Jessica Runandy
November 21, 2019
Program 4: Sorting Algorithms Performance Times

Data Size	Sorting Algorithms Performance Times in milliseconds					
	Bubble Sort	**Insertion Sort**	**Merge Sort**	**Iterative Merge Sort**	**Quick Sort**	**Shell Sort**
50	0	0	0	0	0	0
100	0	0	0	0	0	0
500	15	0	0	0	0	0
1000	31	16	0	0	0	0
5000	672	266	15	0	0	0
10000	2687	1047	47	15	0	15
15000	6027	2406	93	16	0	16
20000	10687	4235	125	31	15	16
25000	16375	6312	188	31	16	16
50000	62969	23602	649	78	31	63
75000	130953	41719	1343	125	47	93
100000	217359	60140	2391	172	62	133

Figure 1: Performance times of bubble sort, insertion sort, merge sort, iterative merge sort, quick sort, and shell sort depending on the different data sizes.

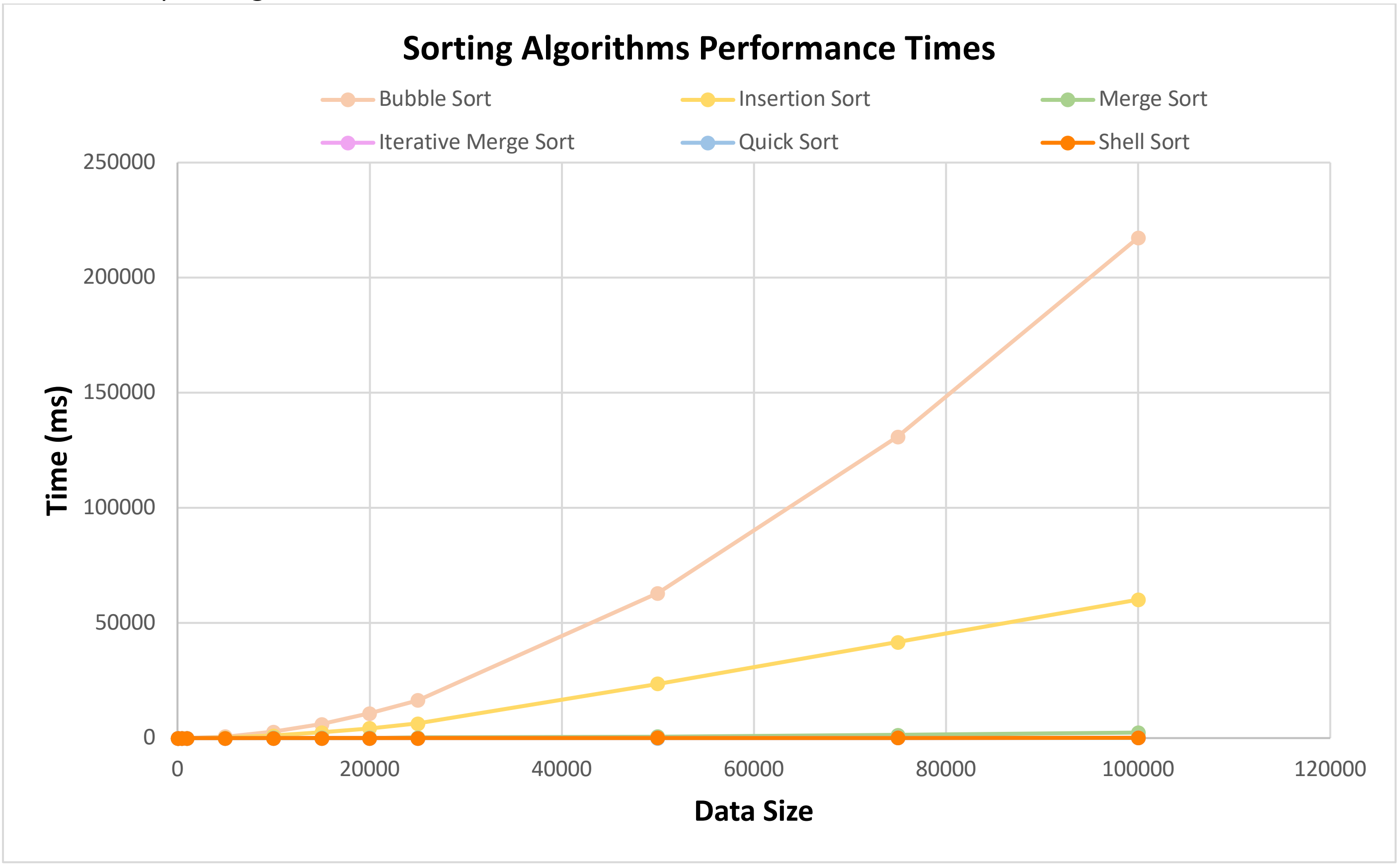


Figure 2: Comparing the performance times of the three slowest sorting algorithms (bubble sort, insertion sort, and merge sort).

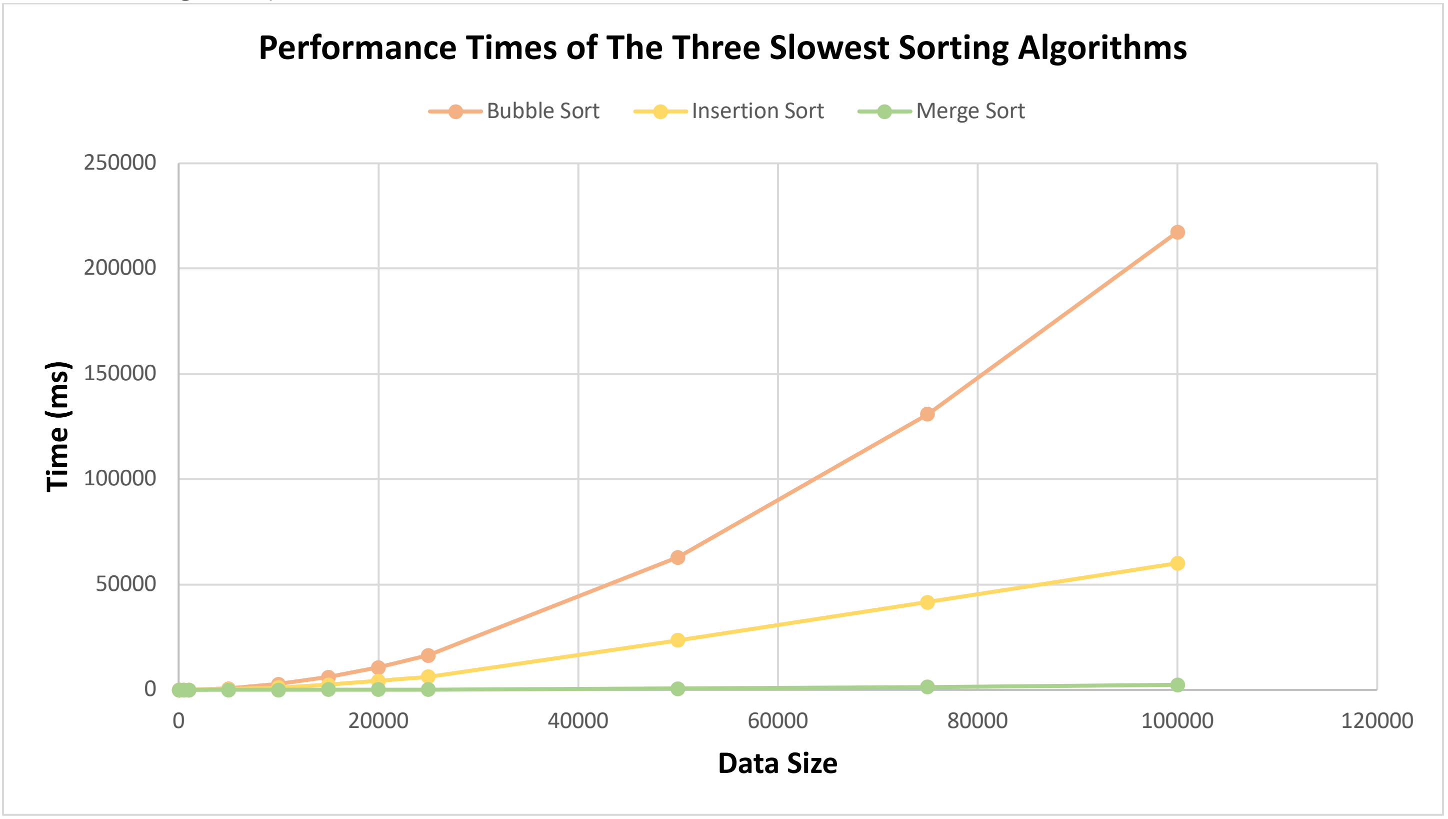


Figure 3: Comparing the performance times of the three fastest sorting algorithms (merge sort, quick sort, and shell sort).

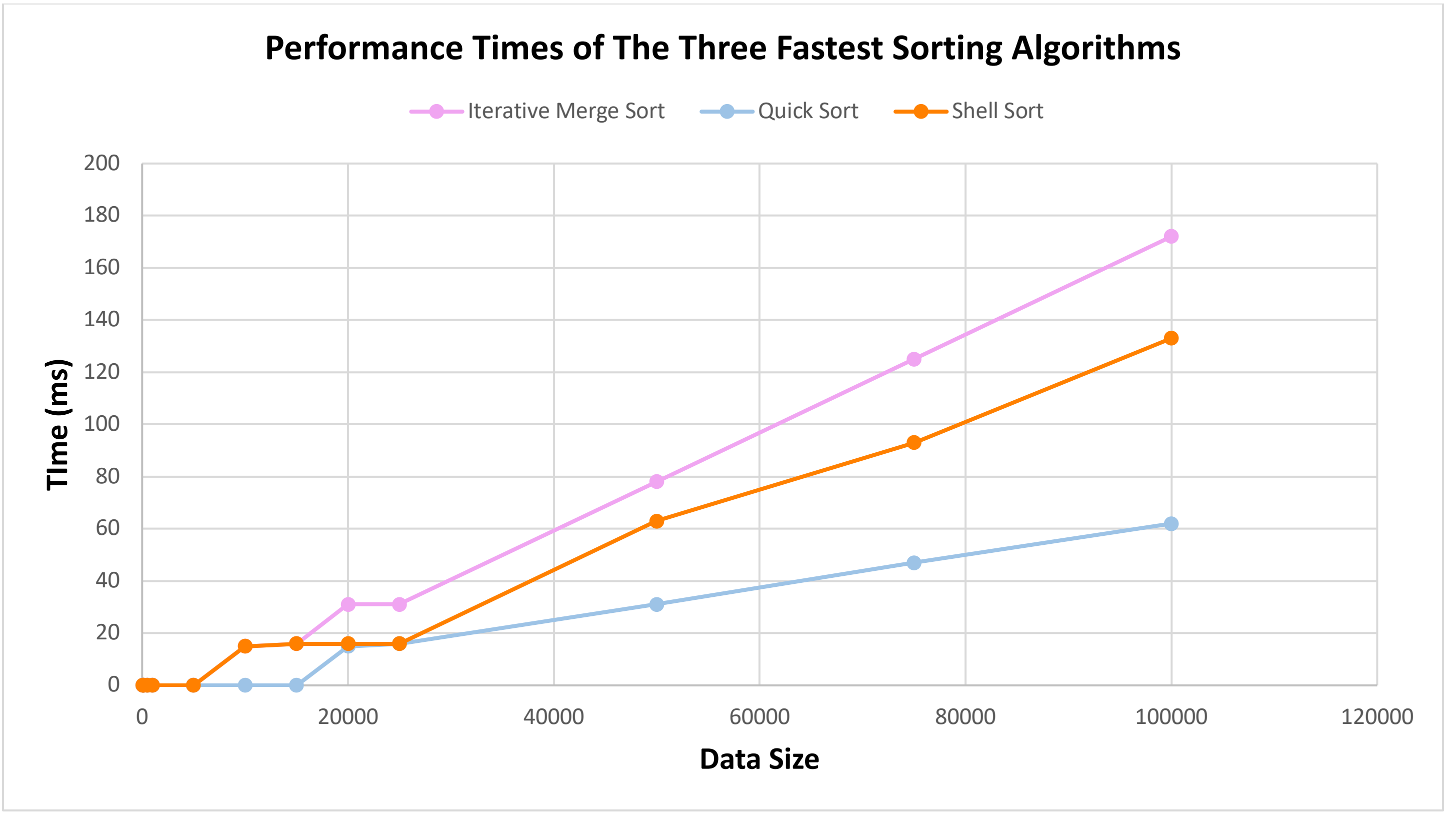


Figure 4: Comparing the performance times of merge sort, iterative merge sort, and quick sort.

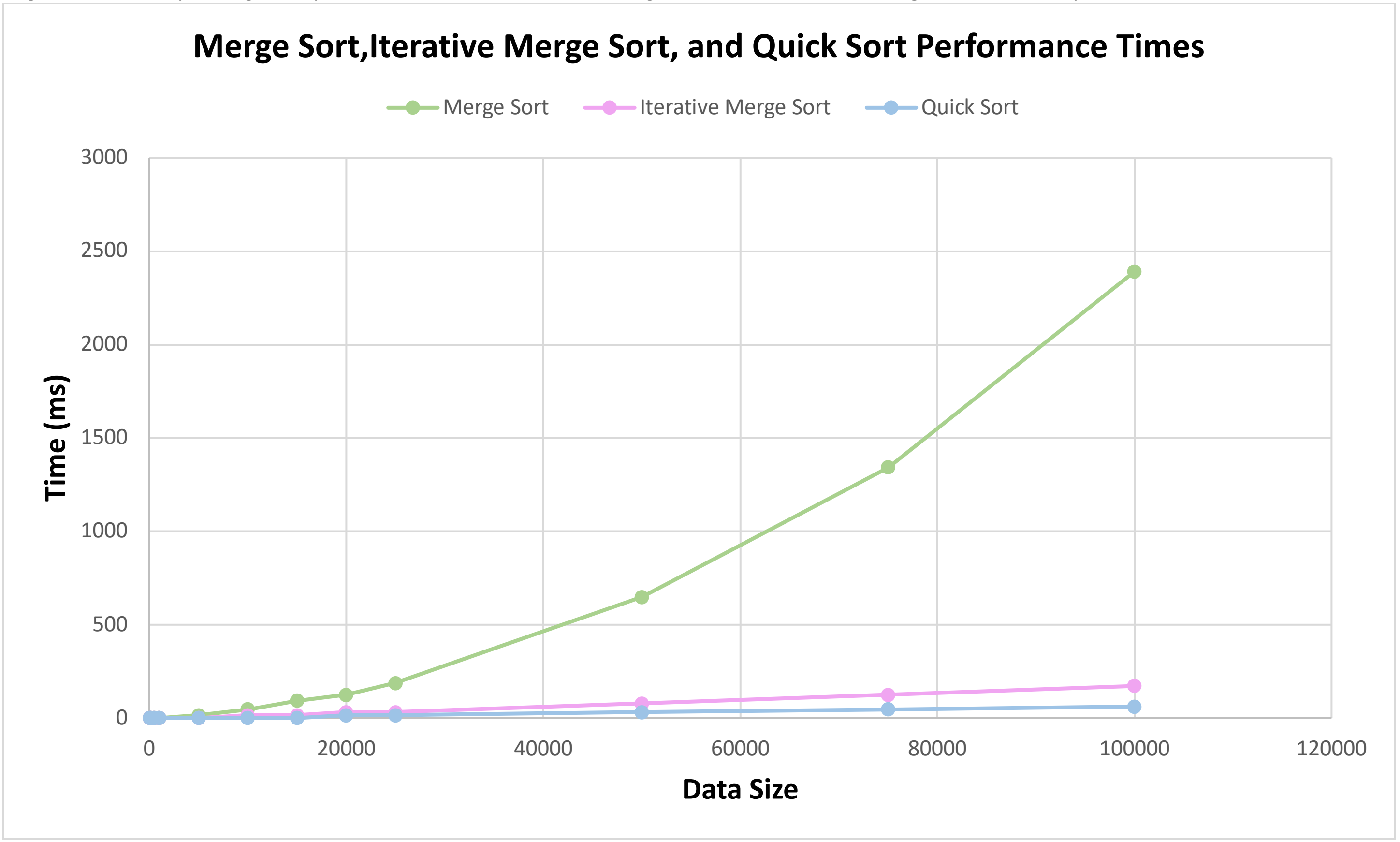
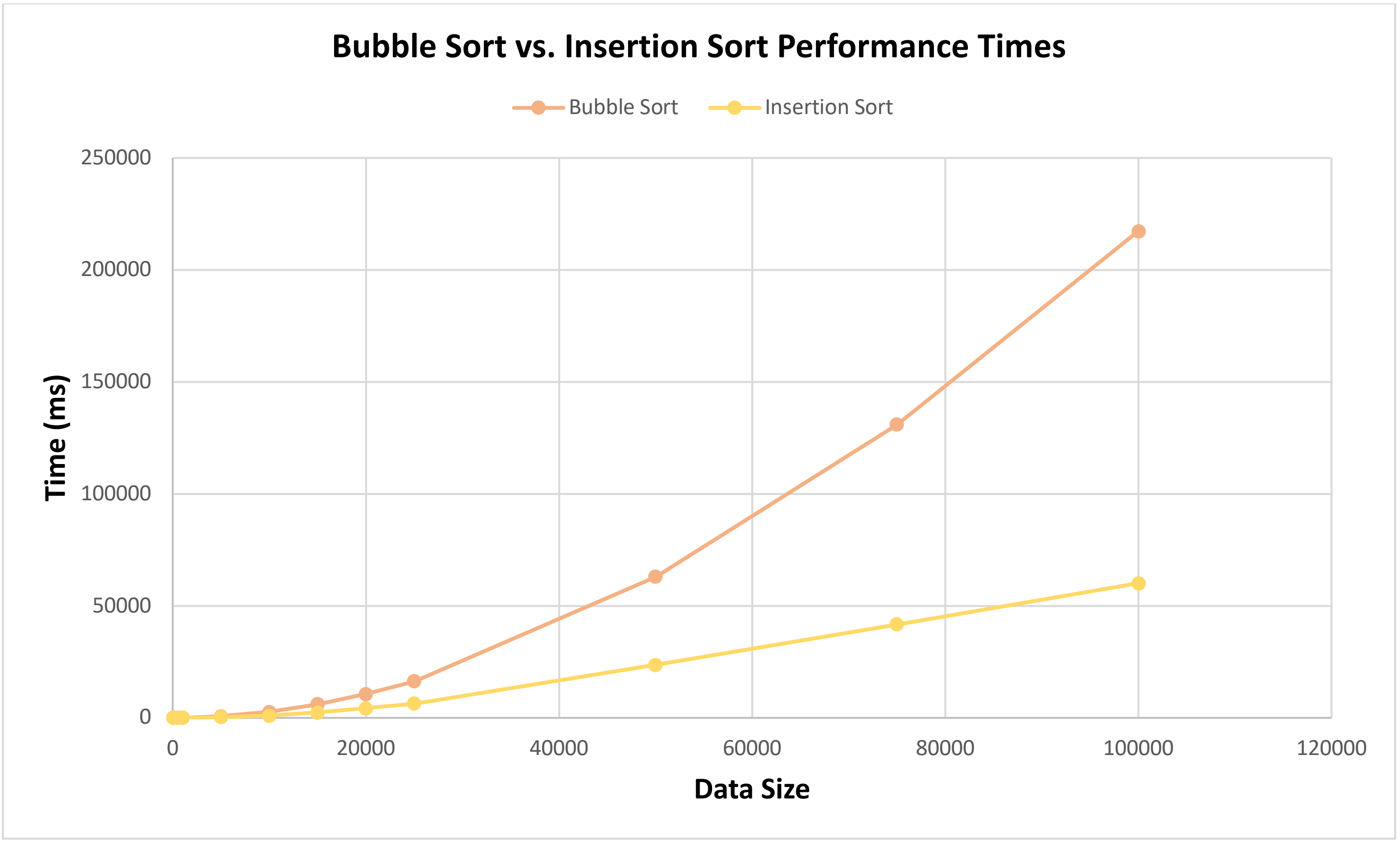


Figure 5: Comparing the performance times of bubble sort and insertion sort (both have the same complexity of $O(n^2)$).



Summary:

Based on the performance times of bubble sort, insertion sort, merge sort, iterative merge sort, quick sort, and shell sort, bubble sort is generally the slowest sorting algorithm while quick sort is the fastest. Bubble sort is the slowest algorithm with the complexity of O(n^2). Even though bubble sort and insertion sort have the same complexity of O(n^2), insertion sort is faster because the bubble sort usually requires several passes over the data, and it compares items next to each other. Bubble sort continuously swaps adjacent items until the data is sorted. On the other hand, insertion sort takes one item at a time and places it in the correct sorted spot of the sorted region until the data is sorted, which allows for a faster sorting time compared to bubble sort.

Although merge sort and quick sort is both O(nlogn), merge sort is slower than quick sort because merge sort uses recursion to sort the data. In addition, merge sort allocates a temporary array at each recursive call, which builds up the stack. Quick sort uses partitions and pivot to separate the array into subarrays, resulting in fewer recursive calls. Quick sort is generally the fastest sorting algorithm with O(nlogn) for the average case and O(n^2) for the worst case, but the efficiency depends on the pivot. Iterative merge sort is a non-recursive method, which does not allocate an additional array on each recursive call. Instead, it creates one additional array and copies the data back and forth between the original and temporary array to sort the data. The data is copied back and forth between the original and additional array as many times as O(logn), which is faster than merge sort.

The shell sort has an average case of O($n^{3/2}$) and a worst case of O(n^2). It is one of the faster sorting algorithms because it uses gap sizes to constantly move the data over large distances quickly. It is faster than bubble sort, insertion sort, merge sort, and iterative merge sort.