Project 1C Report

In theory, I would expect runtime of Θ(n) for the linear search execution time, a Θ(nlog(n)) runtime for merge sort, and a Θ(log(n)) runtime for binary search. Since we are doing merge sort only once and linear search and binary search 5000 times, I would assume that merge sort + binary search would be faster for this number of searches, however, if I were to do it for a smaller number of searches such as 10, I could see linear search being faster due to having a smaller runtime complexity than merge sort.

|  |  |
| --- | --- |
| Linear Search Samples | Time (ms) |
| 5000 | 185 |
| 10000 | 385 |
| 20000 | 755 |
| 40000 | 1494 |
| 80000 | 2974 |

|  |  |
| --- | --- |
| Merge Sort Samples | Time (ms) |
| 5000 | 126 |
| 10000 | 285 |
| 20000 | 621 |
| 40000 | 1211 |
| 80000 | 2531 |

|  |  |
| --- | --- |
| Binary Search Samples | Time (ms) |
| 5000 | 8 |
| 10000 | 20 |
| 20000 | 35 |
| 40000 | 63 |
| 80000 | 127 |

|  |  |  |
| --- | --- | --- |
| Samples | Linear Search (ms) | Merge Sort + Binary Search (ms) |
| 5000 | 185 | 134 |
| 10000 | 385 | 305 |
| 20000 | 755 | 656 |
| 40000 | 1494 | 1274 |
| 80000 | 2974 | 2658 |

As we can see, merge sort + binary search’s execution time is consistently quicker than Linear Search’s execution time. This all lines up with the theoretical performance that I expected, as having a major part of the execution time was the 5000 runs of Θ(n) time complexity vs 5000 runs of Θ(log(n)) time complexity. This is where the binary search + merge sort algorithm gains time on the linear search algorithm, as without this many runs, only doing somewhere like 5/10 runs would make linear search faster due to not having to run merge sort, an Θ(nlog(n)) time complexity algorithm.

I did not witness any anomalies during my experiment.

One thing that surprised me was that the efficiency of linear search was as close to the efficiency of the merge sort + binary search, as I expected this gap to be much larger. If we were doing more searches, this gap would widen for sure, as the pro of using binary search is the search speed being much faster than linear search, with the big con being that the merge sort takes a while at the start. Another reason this difference is not greater is the size of the object that is getting sorted in merge sort is much bigger than just an integer, as it is a whole book with five different pieces of data that are stored and need to be transferred to the new location when doing a swap.

While I was doing this experiment also, I tried a recursive binary search and it was much slower than the iterative version of the algorithm, so I switched the binary search algorithm to an iterative algorithm. I believe this is because of how many function calls were needed in the recursive version, so in other environments other than the one I ran it in, Visual Studio, this could lead to a different result for recursive vs. iterative binary search.