Writeup

Assignment 3: Sorting

Time complexity:

Sorting Algorithm	Best Case	Worst Case
Bubble Sort	$O(n)$	$O(n^2)$
Shell Sort	$O(n\log n)$	$O(n^2)$
Quick Sort (Stack)	$O(n\log n)$	$O(n^2)$
Quick Sort (Queue)	$O(n\log n)$	$O(n^2)$

What I learned about the sorting algorithms:

I learned that there are many more ways to sort arrays using comparisons than I thought. I found Bubble Sort to be both an interesting and intuitive comparison-based sorting algorithm.

I found Shell Sort to be less intuitive than Bubble sort, and I learned that its time complexity, performance, and functionality is entirely dependent on the gap sequence.

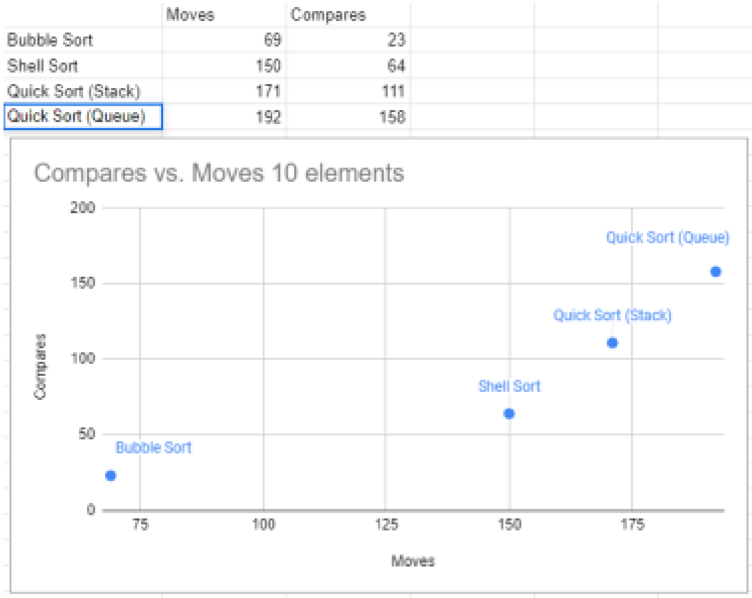
I learned that Quick Sort is clearly the overall *best* comparison-based sorting algorithm, and I also found it to be the most difficult to understand. I learned how Quick Sort effectively makes use of Stack and Queue data structures.

How I experimented with the sorting algorithms:

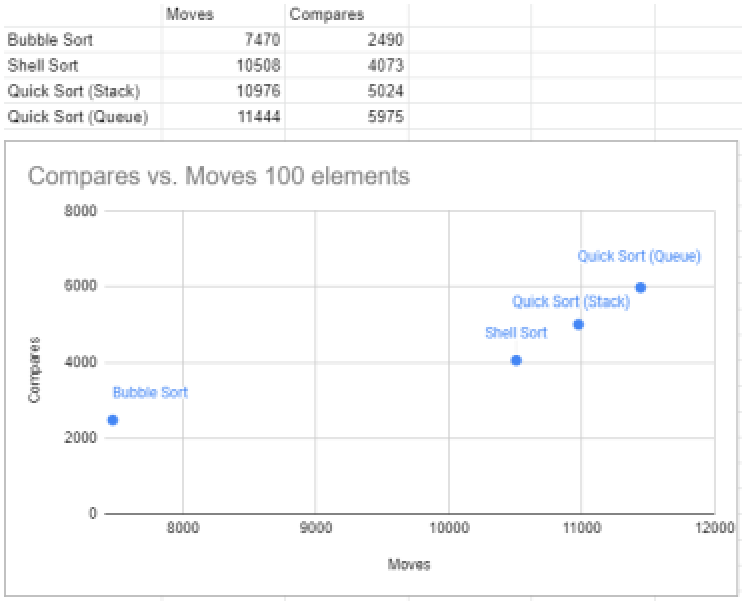
I made sure the random seed has no consistent or unexpected effect on any of the sorting algorithms. I compared the sorting algorithms to each other in all ways, especially in terms of performance, number of moves, and number of comparisons. I tested each of the sorting algorithms with array sizes of varying orders of magnitude as shown by the graph below.

↓

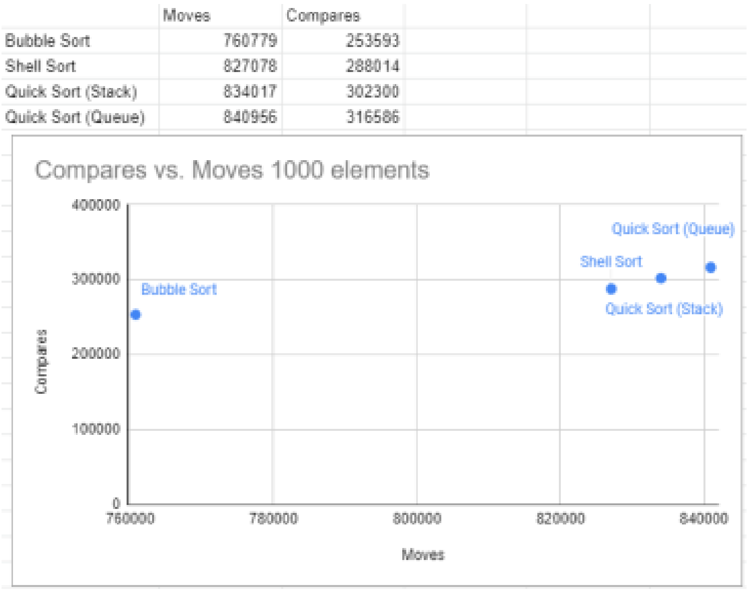
Graphs:



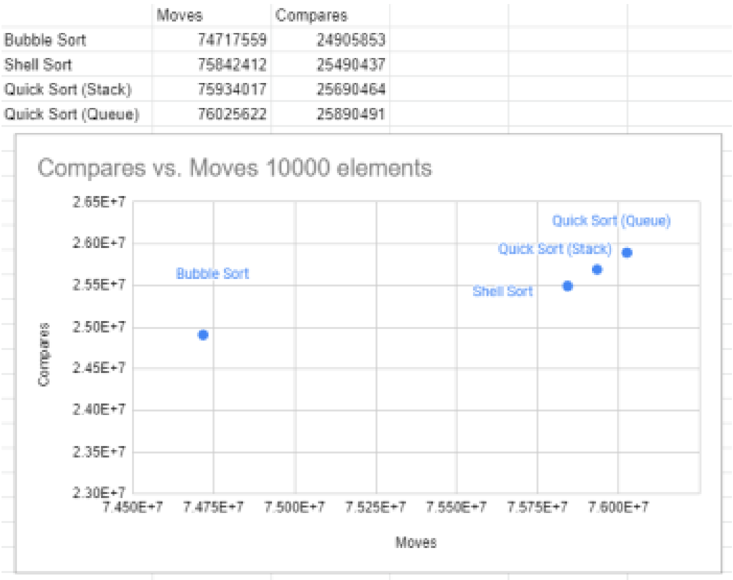
	Moves	Compares
Bubble Sort	69	23
Shell Sort	150	64
Quick Sort (Stack)	171	111
Quick Sort (Queue)	192	158



	Moves	Compares
Bubble Sort	7470	2490
Shell Sort	10508	4073
Quick Sort (Stack)	10976	5024
Quick Sort (Queue)	11444	5975



	Moves	Compares
Bubble Sort	760779	253593
Shell Sort	827078	288014
Quick Sort (Stack)	834017	302300
Quick Sort (Queue)	840956	316586



	Moves	Compares
Bubble Sort	74717559	24905853
Shell Sort	75842412	25490437
Quick Sort (Stack)	75934017	25690464
Quick Sort (Queue)	76025622	25890491

Analysis:

The four sorting algorithms perform similarly when the size of the array is low, but as the size of the array increases, Bubble Sort becomes less feasible and Quick Sort becomes the most attractive sorting algorithm by far. The number of moves and comparisons made by Quick sort to not increase exponentially compared to the size of the array, nor do they even increase linearly. The number of moves and comparisons made by Quick Sort only increase logarithmically, which makes Quick Sort arguably the best known sorting algorithm.

Also, after examining the program output, I observed that the stack and queue size are directly related to the size of the array. As the number of elements increases, so do the max stack and max queue sizes.