Saxon McIntosh: CSI Temple Analysis

*****DISCLAIMER*****

Spencer actually set up the script to output to a CSV file before he even sent it to me. I feel bad for not having done the same for him (although we weren't actually supposed to based on the project description), but it would also be seriously difficult for me to separate the sorting algorithms themselves from his original output schema without looking at them, so instead I've used it for my analysis.

*****END DISCLAIMER*****

Sort 1 I believe to be Insertion Sort. The number of comparisons and exchanges are very high compared to the others. The comparison and exchange counts should be equal, although the last 'exchange' may be to replace the the indexed number with itself.

Now after this I'm in a bind. I know that one of the remaining two algorithms is intended to be Quicksort because of the project specifications. But of the two only one results in more comparisons than exchanges, and it's not even consistent in that respect. Time to look for more clues elsewhere.

At this point in the game after examining the comparisons and exchanges for Sort 3, I want to conclude that it has to be Timsort. Mergesort is the only sorting algorithm of those given that will always result in more exchanges than comparisons – partially because it cannot sort in place, and partially because elements are only compared until one of the two arrays is out of elements, at which point all of the remaining elements are appended. So Insertion Sort, Quicksort, Shellsort and Heapsort all require more comparisons than exchanges to complete. And Mergesort requires more exchanges that comparisons, but it's very consistent in that respect. That only leaves Timsort, which does have Mergesort as one of its components so it makes sense that the difference between comparisons and exchanges would be more variable. I guess that means Sort 2 must be Quicksort, but I have no idea why that would be and the time required for each sort is giving me no clues.

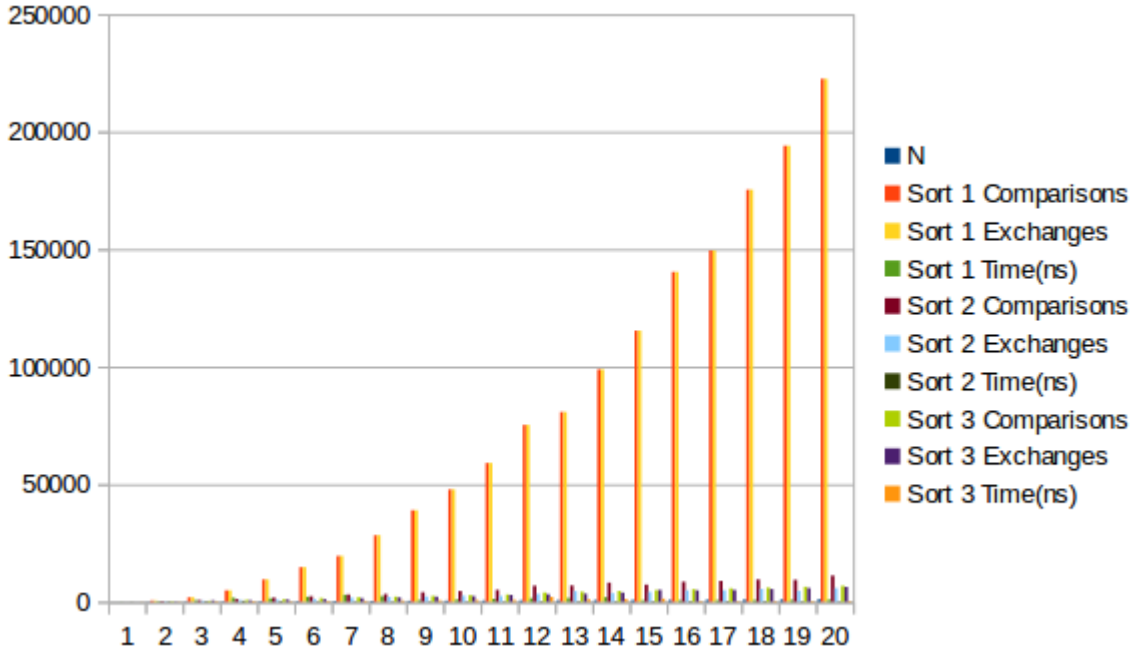
30 minutes later

Whelp, I did a bad thing. Sometimes when a case runs cold, the investigators have to drop it for lack of evidence. Then years later the killer reveals themselves after the statute of limitations runs out. That's what happened here – I scoped out the algorithms. Sort 3 is actually Shellsort, which is generally a more efficient variant of an Insertion Sort. So I'm not positive why there are fewer comparisons than exchanges at n = 100, for example. Then I looked at Sort 2, which was in fact Quicksort. Every exchange was accounted for – unfortunately this also included every time a temp variable had to have its place traded with, so the exchange count ballooned. The subsequent output after I nixed the extraneous exchanges fell more within my expectations.

It seems as though some of the problem comes from each programmer being unsure of what should be defined as an exchange. Should trading values with a temp variable be counted, or should the final result of the exchange be the only thing that matters? I had a moment of uncertainty myself when I was building out Mergesort. Should building out the smaller arrays be counted as an exchange, even though values aren't actually trading places but being assigned? After all, it does add to the time it takes the algorithm to complete. This has been a significant learning experience in sorting forensics for me.

Saxon McIntosh: CSI Temple Analysis

The following graph shows the values including Insertion Sort, which dwarfs the others beyond recognition:



Next is the same graph sans Insertion Sort: