Brandon Watkins

Isaac Griffith

CS 2235

9/27/2019

PP01 – Comparing Sorting Algorithms

To begin, it's worth mentioning that I strayed from the pseudocode given, for the merge sort. I implemented a standard merge sort, creating new subarrays at each recursive call. I suspect merge sort described in the assignment would be slightly faster, and use significantly less memory. I also took the left, mid, and right index values, and used the median value as the pivot, for my quick/hybrid sorts, instead of simply using the left index value. I suspect this helped the hybrid/quick sort times.

Insertion sort, selection sort, and hybrid sort are all in $O(n^2)$, quick sort is technically also in $O(n^2)$, but realistically should be somewhere between $O(n \log(n))$ and $O(n^2)$, while merge sort and Tim sort are both in $O(n \log(n))$. I was surprised to see that quick sort gave merge, tim, and hybrid sorts a run for their money. I was expecting to see quick sort somewhere around double the $O(n \log(n))$ sorts. I was even more surprised to see how well the hybrid sort did. I thought the hybrid would perform significantly slower than the quick sort. Combining 2 $O(n^2)$ algorithms to be faster than several $O(n \log(n))$ algorithms still boggles my mind.

I learned three primary things from this project. One, The difference between an $O(n \log(n))$ and $O(n^2)$ algorithm adds up very quickly. Two, sometimes the Big Oh value isn't everything. Clearly some algorithms that should be slower, excel with certain data types (quick/hybrid). Three, as I found out the hard way, it doesn't take much to create massive memory sinks, and creating recursive or looping algorithms exasperates this.

As far as the original question goes, I'd recommend using the Tim Sort for sorting large arrays, because, while Quick Sort and Hybrid Sort outperformed Tim Sort, I feel like the Tim Sort will perform better across a broader spectrum of data types. If my boss had specific data types in mind, like integers, I might have to consider Hybrid Sort further, and test it further, but either way Tim Sort is comparable, and above all, predictable.

My results are posted below, I broke it into two plots to get a better view of the more competitive sorting algorithms.