Andrew Melland
CS3353
9/14/2020

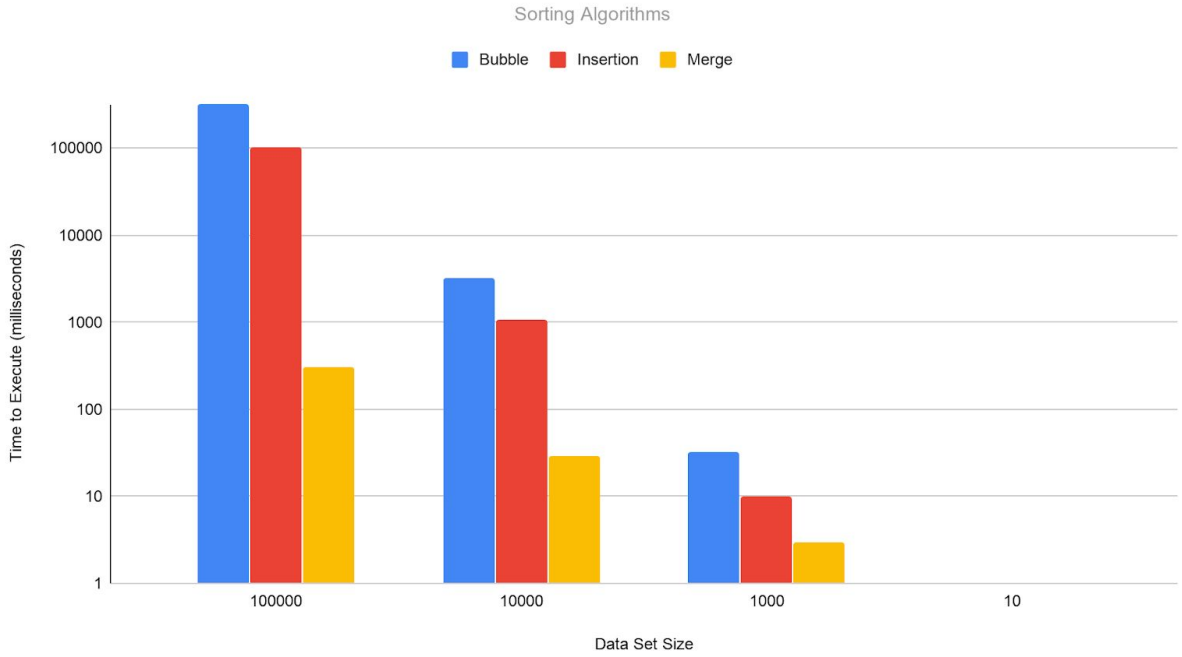
Lab 1

Data

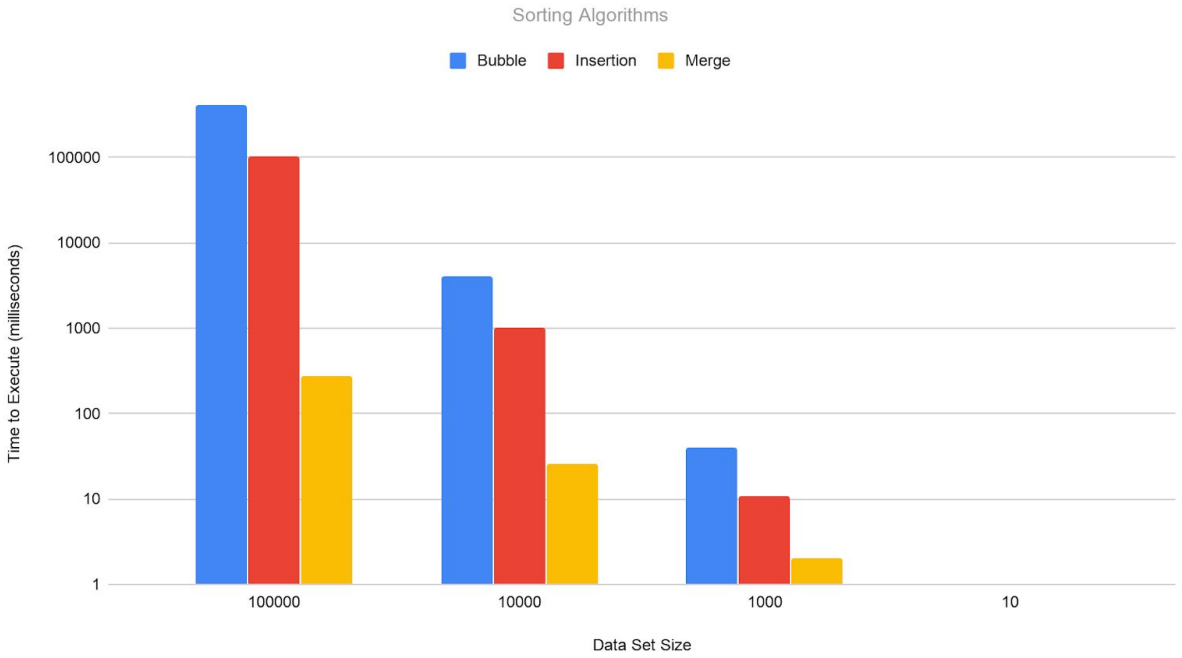
DataSet Type	Random		
Algo Used	Bubble Sort	Insertion Sort	Merge Sort
10	0	0	0
1000	32	10	3
10000	3203	1082	29
100000	320112	103575	304
DataSet Type	Reverse Sorted		
Algo Used	Bubble Sort	Insertion Sort	Merge Sort
10	0	0	0
1000	40	11	2
10000	4076	1030	26
100000	409004	102714	278
DataSet Type	20% Unique		
Algo Used	Bubble Sort	Insertion Sort	Merge Sort
10	0	0	0
1000	32	13	2
10000	3176	1056	27
100000	319503	101846	296

DataSet Type	Partially Sorted		
Algo Used	Bubble Sort	Insertion Sort	Merge Sort
10	0	0	0
1000	25	11	2
10000	2539	1022	27
100000	255020	99218	294

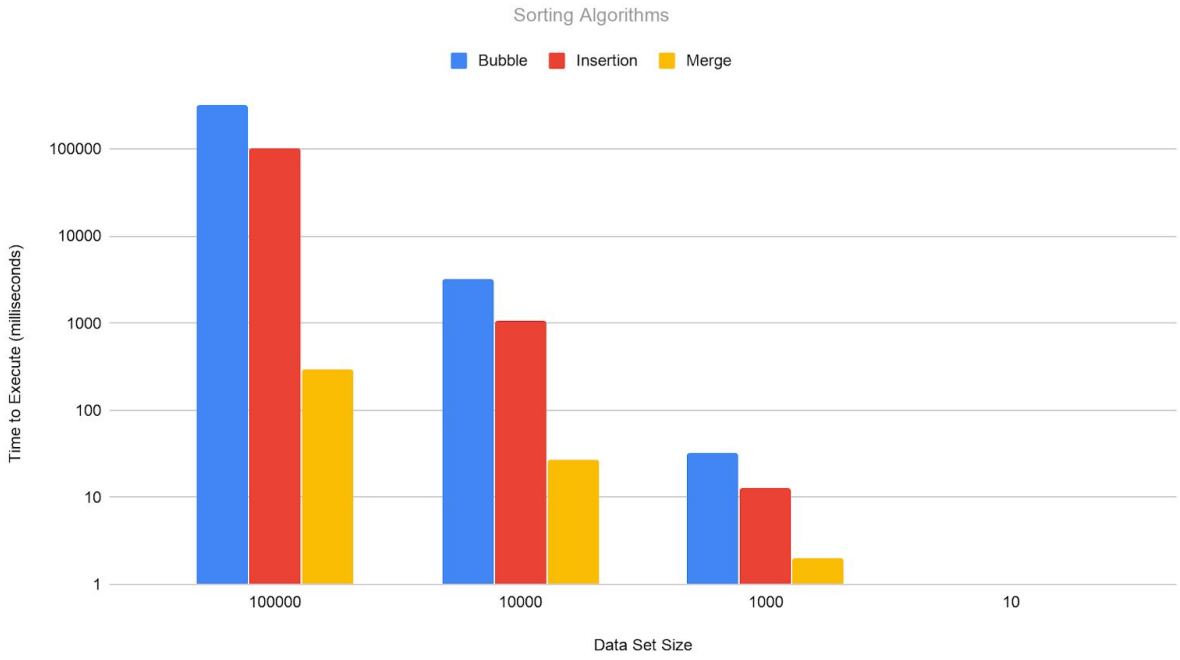
Random



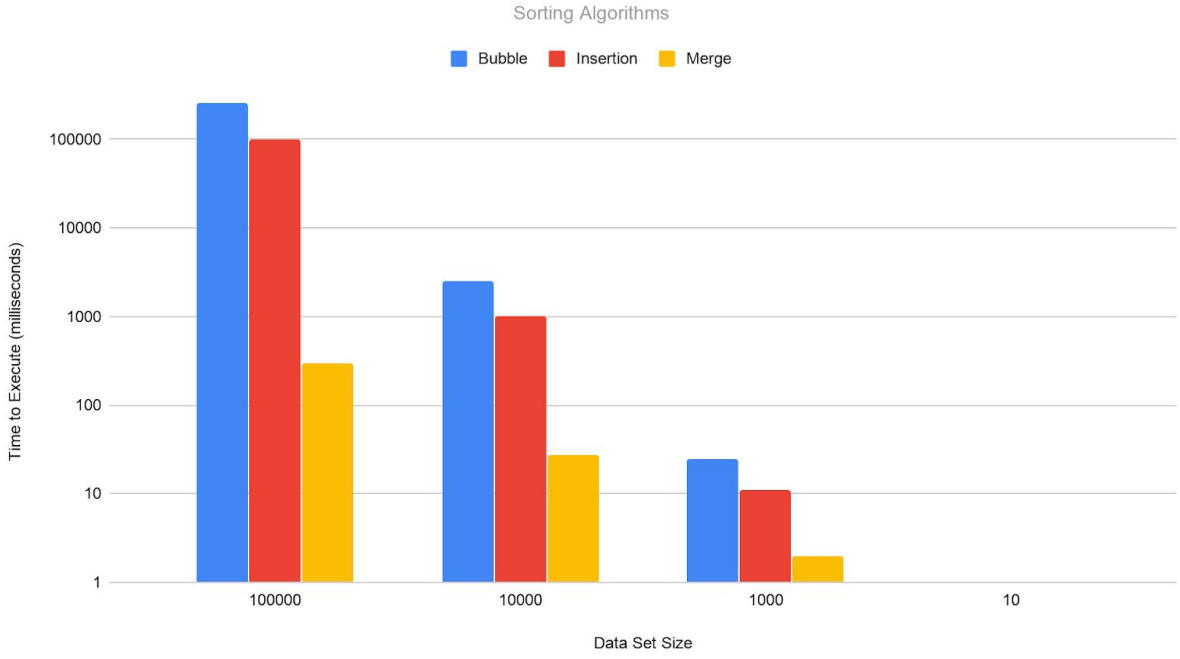
Reverse Sorted



20% Unique



Partially Sorted



Analysis

For this lab, we used three common Sorting algorithms (Bubble, Insertion, and Merge) and sorted a total of sixteen data sets in order to see which sorting algorithm is best for what kind of data. The different data set types were random numbers, numbers in a reverse sort, numbers consisting of only twenty percent unique values, and a data set where only thirty percent of the data is out of order. The 4 data set types consisted of four different sizes (10, 1000, 10000, and 100000) of elements. To analyze the data, I plotted the times of the sorts on a logarithmic scale because the original range of values was too large to visualize and compare with each other. Based on times from the graphs and charts above, in all cases tested the merge sort is a much better algorithm to use for sorting. However, since the high-resolution clock, which is a part of the Chrono class, could not measure the short period of time that it takes to execute the sorts on a data set size of only 10, we cannot deduce from our results which algorithm is faster for the smallest data size. However, we can infer that the bubble sort would be the fastest algorithm for the smallest data set types because bubble sort is very good with partially sorted small data sets and doesn't use recursion or extra arrays like merge that have a high flat cost to computation time.

The bubble was the worst sorting algorithm used as it has an average bigO of $O(n^2)$. This hypothesis was proven for all data sets. The insertion sort has the same big O as the bubble sort in both the best and worst case, however, the insertion sort out preformed the bubble sort on all data sets. The merge sort in general is a much faster sorting algorithm than both bubble and insertion. This is because the merge sort has a bigO of O(n*log(n)) in all cases. The merge sort outperformed bubble and insertion so well that its time is not even visible on the graph unless the vertical axis is a logarithmic scale. A bigO of O(n*log(n)) will outperform a bigO of $O(n^2)$ which is why the merge sort is faster than both bubble and insertion on all data sets tested.

As for what effect the type of data was used had on the sorting times of the algorithms, Bubble sort was much more greatly affected than the other two sorting algorithms. Both insertion and merge are barely affected by the type of data used, benefiting most from the partially sorted list. Bubble sort on the other hand was almost twice as fast when sorting the partially sorted algorithm when compared to sorting its worst-case scenario, the reverse sorted list.

asd