Sorting Algorithms

Tyler Seppala

May 15, 2019

For this assignment, I tested the algorithms BubbleSort, InsertionSort, MergeSort, and QuickSort for efficiency. I found that their efficiency is in the order which they are listed. With a small set of data to sort, they all worked at acceptable speeds, all under a second. But as I increased the dataset from 100 to 1,000 all the way to 1,000,000 items, QuickSort and MergeSort quickly showed their dominance. BubbleSort was by far the slowest, taking several minutes on the larger datasets with InsertionSort only working slightly more quickly. QuickSort always finished a fraction of a second before MergeSort, but both sorted 1,000,000 items in less than a second. While I knew going into the assignment that these two were the fastest, I was honestly amazed that any algorithm could sort that many items that quickly.

Upon observing the resource monitor, I found that none of these algorithms really had any observable effect on my computer's memory. Both BubbleSort and InsertionSort caused a large CPU spike for the time they were running. MergeSort and QuickSort both caused a small CPU spike. MergeSort's spikes were often a bit smaller than QuickSorts', and upon further research I found that MergeSort does less comparisons on average than QuickSort, it just isn't as efficient in most cases.

My conclusions from this assignment are that BubbleSort or InsertionSort would work well on a fairly small collection of data. When it comes to larger datasets, you would use QuickSort for speed, or MergeSort to conserve CPU resources.