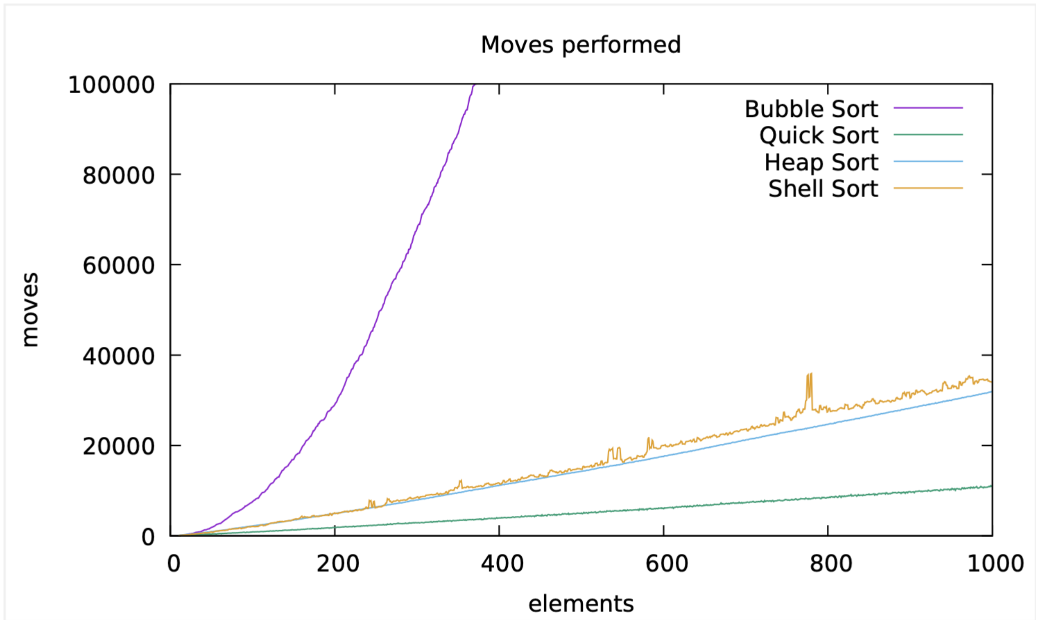
Writeup Assignment 4

Tanya Gyanmote

cruzid-tgyanmot

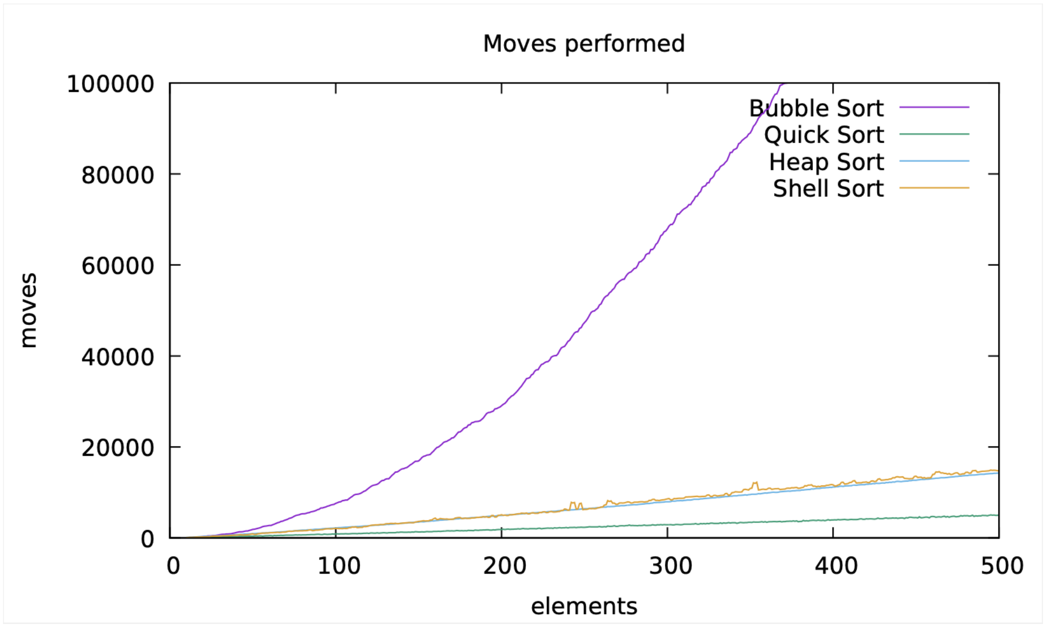
10/23/22

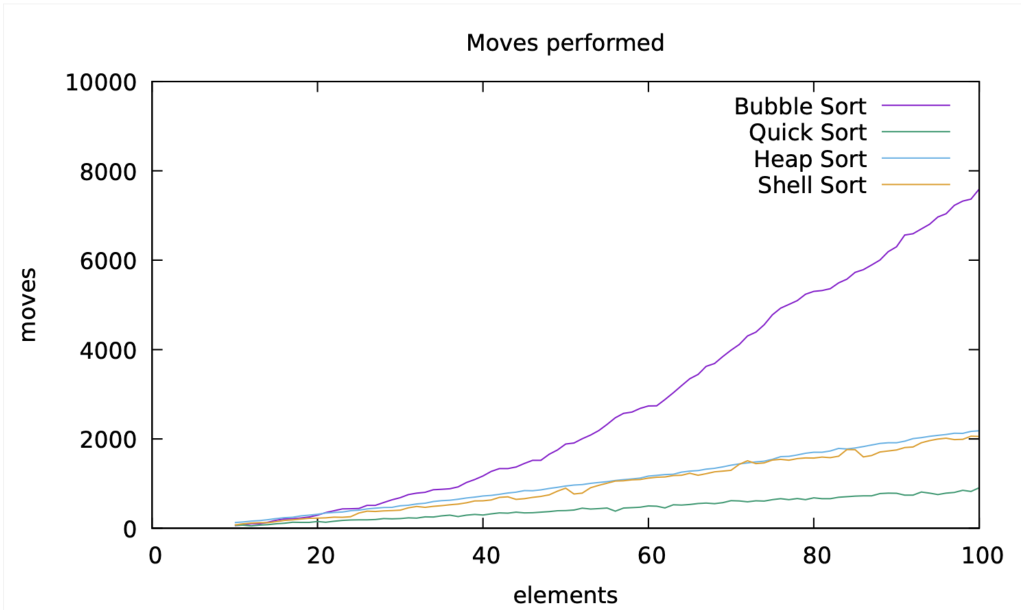
- I learned from the different sorting algorithms that mergesort is a comparison-based algorithm that merges two pre-sorted arrays into a single sorted array. Bubble sort is a comparison-based algorithm that has a repeating process until it’s sorted, compares each pair of elements in an array, and swaps them if they are out of order. Heapsort is a comparison-based algorithm that sorts elements using a binary heap data structure. It divides the array into sorted and unsorted parts, then shrinks the unsorted region iteratively by extracting the largest element and moving it to the sorted area. Quicksort is a comparison-based algorithm that sorts an array using divide-and-conquer. A pivot variable divides the array into two subarrays, where all elements are less than the pivot and subarray. It repeats till all elements are greater or equal to the pivot. The conditions in which the sorts perform well are that for Bubble if the first swap was sorted, the array is always in order, the time complexity would be linear, and the worst case would be if the array is reversed. For merge sort, the best conditions would be the same as the poor conditions because you are diving an array into subarrays. For quick sort, the best condition would be if the median was found correctly, and the poor condition would be if the array is in reverse order. For heapsort, the best condition would be the same as the poor condition because the heap structure takes a long time. The conclusion I made from my findings is that Quick Sort was the most effective and quickest.



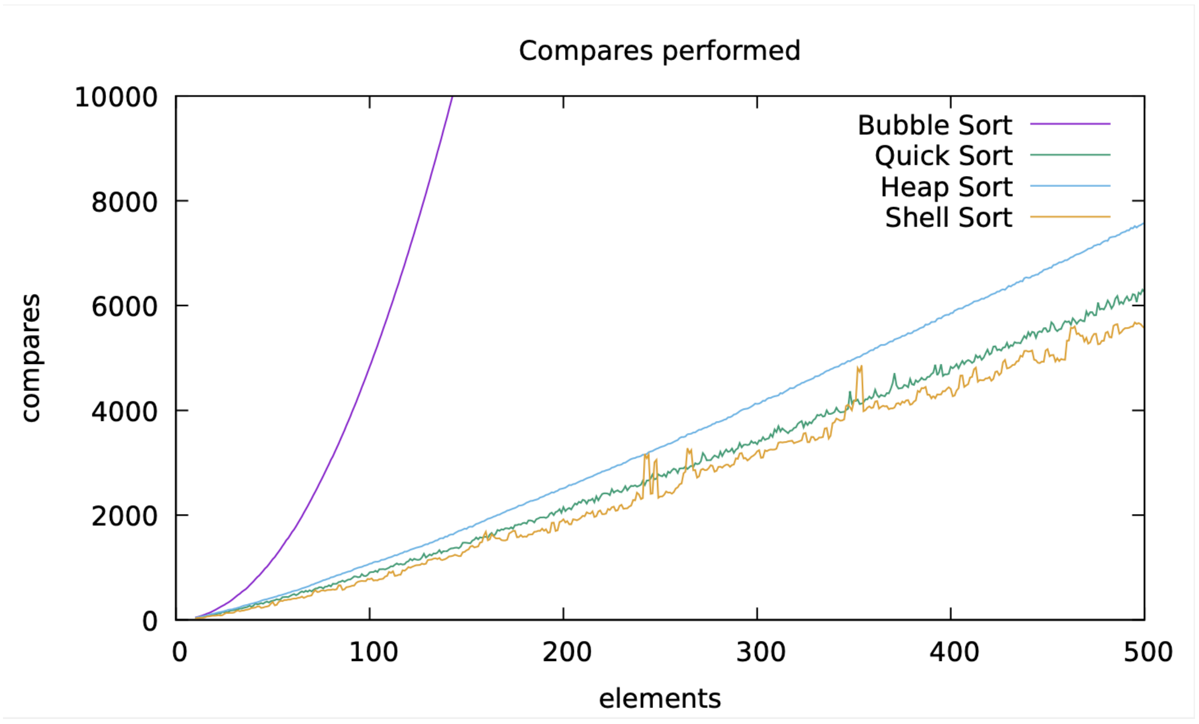
- With the graphs I produced, arrays with smaller numbers versus bigger numbers. I used a large array to signify an array in reversed order and a small array to reflect a mostly sorted array. From the graphs, you can see that the more elements there are every function gets affected by the number of compares and moves. Both the charts below show that bubble sort has an exponential increase. In the charts, you can see that Bubble sorts use the most moves, as the data for it shoots straight up, compared to heap and shell, which have more of a linear data set, and lastly you can see that quick sort is the most effective as it has the least number of moves. Even with an increase in elements, the data graphed seems to stay the same and have a similar pattern, as with the bubble sort it looks like it has exponential growth. You can see from the graph that heap and shell both have similar data points, making them closer in efficiency.

- The conclusions I had come up with from these graphs are that when it comes to comparing the sorts, not including bubble sort they are similar but Quick sort has the least amount of moves, so its the best option if the array is large and unsorted, but if the array is small and sorted then bubble sort would be the best.





- Similarly, comparing the different sorts with the number of comparisons from moves, you can see that the bubble sort takes the most compares, which makes it less effective than the other sorts. Also, you can see that Quick sort is one of the most effective and has the least number of moves, but more compared. You can see from the graph that heap and shell both have similar data points, making them closer in efficiency.


Compares performed
Bubble Sort
Quick Sort
Heap Sort
Shell Sort
compares
elements