Assignment 5: Quicksort and Mergesort

Analysis Document by Jacob Brown

My partner was Anthony Iovino, I (Jacob Brown) am submitting the source code for this assignment.

I really liked my partner. He kept a schedule with me, and he was really helpful when I didn't know how to implement/understand something. if I get the chance to work with him again I will.

Below are the are two graphs for the run times of our implementation of Mergesort. One graphs a Linear Threshold and the other graphs a Fractional Threshold.

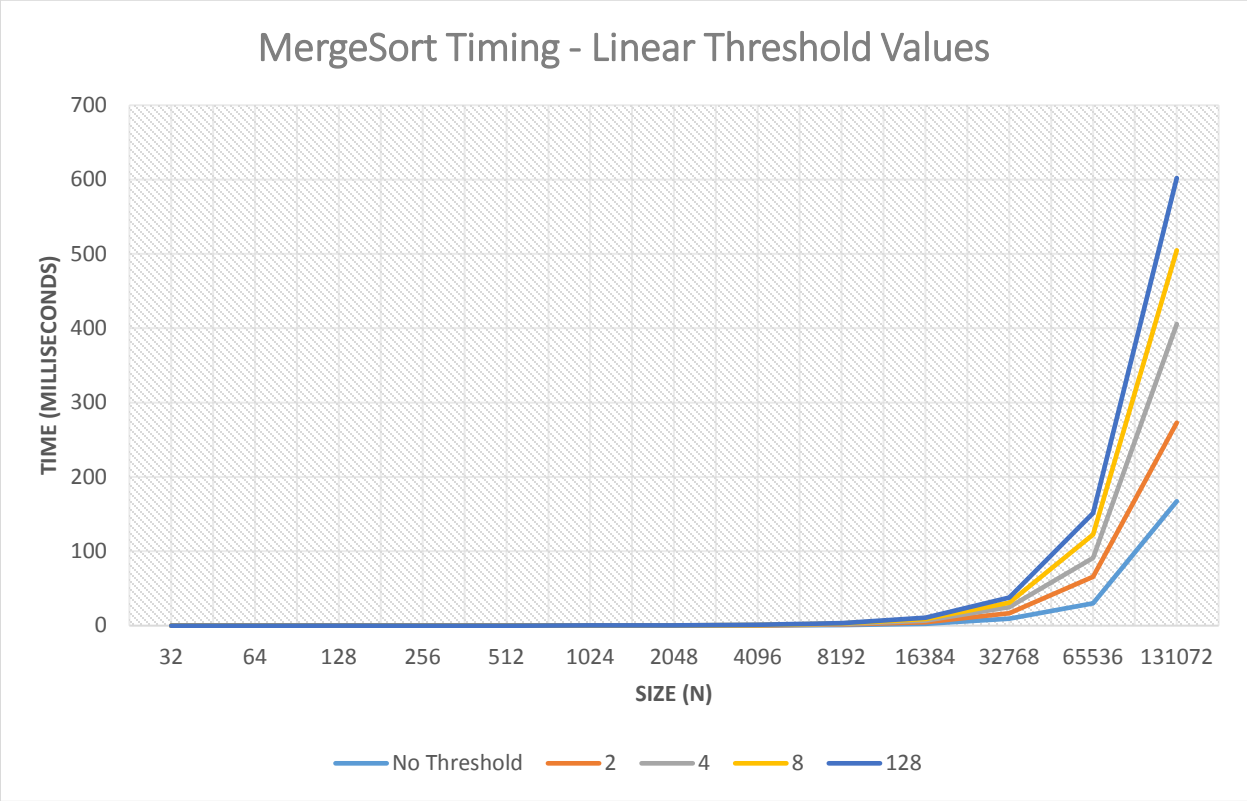


Figure 1. Mergesort with Linear Threshold Values

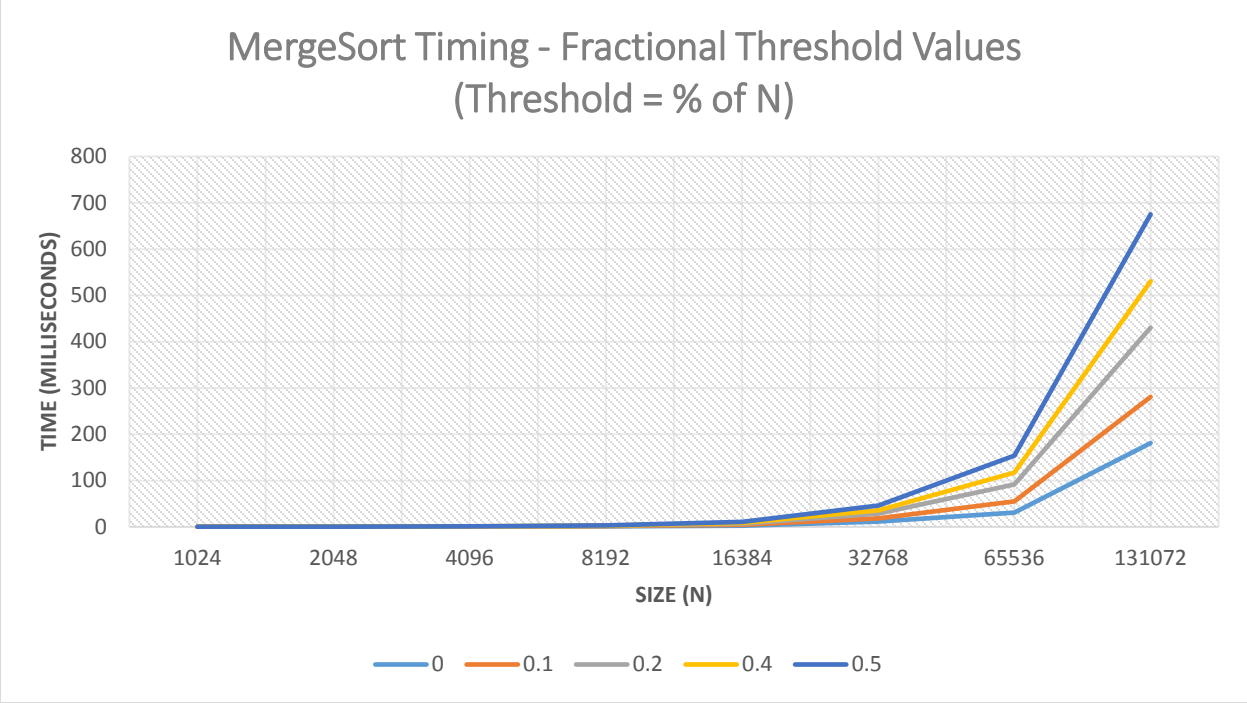
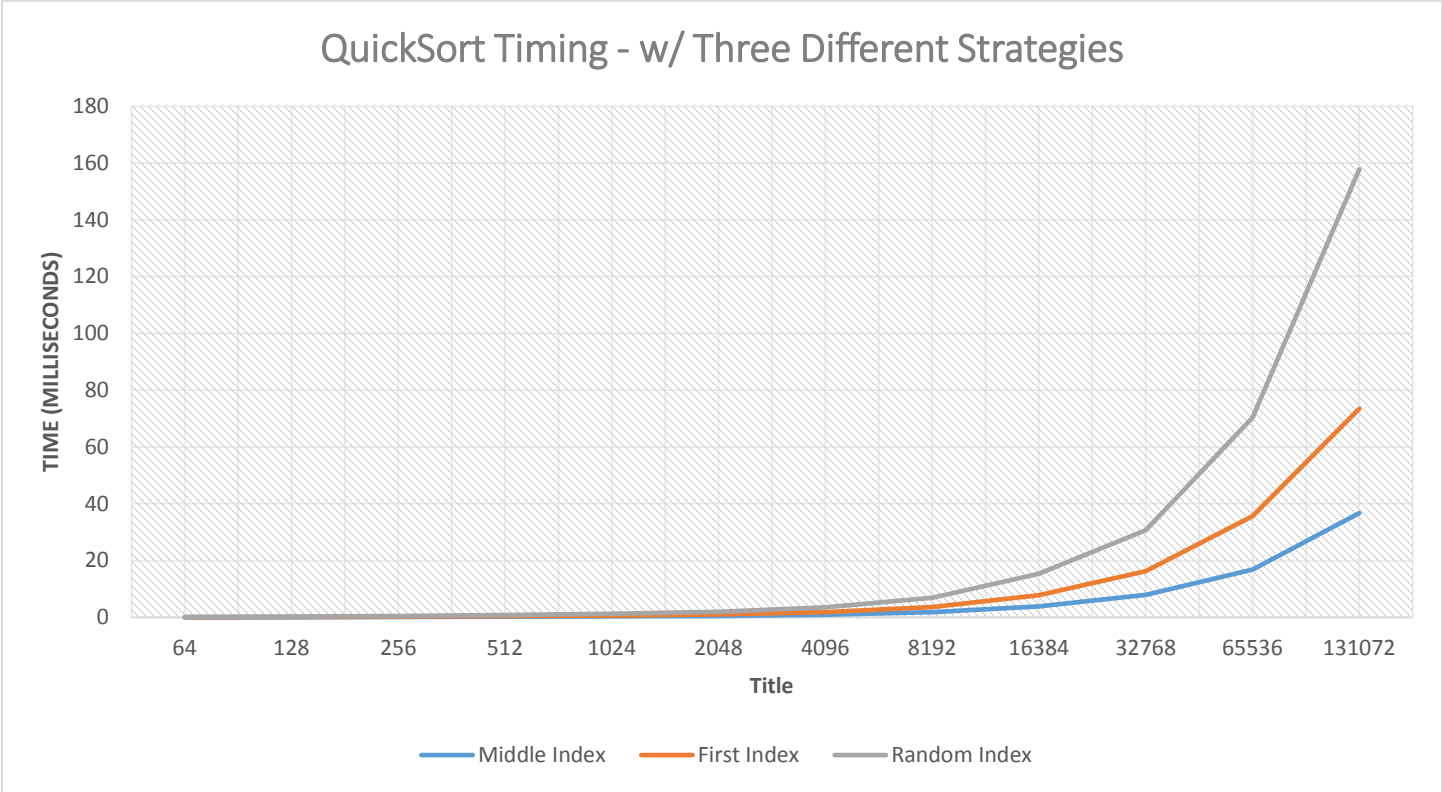


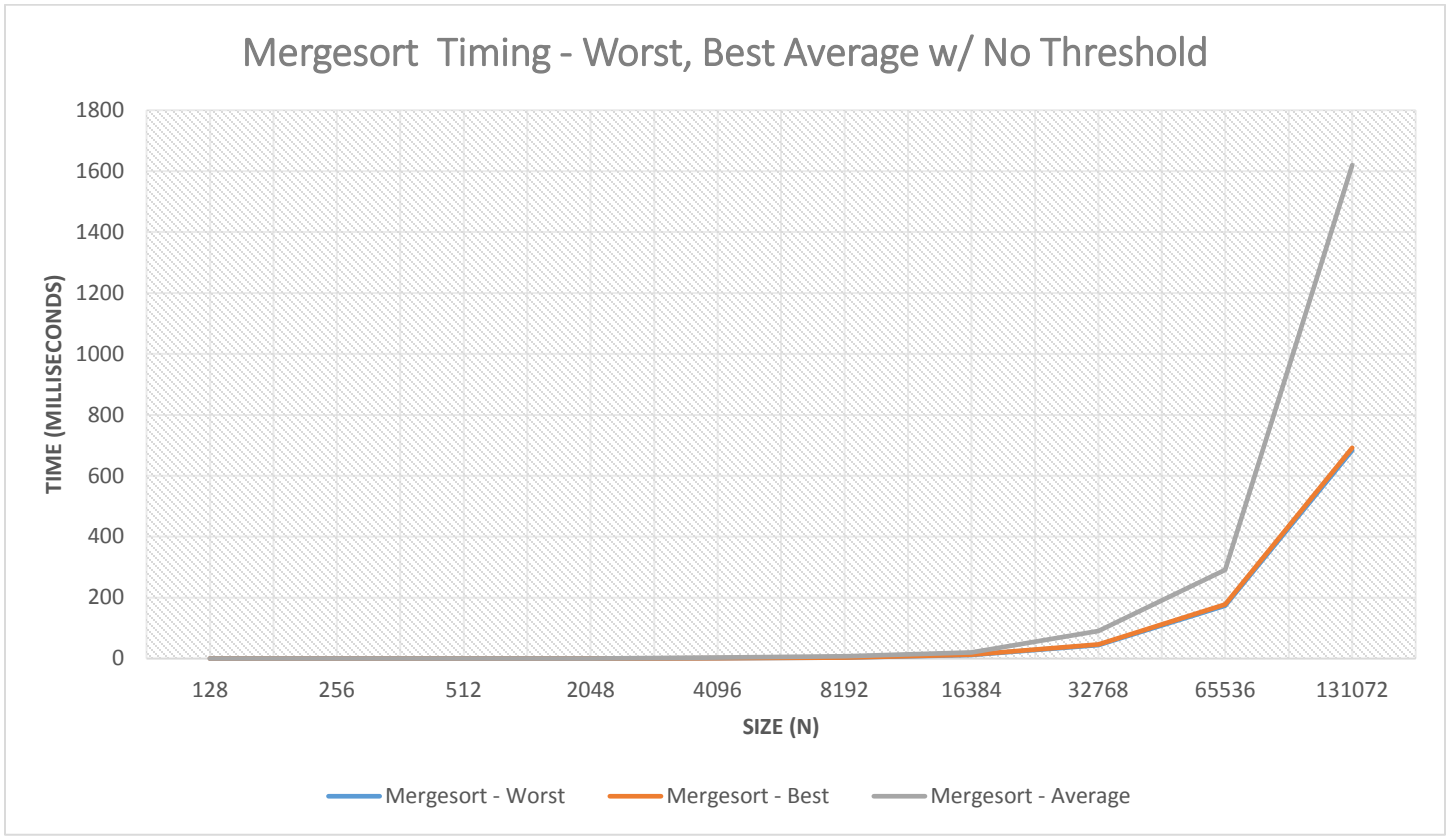
Figure 2. Mergesort with Fractional Threshold Values ("0" Meaning No Threshold)

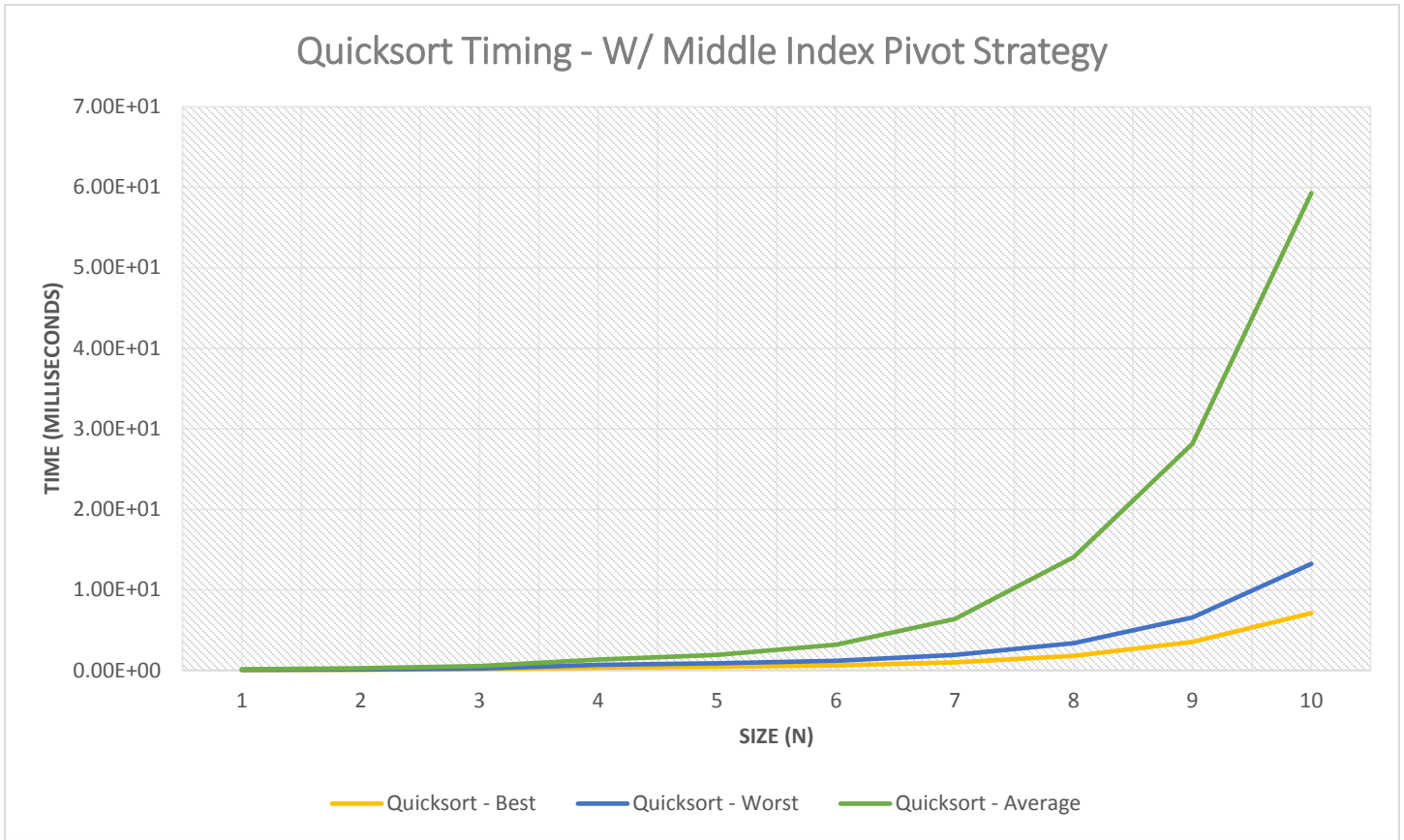
Below is the run time graph for our implementation of Quicksort.



Each line is a different Pivot Strategy: Middle Index is the middle element in array; First Index is the first element in the array; Random index is a random element in the array.

Below is the run time graphs for the Best, Worst, and Average sets of both Mergesort and Quicksort using the strategies we found most efficient.





The graphs show something very unexpected in this assignment, there could be some portion (bug) of both our sorting algorithms which warp the algorithmic complexity. I'm kind of perplexed at the fact that our average times for both algorithms exhibit a longer runtime (n^2) than both the worst and best cases. I'm also surprised that the thresholds for MergeSort did not produce better outputs than with no threshold since the best case for insertion sort is o (n). The best case sets for Quicksort and Mergesort make sense since it exhibits n log n. Also the worst case set for Mergesort was the same as the best case set, so this also makes sense.

This assignment took around 15~ hours to complete, and it was brutal.