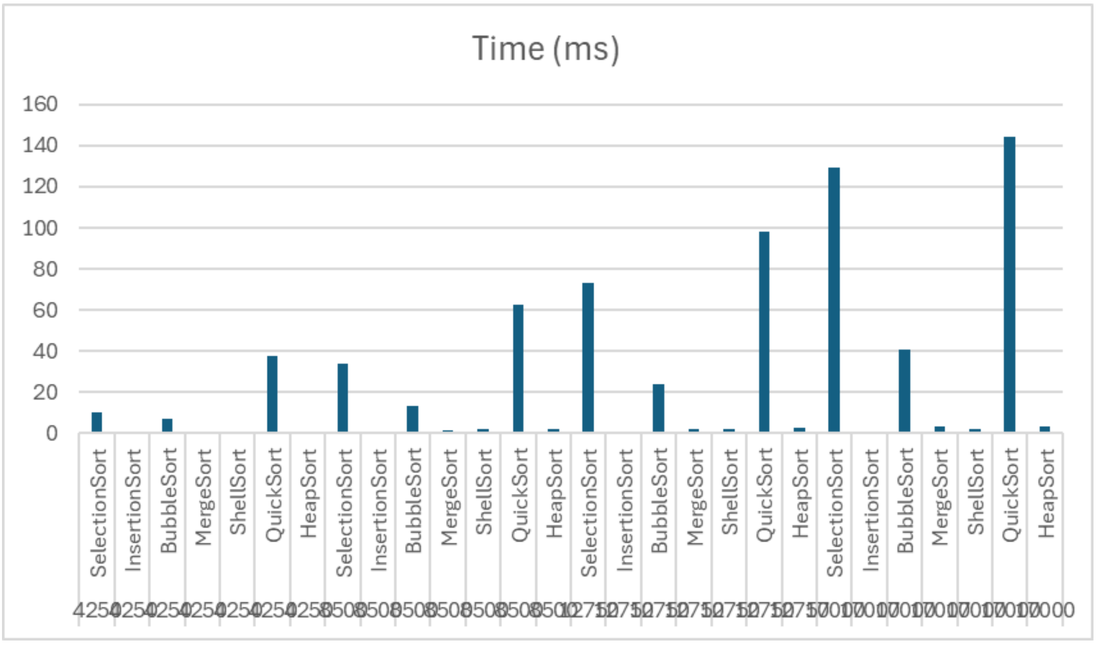
Introduction:

The outputs of seven different algorithms were tested against each other and tasked with sorting arrays filled with increments of 4,250 elements. The max number of elements in this analysis was 17,000 due to processing restrictions.

Findings:



The best performing algorithm in THIS analysis is hands down Insertion sort. Throughout the 4 trials of increasing elements, the time complexity of insertion sort remained under 1ms and didn't increase by much. However, when scaling higher in elements the growth rate may be a problem. The next best performing algorithms in this analysis are Merge sort, Heap sort and Shell sort. These algorithms also did not increase much throughout the trials and remained under 5 ms. The next 3 algorithms are among the worst performing for This data set, and are listed in increasing efficiency: Quick Sort, Selection Sort and Bubble Sort. Quick and Selection sort exceeded 100 ms by the last trial while Bubble reached 40 ms.

Reasoning and conclusion:

In this small data set algorithms like Insertion sort performed well because even though it has a time complexity of O(n^2) , n was not large enough for the quadratic scaling to take effect. The same can be said about Quick sort which normally has complexity of O(n logn) but can be degraded to O(n^2). In a larger Data set, the best algorithms to use would be Merge,Heap, and Quick sort as they have complexities of O(n logn) and are the most efficient overall. Shell sort is the next most efficient with a typical complexity of O(n (logn)^2) in worst case ,and all other algorithms follow with O(n^2).

Citations:

bigocheatsheet.com