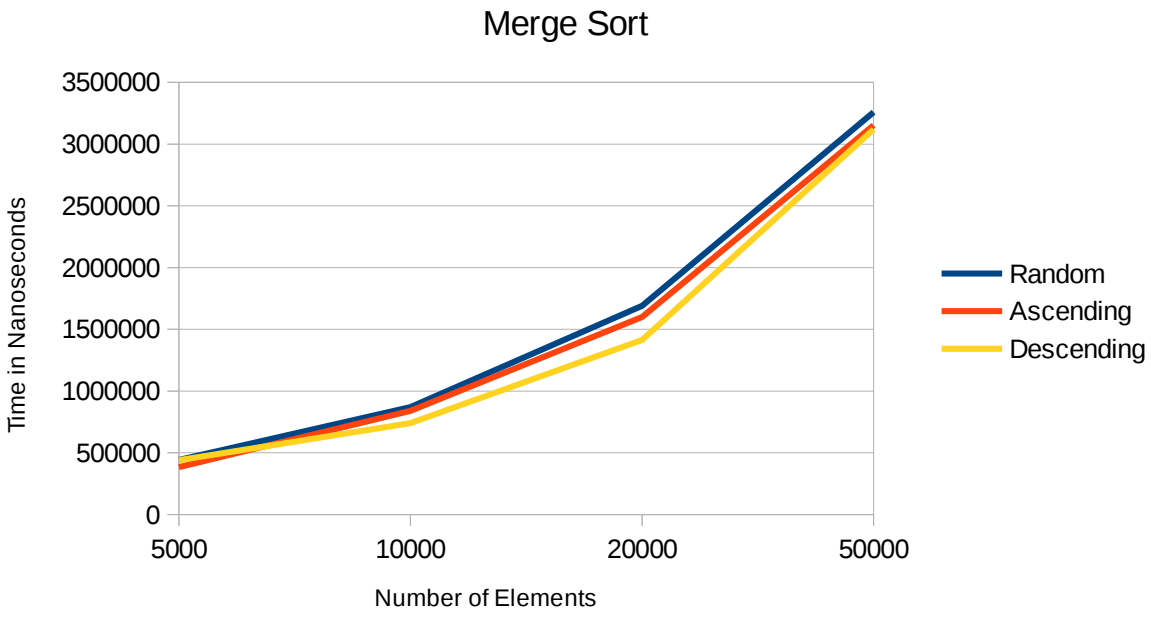
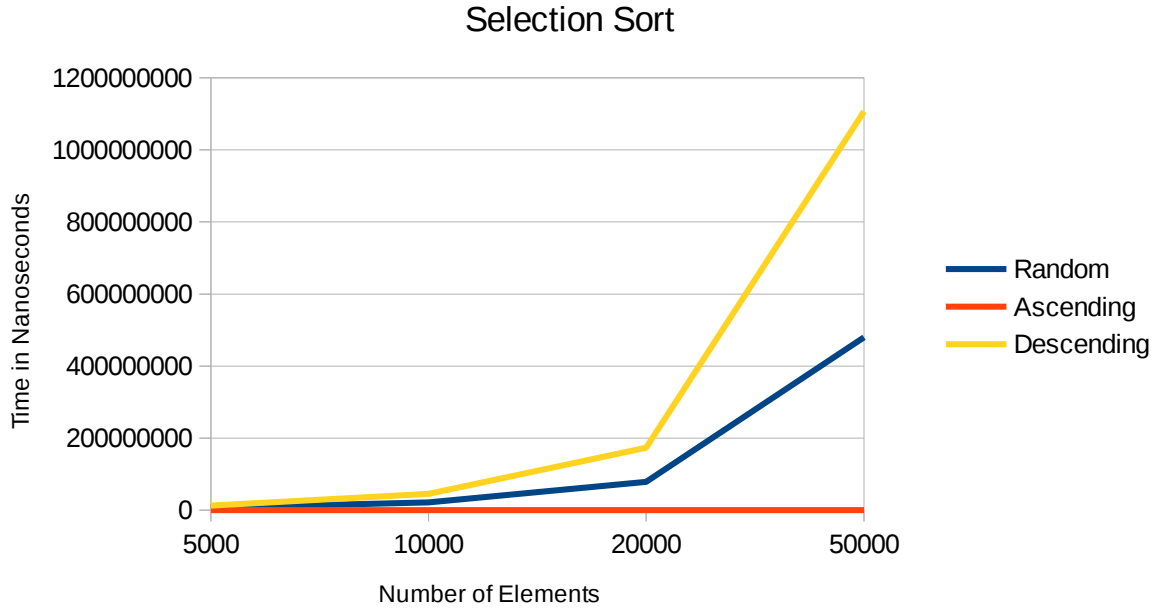
Merge Sort				
Elements	5000	10000	20000	50000
Random	441720	869541.66	1690895.82	3257681.12
Ascending	383073.71	839602.14	1598641.43	3151475.12
Descending	440460.75	739903.17	1414479.3	3118960.88



Selection Sort				
Elements	5000	10000	20000	50000
Random	6196503.6	21321028.76	78286489.57	479909184.89
Ascending	51072.08	45148.2	49243	67561.54
Descending	12252013.66	45207746.03	173406251.29	1106892864



Tao Ling

Data collection

The data was gathered by running 100 trials on each and all cases (ascending, descending, and random) of the input file sizes of 5000, 10000, 20000, and 50000. The data above is the collected average of these trials in nanoseconds.

Relative run times

When we compare the run times of insertion sort and merge sort in the graphs above we can see that merge sort takes significantly less time than insertion sort in both random and descending lists. In ascending lists we see that insertion sort is much faster than merge sort. This is due to the fact that insertion sort is O(n) in the best case which is ascending lists, and O(n^2) in the worst and average case. When we compare this to merge sort which has a best case of O(n log(n)) in all cases, best, average, and worst; we can see that merge sort is a much better option than insertion sort in terms of time complexity in most cases.

Effect of the order of the data

The major point to take away here from the effect of the order of data when comparing insertion and merge sort is that in a sorted list in ascending order, insertion sort will beat out merge sort due to the fact that it is already sorted and thus will make less comparisons. In contrast, merge sort when operating on the same sorted ascending list will split and compare the smaller lists thus taking more time on a list that is already sorted. However, in any other case besides this one, merge sort will be the faster algorithm.

Size of the files

The size of the files had different impact on both algorithms. Because merge sort is a divide-and-conquer algorithm, the time difference between smaller to bigger files was a gradual change. In contrast, insertion sort was severely impacted by an increase in file size due to the fact that it is O(n^2) in the worst and average case. However, in an already sorted list, the size of the file has minimal impact on the runtime when using selection sort as seen in the graph where it is almost a horizontal line. This is as opposed to merge sort which still increases at a gradual pace.

Tao Ling

Indicated computational complexity

The known complexities of merge sort and insertion sort are very different due to the fact that insertion sort is a quadratic algorithm which means it scales much faster than non quadratic algorithms. Thus, because the average, best, and worst time complexity of merge sort is O(n log(n)) while the average and worst time complexity of insertion sort is O(n^2). Merge sort is usually the better choice except in a sorted list or nearly sorted list when one is only concerned with the amount of time it takes to get a list sorted.

Conclusion

From this comparison we can see that in most cases merge sort is the better algorithm to use. However, there are some cases where insertion sort may be useful such as in short lists or in a list that is nearly sorted. Another reason why one may choose to use insertion sort over merge sort is if space complexity is needed to be taken into account as insertion sort does not cost any space, while merge sort does.