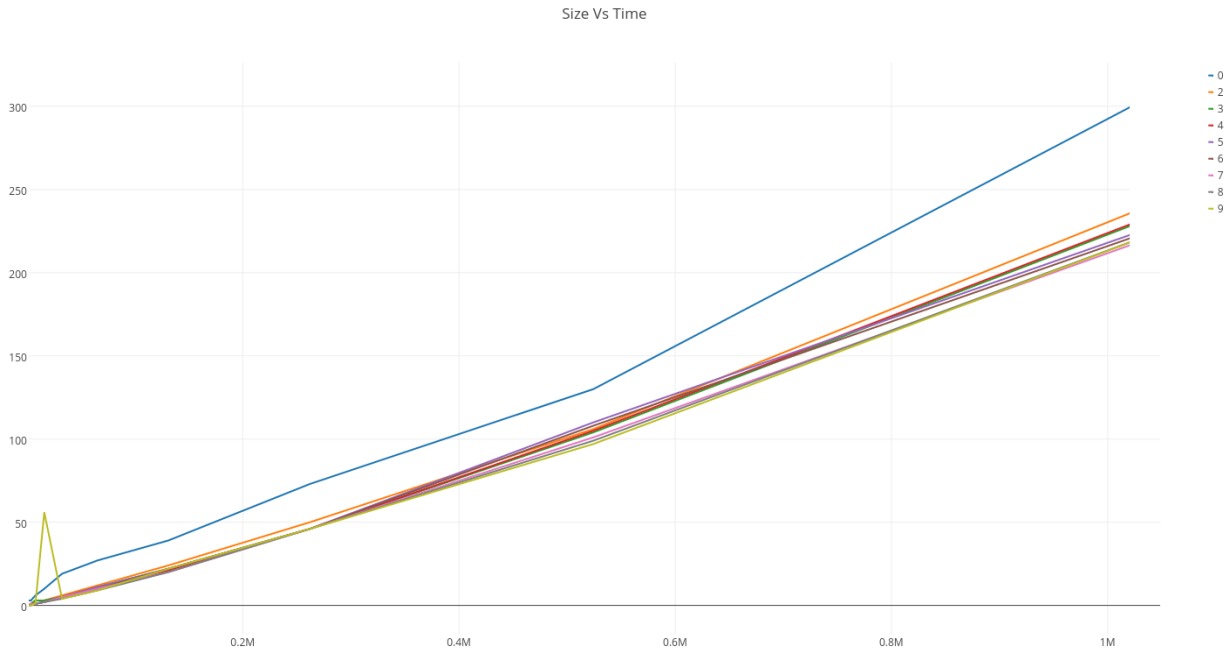
Abdulaziz Aljanahi
U0901606

1. My programming partner is Casey Yip (Ho Lam Yip), I will be submitting the code.
2. On this assignment, my partner was barely did anything, I was forced to do most of the coding, which is expected in since we are a team someone might do more work than others. However, on the last day of the assignment we had a few errors in our code that I was trying to fix. While I was doing that, he said that his battery died and just left. Without doing anything more. Which is why we had to submit the assignment late. Since I had to fix all the coding, and had to do all the testing, and timing.
3. I honestly don't like working in pairs that much. The best part, is probably being forced to meet up early and work on the assignment. However, I find coding in partners very disconnected. We always end up working on separate things, or sometimes just writing the same part of code separately then combine our work. I guess for me to be a better partner I should start writing the code with my partner instead of each of us working alone on different parts of the program.

All the lists used in the timing process are lists ranging from size 2^{11} To 2^{20}. and all generated lists were copied as instructed.

Abdulaziz Aljanahi
U0901606

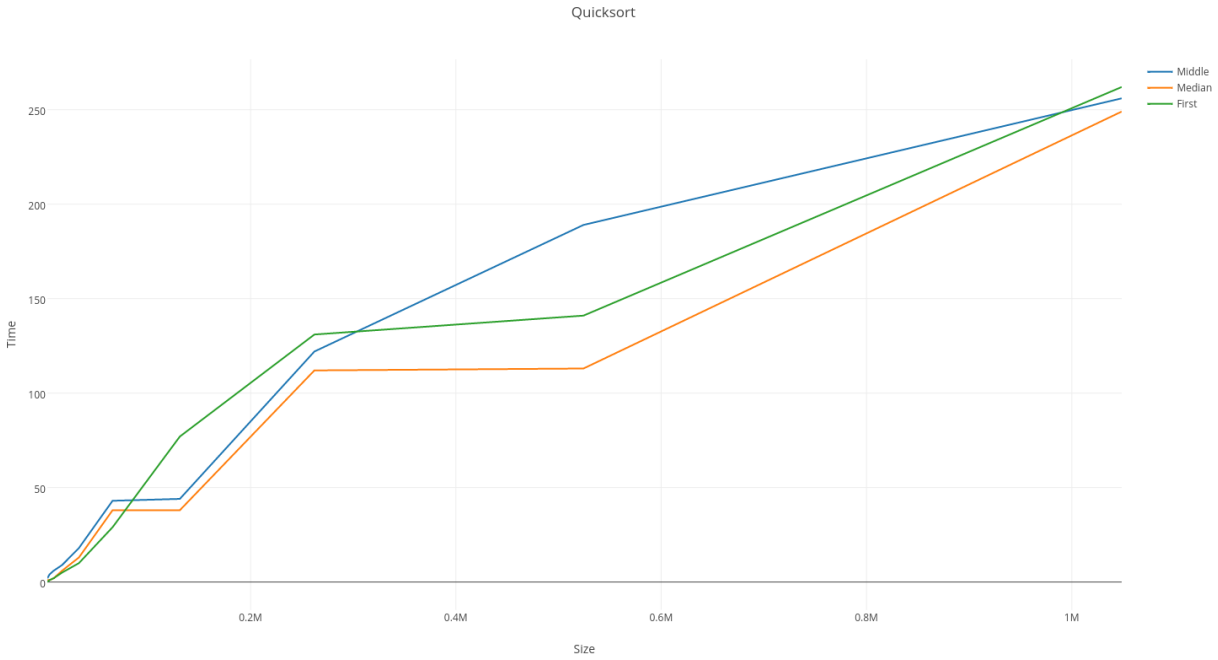
4. From the tests I've found that the most optimal threshold is at around 7.5



This is a plot of Time Vs List Size. All different thresholds were very close to each other. And there is very minimal differences between thresholds. However, the best is still at 7.5 . The blue line is shows only using mergesort. And it is clearly the least efficient.

Abdulaziz Aljanahi
U0901606

5. While testing Quicksort with different pivot points. I found that the best is overall is just picking the median pivot point.

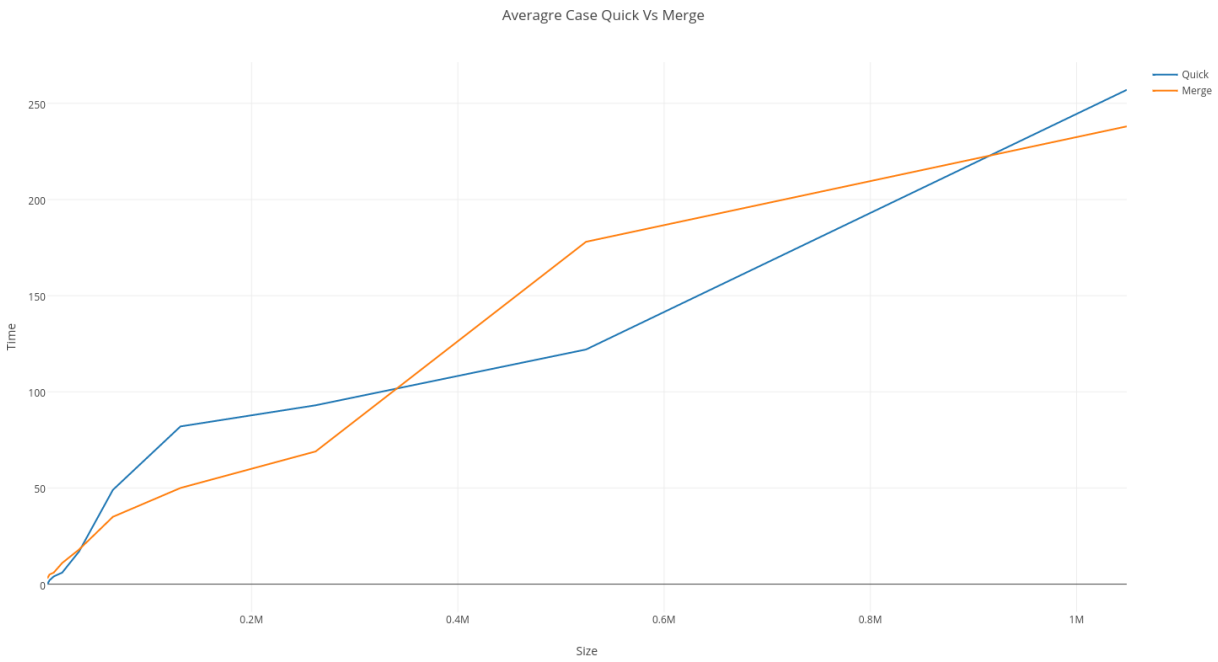


And the worst case is surprisingly sometimes faster, than picking the middle pivot

Abdulaziz Aljanahi
U0901606

6. These Graphs represent the Quicksort vs Mergesort in the three different cases

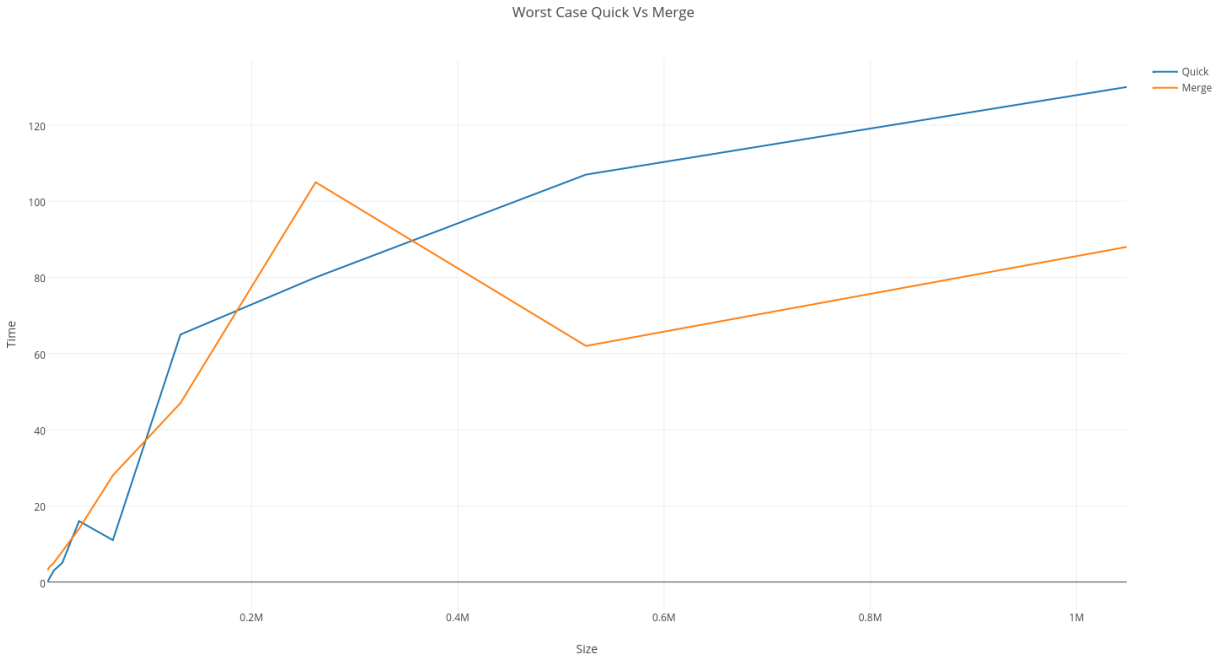
Average Case



In the average case, between .3 & .9 million the quicksort seems faster, however in smaller cases mergesort is somewhat better. At the end it seems like mergesort is faster, it might be that the switch depending on the size of the list.

Abdulaziz Aljanahi
U0901606

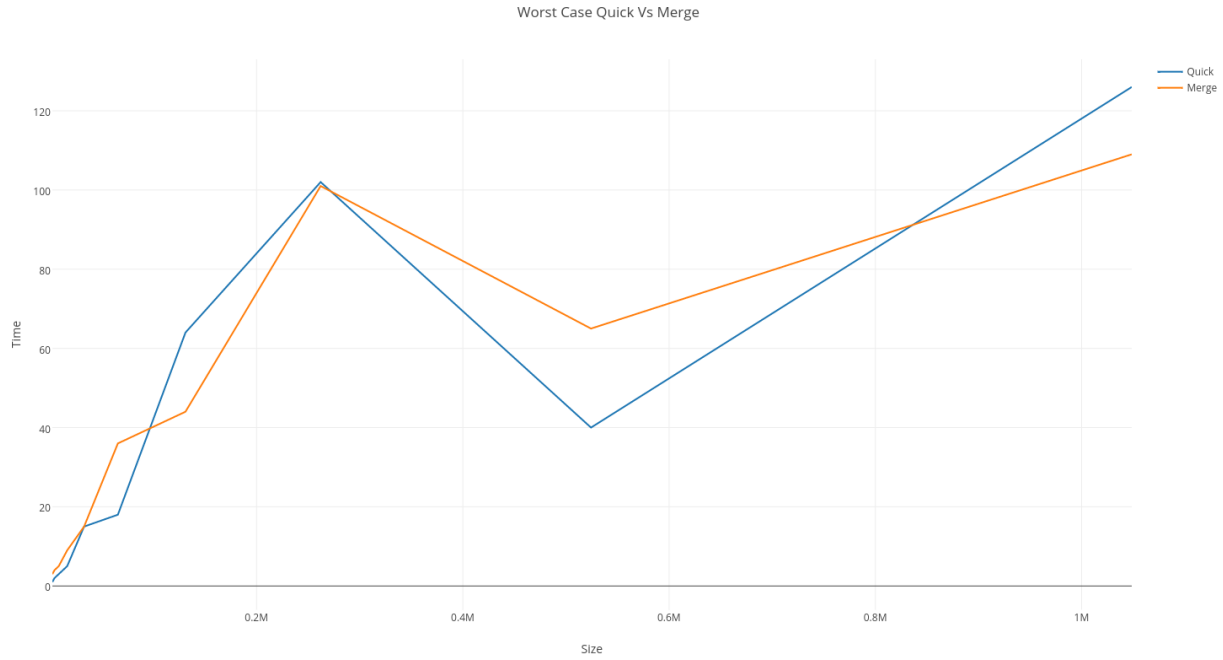
Best Case



At the best case merge sort is clearly better.

Abdulaziz Aljanahi
U0901606

Worst Case



And finally at the worst case, quick sort seems to be slightly better, however it seems that the interchange depending on the list size. Similar to the average case.

7. Yes they do generally do. Mergesort's graph is almost identical to Θ(n log(n)) graph which is what it is expected to be. Quicksort also, even though it jumps around more the general behavior of the graph is close to a O(n log(n)). However when starting to change, pivot points, and adding insertion sort, it seems that the graphs start behaving differently. A little, however there general behavior is still close to a O(n log(n)).

   Quick sort at it's worst should behave as a O(n^2), however since it also switches to insertionsort at a certain threshold, that might be the reason why it's not behaving as a O(n^2) even in the worst case.

8. I've spent around 15 hours on this assignment.