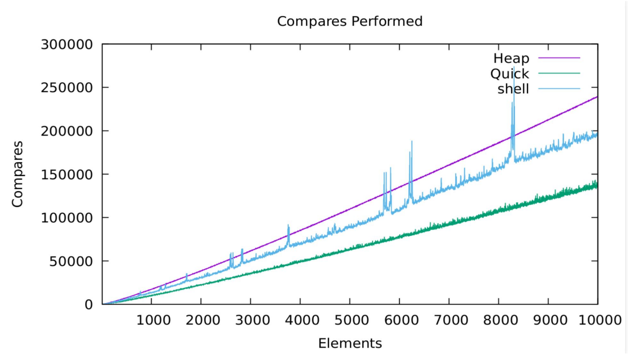




With a small number of elements (0-30) All of the sorts are quite similar. However once there are more than 30 elements, bubble sort begins to pull away from the rest extremely quickly. It also shows just how many more compares and moves an O(n^2) algorithm needs compared to a nlog(n).

Results of Large Amount of Elements

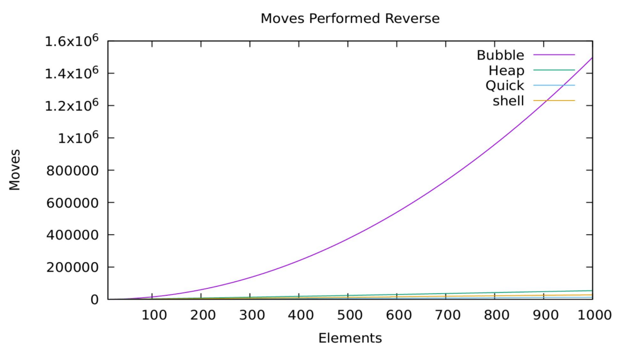


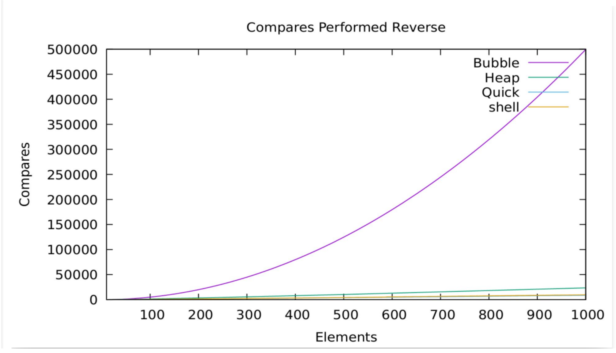


When testing 10,000 elements, I decided to disable Bubble sort as it was taking exponentially more time to sort. Something I found interesting is that Heap sort and Quick sort seem to be opposites in terms of moves and compares. I would believe that the use of either of these depends on the cost of a move or compare.

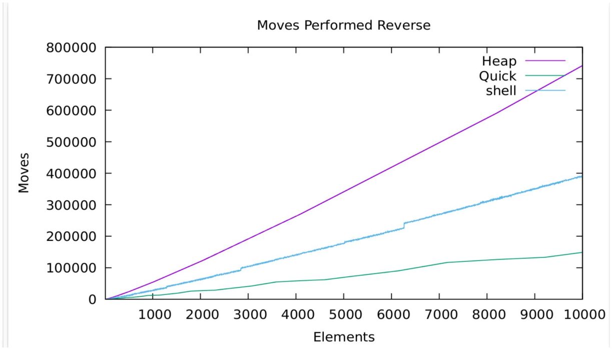
Something I found very interesting is how quicksort beats out everything else. I wonder why you would want to use any other sort when quicksort will be so much better as you increase the elements in the array.

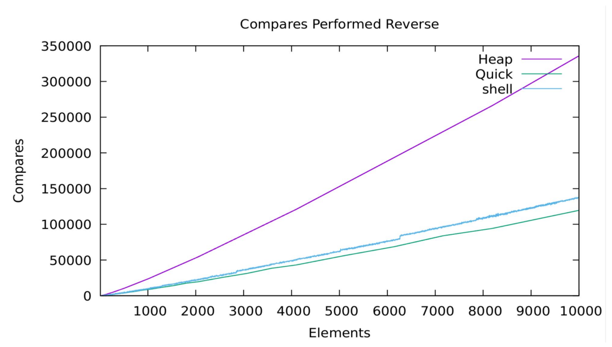
Results of Medium/Large Amount of Elements Reversed





When giving the algorithms an array that is reversed, but in order, we can see that the previous trends are still followed, Bubble sort pulls away quickly and grows at an increasing rate compared to the other sorting methods.





Something that I found interesting is how the Heap sort seems to deal with a pre-sorted list quite poorly when compared to shell sort now. In fact, in terms of comparison, shell sort is approaching quicksort. Despite this, quicksort ends up beating out both of these other algorithms. However, this does seem to suggest the conclusion that if an array is already partly sorted, shell sort is not a bad option to choose instead of heap sort.

Other Features

Something that was also extremely interesting is that for all of the algorithms except Bubble, their graphs were not smooth. Quicksort seemed to wiggle around slightly and certain lengths of shell sort caused drastic increases in the sorting compares and moves. I believe this might be caused by how shell short swaps values, it is possible that a value may be looked over for quite a long time which could lead to seemingly random lengths taking much longer than expected. This is also what could be causing quicksort to wobble a little bit on the graph as it slightly depends on quicksort for some of its swaps and compares.

I believe bubble is relatively smooth because the amount of time it takes is quite similar since it will reliably take O(n^2) time. The algorithm will still be comparing tons of values no matter how many elements are in the array making it extremely reliable although it is quite inefficient at higher lengths.