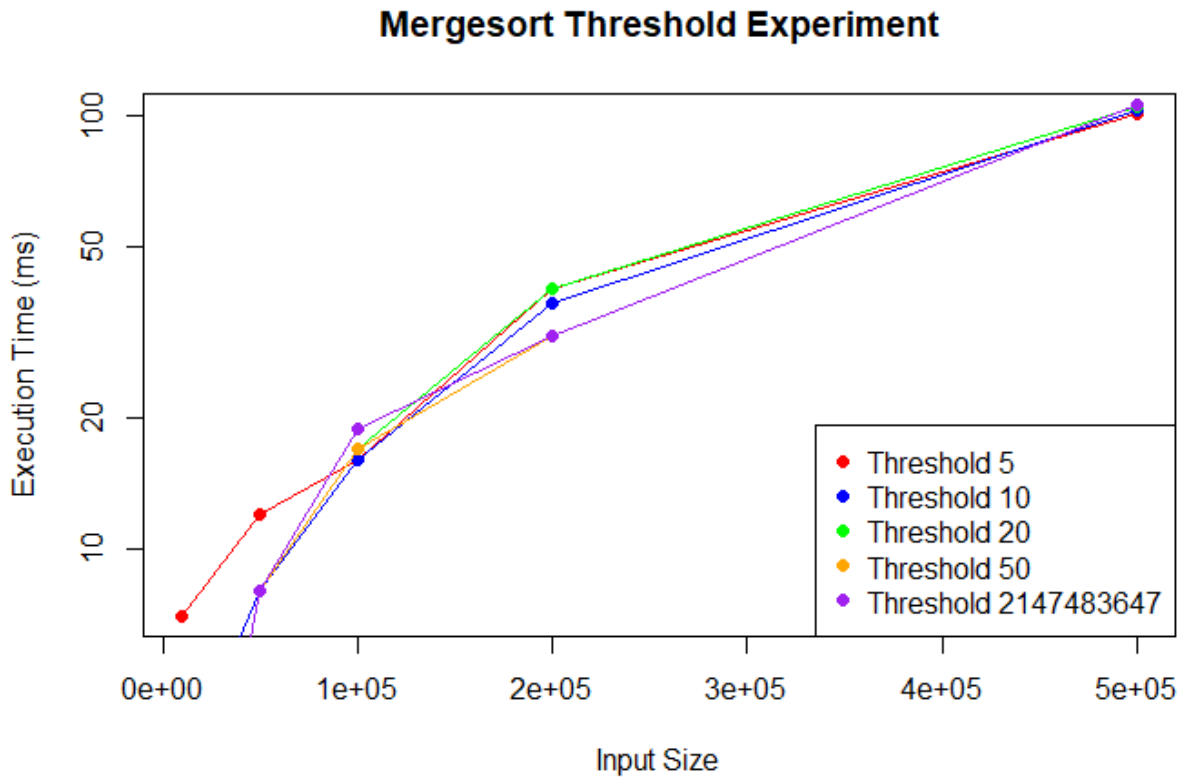
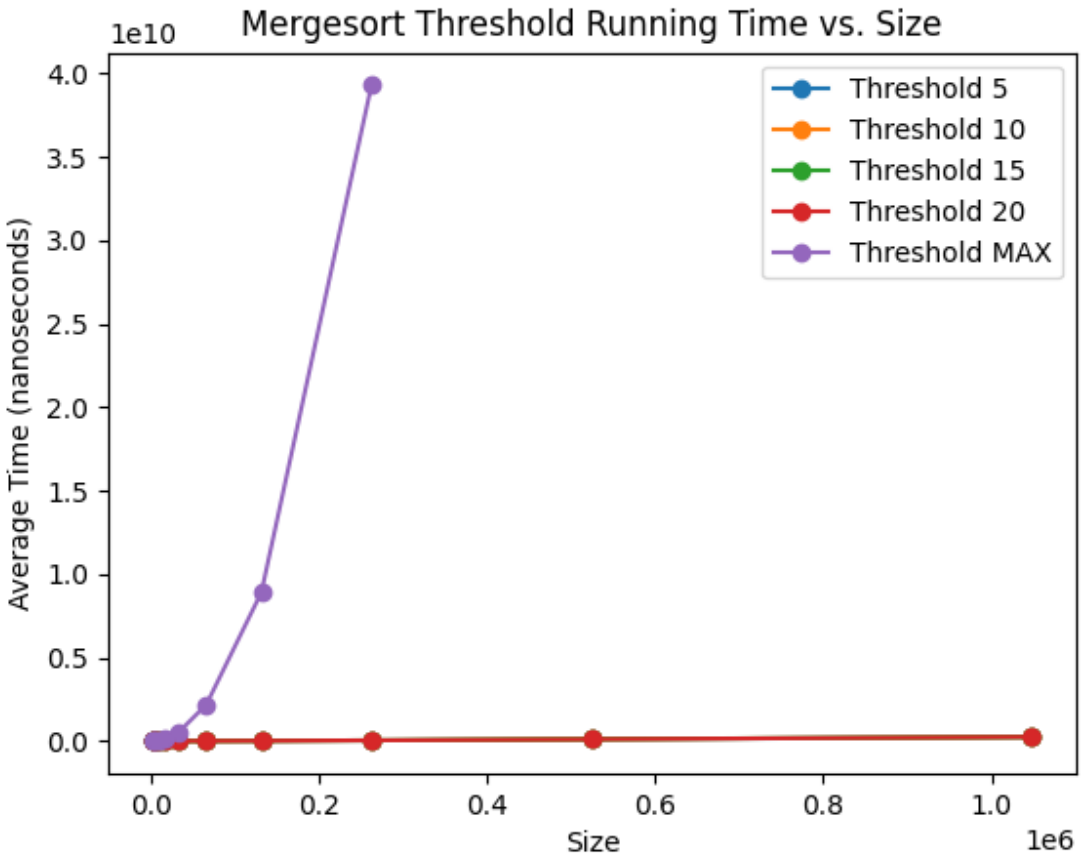
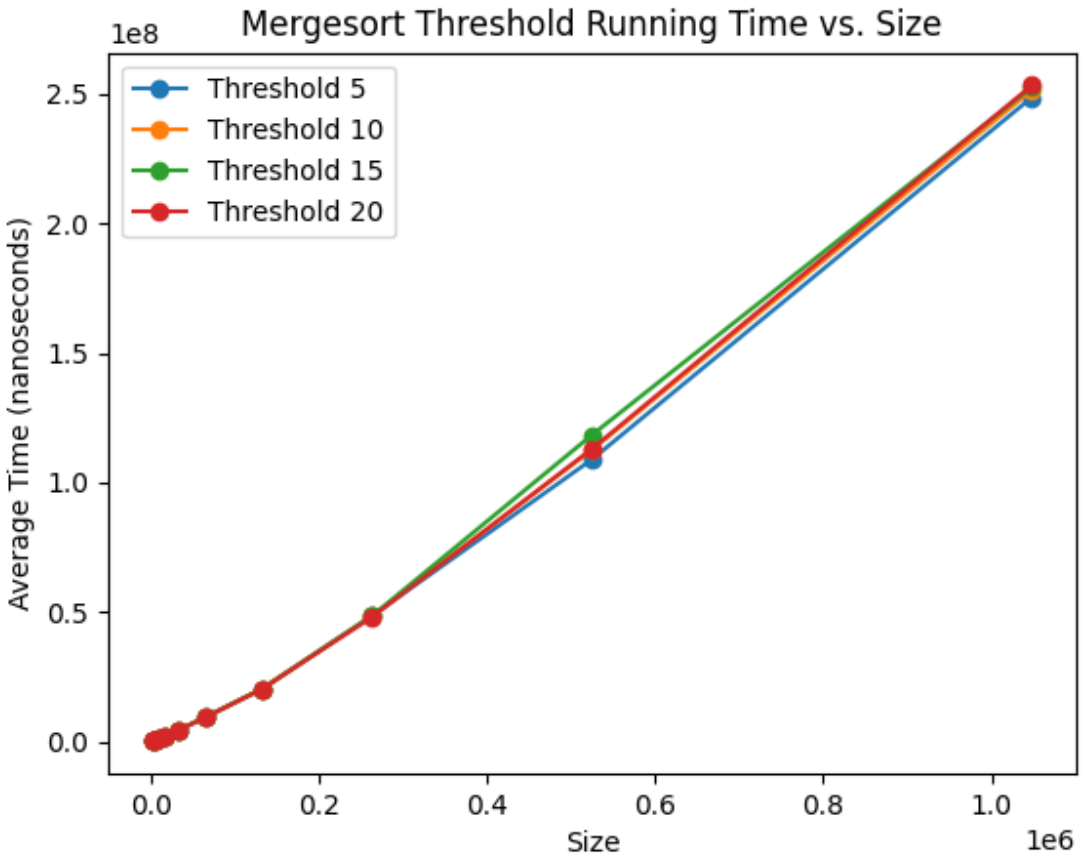
Who are your team members? Brian Erichsen Fagundes & Mina Akbari

Mergesort Threshold Experiment

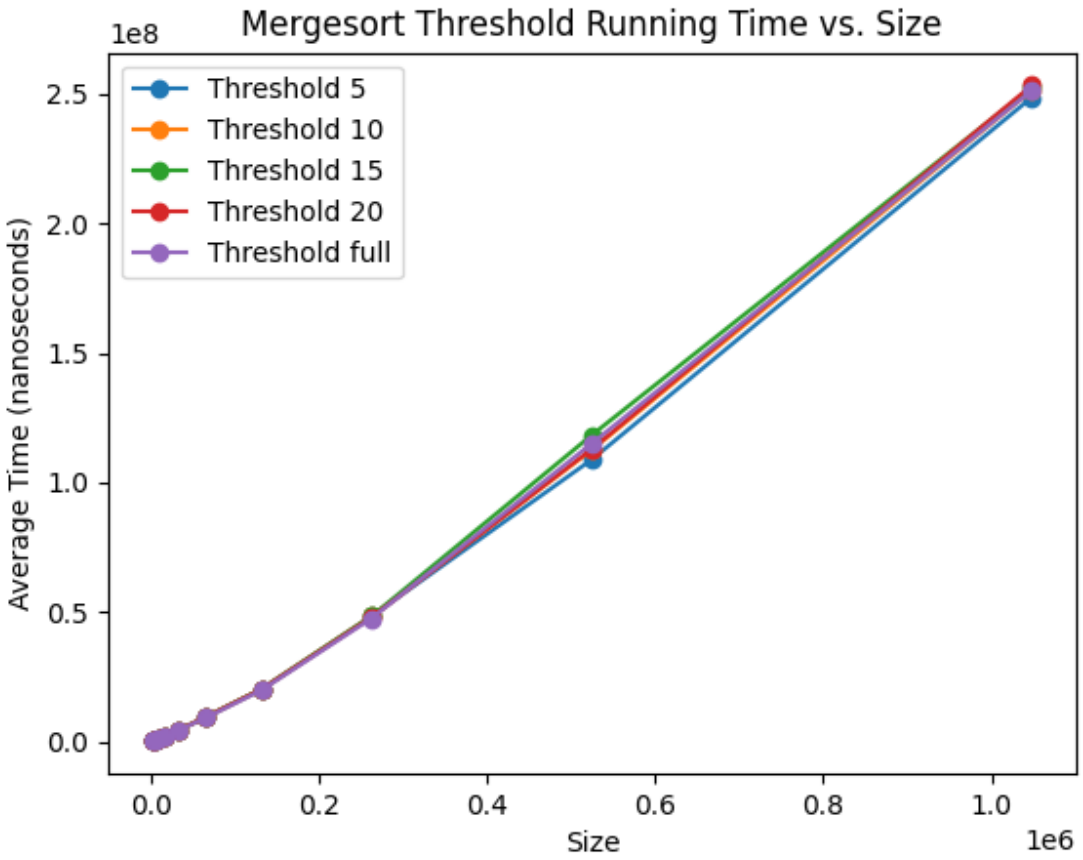




Here the threshold is set to Integer.MAX_VALUE; so it pretty much does insertion sort all the way through; which shows how much slower it is compared to mergesort method.

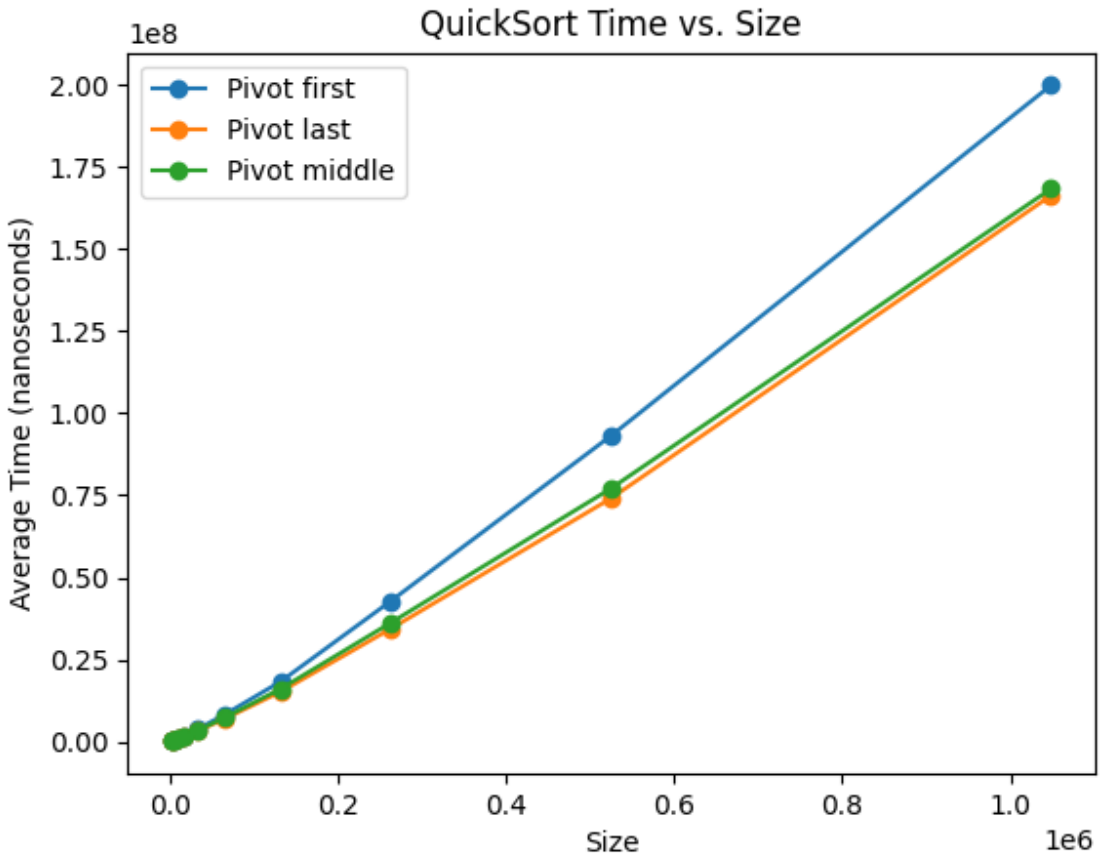


Here the plot shows the lines for 4 different thresholds where the result is not much different than each other but the blue line representing a threshold of 5 is slightly more efficient than the other thresholds.



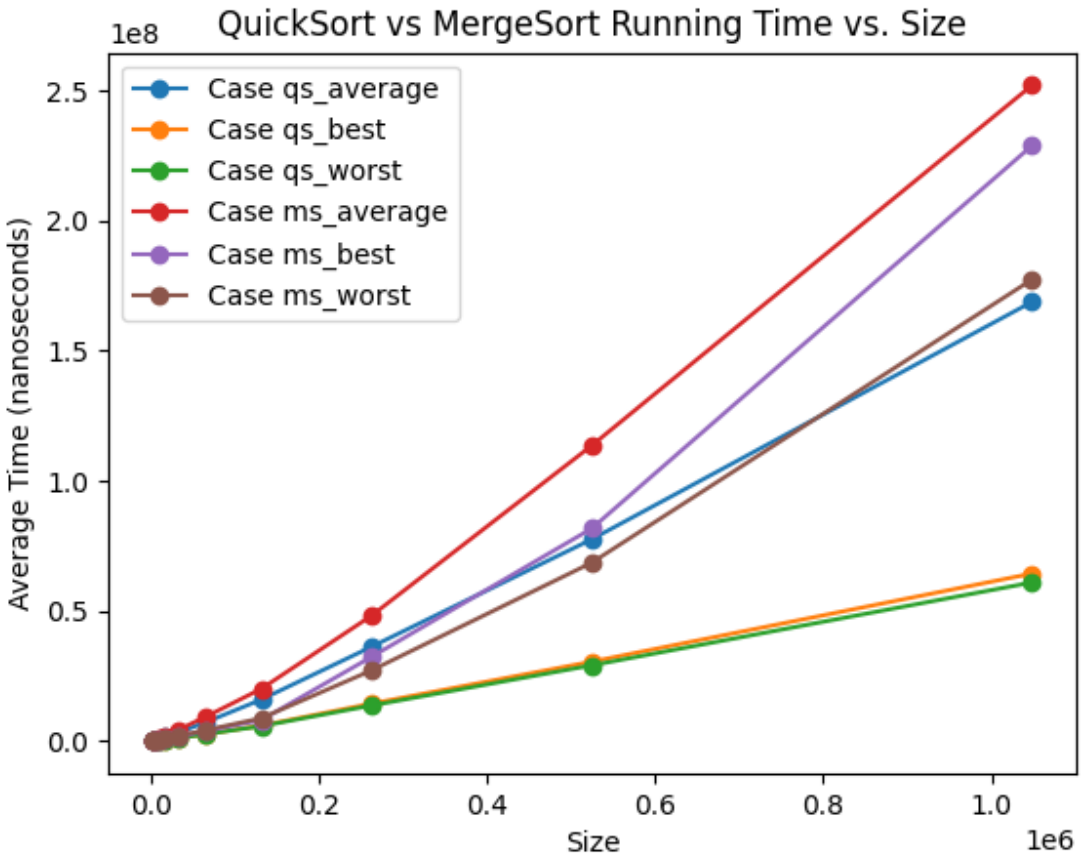
Here the purple line shows using the Mergesort method without switching to insertion sort which points that using the threshold is slight better than not using it.

Quicksort Pivot Experiment



In this experiment, the best strategy for pivot is using the last element as the pivot point at least from the data collected. Using the last element versus the one in the middle or the first element showed a slightly better run time compared to using the middle or the first instead. Using the middle as a pivot does show better performance for smaller data sizes and is often treated as the best pivot when compared to last and first indexes.

Mergesort vs. Quicksort Experiment



The best algorithms for best, worst or average cases depend on various factors but as far as **average cases** the quicksort it did outperformed all cases of merge sort in the collected data. This usually happens whenever the data is randomly or uniformly distributed however mergesort has a consistent O(n log n) time complexity for all cases.

As far as **worst cases**, quicksort can have worst time complexity of O(n^2) when poor pivot choices lead to unbalanced partitions when mergesort has the guaranteed O (log n).

And finally as far as **best cases** quicksort did outperform mergesort which both exhibit a O (n log n) pattern however the mergesort has it's recursive nature and additional memory requirements that can make it a bit less efficient in terms of space complexity and cache complexity compared to quicksort.

Do the actual running times of your sorting methods exhibit the growth rates you expected to see?

Mostly yes. However in the collected data the quicksort method for worst case exhibited better performance than all the other data when quicksort worst case should exhibit run complexity of O(n^2) instead. Furthermore with the quicksort method, the pivot selection when comparing the first index, the middle one or the last, one; the last index exhibited surprisingly good performance almost identical to using the middle index to pivot. However quicksort has the downside of being not as stable sorting algorithm when compared to other sorting algorithms due to its pivot choices that can possibly lead to unbalanced and possibly unpredictable partitions; meanwhile the mergesort is a very stable algorithm that preserves the relative order of equal elements.

==========