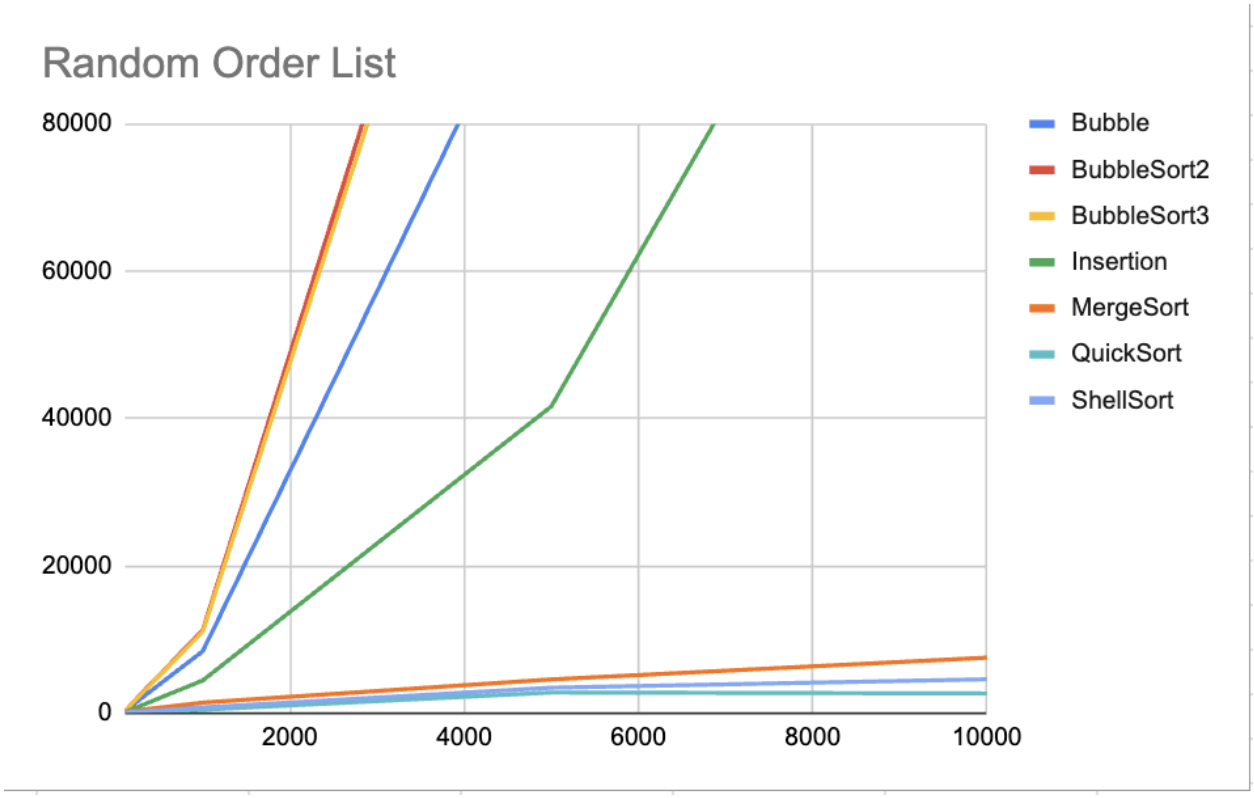
Lyla Shami
Professor Arian
Sunday, July 30th

Lab 3 Report

Random Ordered List

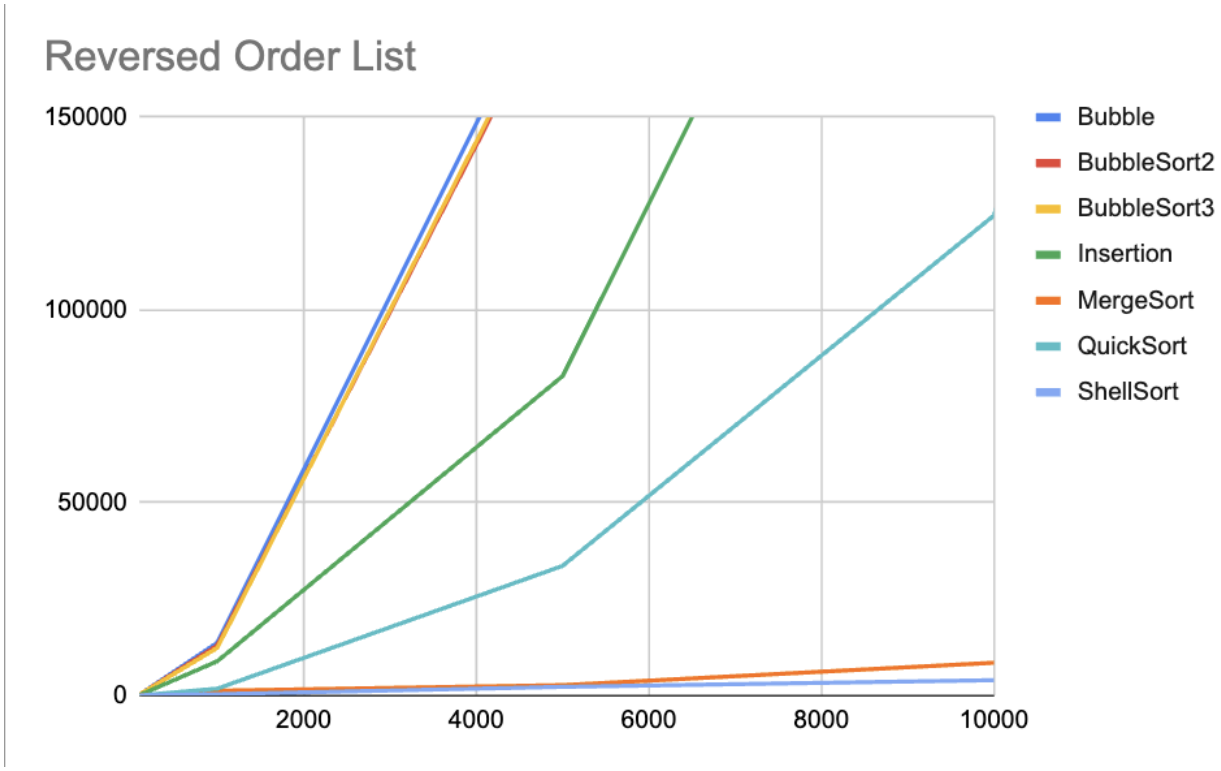
For my first graph, I ran all of my sorts which were: Bubble Sort, Insertion Sort, Merge Sort, Quick Sort, and Shell Sort with a randomly ordered list. Here is the graph with all the runtime measurement results:



Analysis for randomly ordered: As you can see, the one that took the longest to run would have been the first three sorts Bubble sort, Bubble sort 2, and Bubble sort 3. I believe those took a long time because of the time complexity being o(n^2). Insertion Sort was a little faster than Bubble Sort but was still because the average time complexity for insertion sort is also o(n^2) it did not perform much faster than insertion sort. The one sort that performed at the fastest speed with the randomly ordered set was quicksort. Quicksort did a bit faster than merge and shell sort as the numbers got bigger, from 100 to 1000 to 5000, it makes sense since the time complexity for merge sort is o(nlog(n)).

Reversed Ordered List:

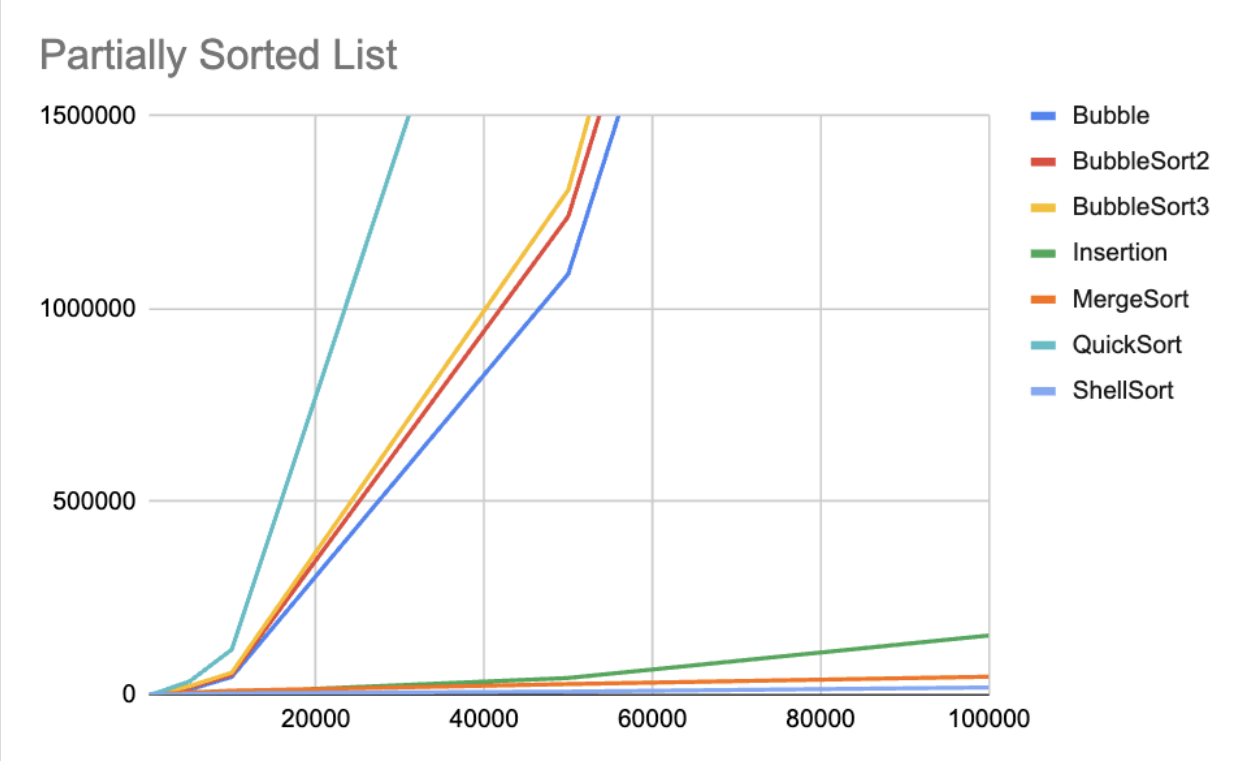
For my second graph, I ran all of my sorts which were: Bubble Sort, Insertion Sort, Merge Sort, Quick Sort, and Shell Sort with a reversed ordered list. Here is the graph with all the runtime measurement results:



Analysis for reversed ordered: Again this time Bubble Sort, Bubble Sort 2, and Bubble Sort 3 took the longest to run, which you can see, with this list it makes sense as their time complexity is o(n^2) that it would take long for it to run. The fastest sort with the reversed list would have to be the Shell sort. When running with shell sort, it took the fastest time to run, this would make sense as its time complexity is o(nlogn).

Partially Ordered List:

For my third Graph, I ran all of my sorts which were: Bubble Sort, Insertion Sort, Merge Sort, Quick Sort, and Shell Sort with a partially ordered list. Here is the graph with all the runtime measurement results:



Analysis for partially ordered: This time as you can see from the graph, bubble sort was slightly faster when it came to the time complexity. It wasn't the fastest but compared to the other recorded times for each list, bubble sort was a little faster in sorting. I believe this is because when bubble sort best case in time complexity is o(n^2), which is essentially when the list is already sorted, so when the list is already partially sorted, the runtime is slightly faster. The sort that had the fastest time with a partially sorted list would be again shell sort.