Assignment 3 - Sorting: Putting your Affairs in Order Writeup

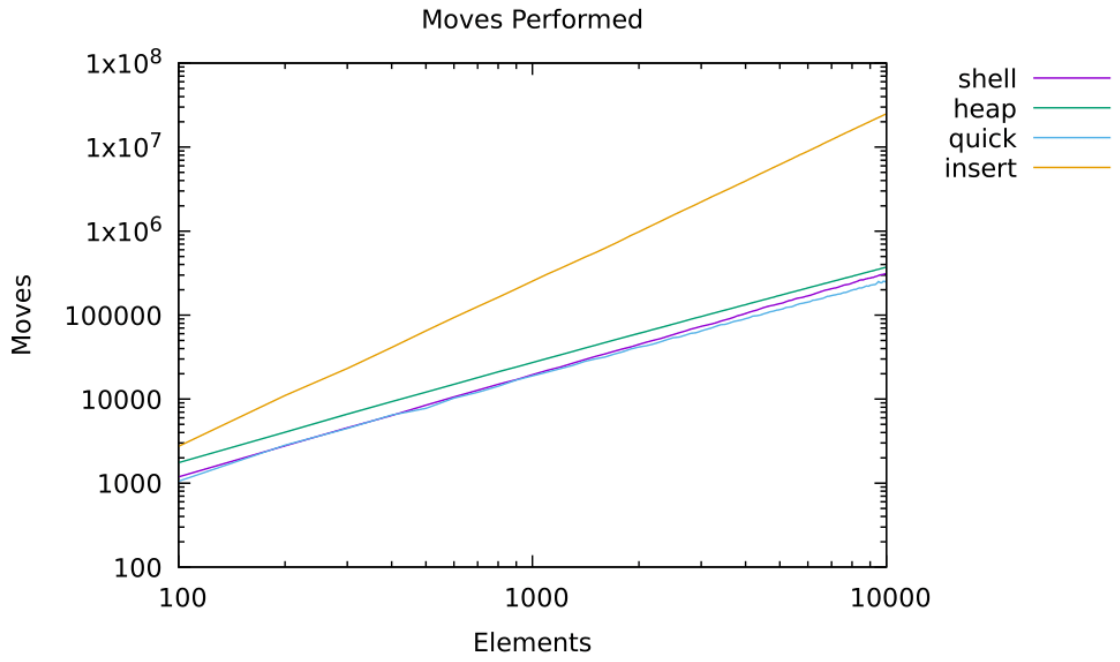
Sean Carlyle

October 12 2021

1 Introduction

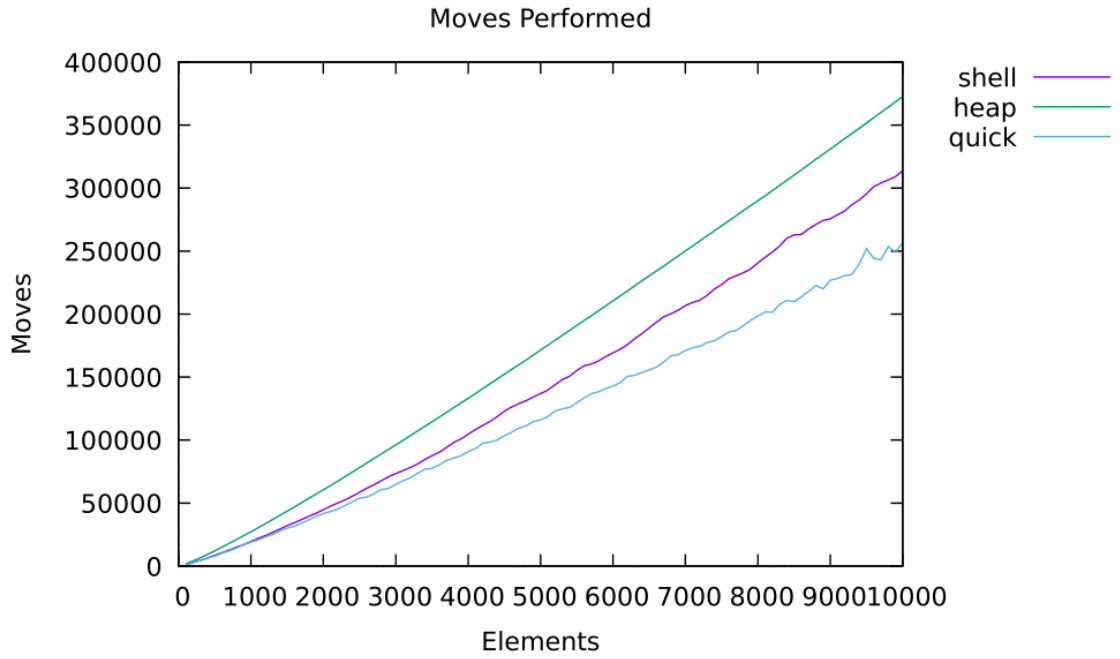
This is my assignment 3 writeup. This writeup includes 6 graphs comparing and contrasting the different times complexities of the different sorting algorithms using the number of moves and comparisons the algorithms use. A move is when the algorithm moves an element in the array, and a compare is when the algorithm compares an element in the array. My Writeup compares the graphs of the different algorithms to try and find their time complexities.

2 Moves Performed Comparison



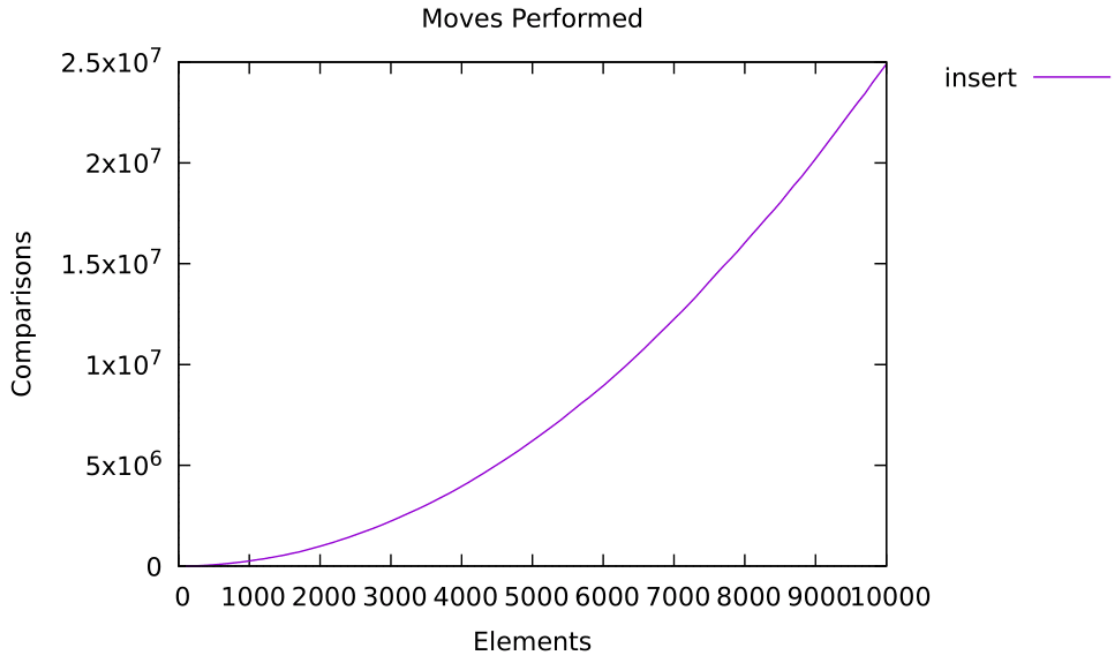
This is the graphed data of the four sorts on how many moves they performed with various amounts of elements in the arrays. As we can see, insertion sort is the least efficient by taking more and more moves to sort for every element. Its hard to tell the time complexity for each algorithm with this graph because it was scaled to fit insertion. Insertion uses so many moves that on normal axes the other sorting algorithms are eclipsed by Insertions moves used. We can see however, that the three sorts heap, quick, and shell have almost identical graphs. This means they may have the same time complexity. I will use the next to graphs to explain the sorting algorithms complexities.

3 Moves Performed without Insertion



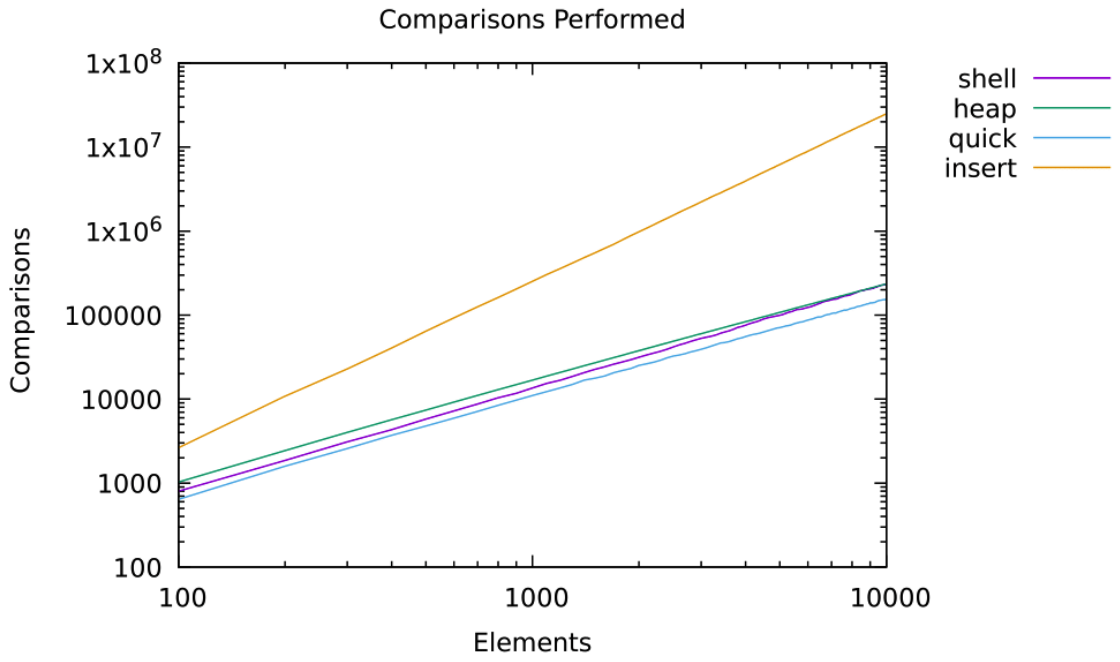
This shows three of the sorts without insertion graphed with normal axes. We can clearly see that heap is the least efficient, eventually taking more moves over many elements. We can see by smoothness of heaps graph however, that it is the most consistent overall and this reinforces the idea that its best case and worst case are the same. Shell is second best, but seems to deviate from the curve more often which leads me to believe it might have some cases that make it worse than its average time complexity. Quicksort while having the least moves used, has the most deviations from the line. This makes it seem like there are some very bad cases in quick sort that make its time complexity much worse. The average time complexity seems to be around n(log(n)) for each of these graphs.

4 Insertion Moves Performed



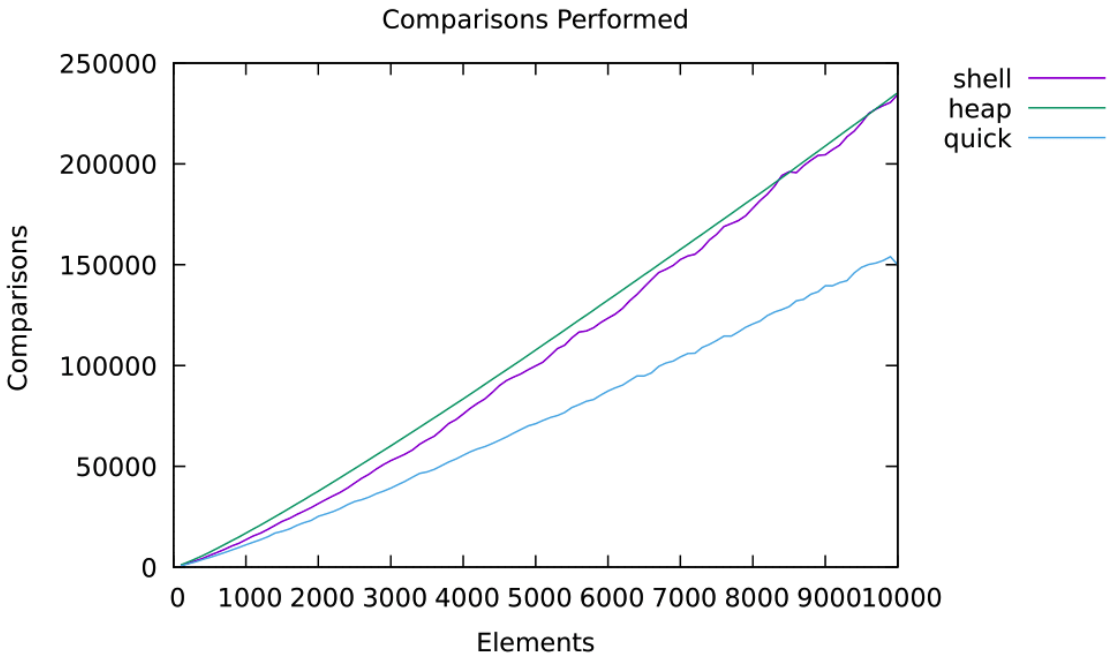
Moves performed by insertion sort start out small but exponentially get greater and greater the more elements. This shows us that the time complexity of insertion is probably n^2. While we could not see this in the first graph due to its different axes. We can now see that the graph looks very exponential.

5 Comparisons Performed Comparison



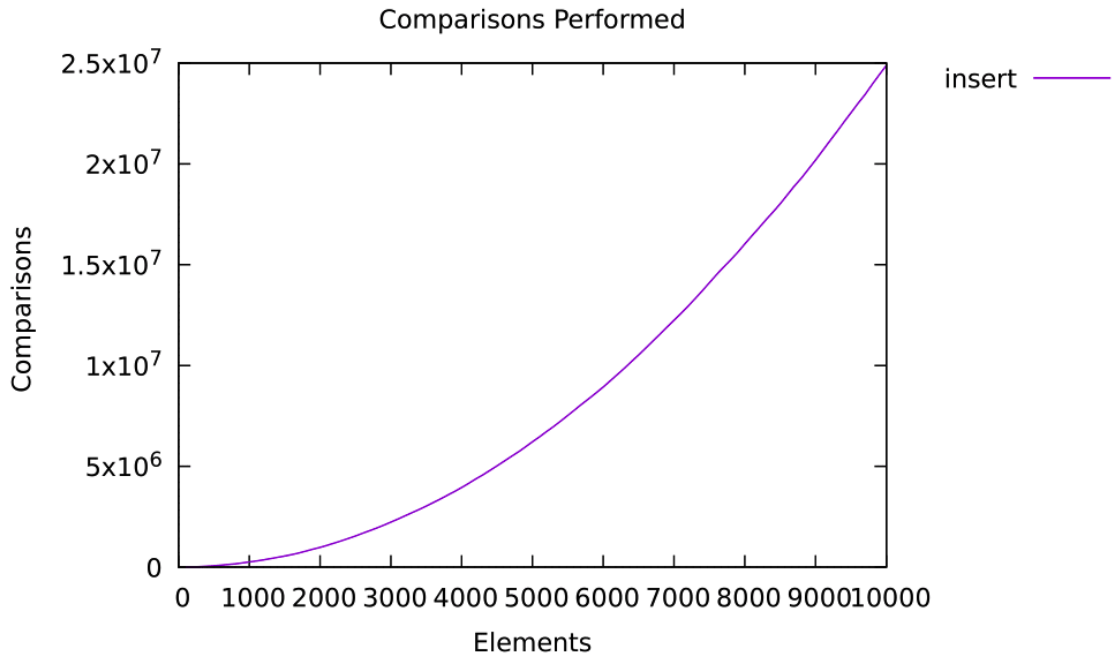
The number of comparisons graphs are similar to the move graphs except the three algorithms shell, heap, quick seem to use slightly less comparisons than moves. Insertion sort seems to use slightly more comparisons than moves.

6 Comparisons Performed Comparison without insert



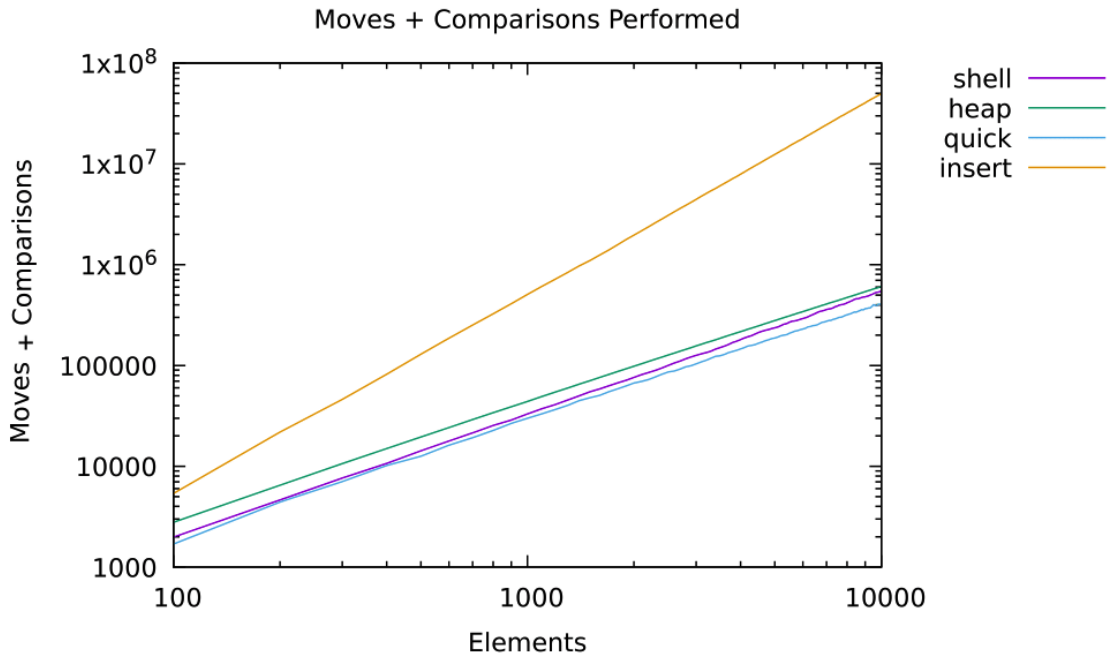
Interestingly, shell and heap sort have a very close amount of comparisons performed at various different amounts of elements. Quick sort seems more efficient using less comparisons. We see that heap sort still has a smooth curve which reinforces that it has a constant time complexity that has no bad or good cases. It still looks like the complexities of these algorithms are n(log(n)).

7 Insertion Comparisons Performed



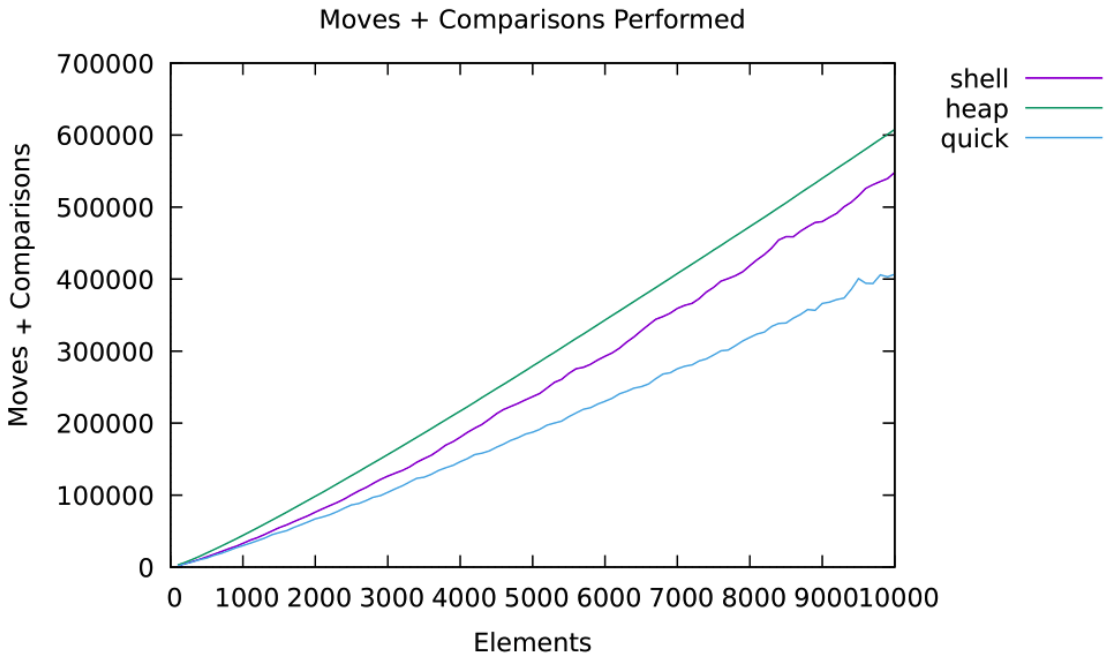
Insertion comparison looks very similar to the moves one. It still looks like it has a complexity of n^2.

8 Both Performed Comparison



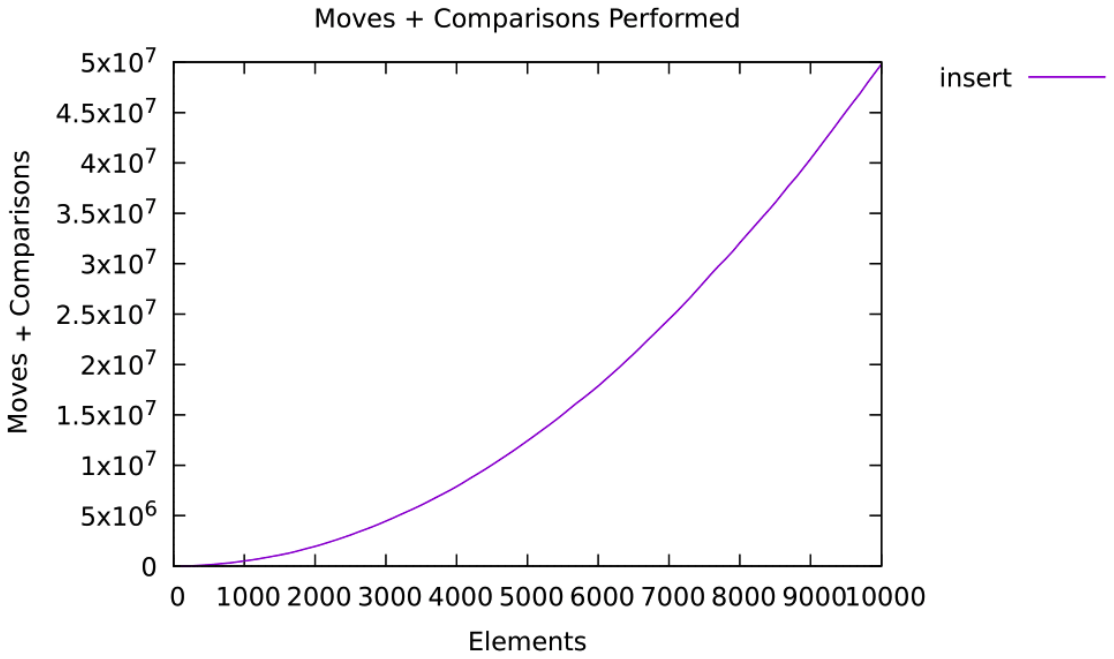
With these graphs, I combined the amount of moves and comparisons used per sort to average out the amount of operations used to sort the arrays. These graphs give the most accurate info on the average complexities of these algorithms because it accounts for most of the operations. This graph still shows us that shell, heap, and quick sort of similar complexities while insert has a much less efficient complexity.

9 Both Performed Comparison without insert



Once again, shell and heap have a similar performance while quick is more efficient. While the difference in this graph looks significant, when you look at the graph with insertion sort you see that the three sorts have a similar complexity.

10 Insertion Both Performed



This graph once again shows insertion has a clearly exponential complexity. It looks the most like n^2 so this supports the idea that it's complexity is n^2.

11 Conclusion

Using these different graphs we come to a conclusion. Insertion sort has an average complexity of n^2. Heap, shell, and quick sort have an average complexity of n(log(n)). Heap sort seems to have no bad or worse cases based on its perfectly smooth graph. Shell and quick sort probably have worse or better cases because of how they slightly fluctuate off the average.