Merge Sort

The sorting algorithm I chose to take a closer look at is merge sort. I find the merge sort algorithm to be interesting because it is much more effective the larger the sort. On a smaller scale, merge sort performs slightly less than quick sort. Merge sort operates by splitting the group into two halves recursively until each group only contains one entry. Once this is done, the algorithm rebuilds the array by comparing each group to the groups adjacent to organize each subarray into two sorted arrays. Once finally when the two original halves are back, now sorted, the algorithm goes through each half at the same time, comparing entries and rebuilding the group in complete order.

The article analyzes the time complexity of merge sort and I found it to be interesting. It claims that in the worst-case scenario, merge sort has a time complexity of O(n)=nlog2n+n-1. This is because, in the worst-case scenario, neither of the two arrays becomes empty, before the other one contains just one element (Pradhan, 2021, p. 8). For this to happen, n must equal n-1. The article also included graphs of the time complexities of merge sort, and I found it very interesting that there is no linear change. The time required metric takes sudden spikes and declines as the number of elements increases.

In conclusion, I find merge sort to be one of the strongest and most interesting sorting algorithms because of its time complexity, and ease of implementation. Merge sort is a recursive algorithm that relies on itself to constantly split an array into halves a not predetermined number of times.

References

Pradhan, A. (2021, September 9). *Everything you need to know about merge sort*. Crio Blog. <https://www.crio.do/blog/merge-sort-algorithm/>