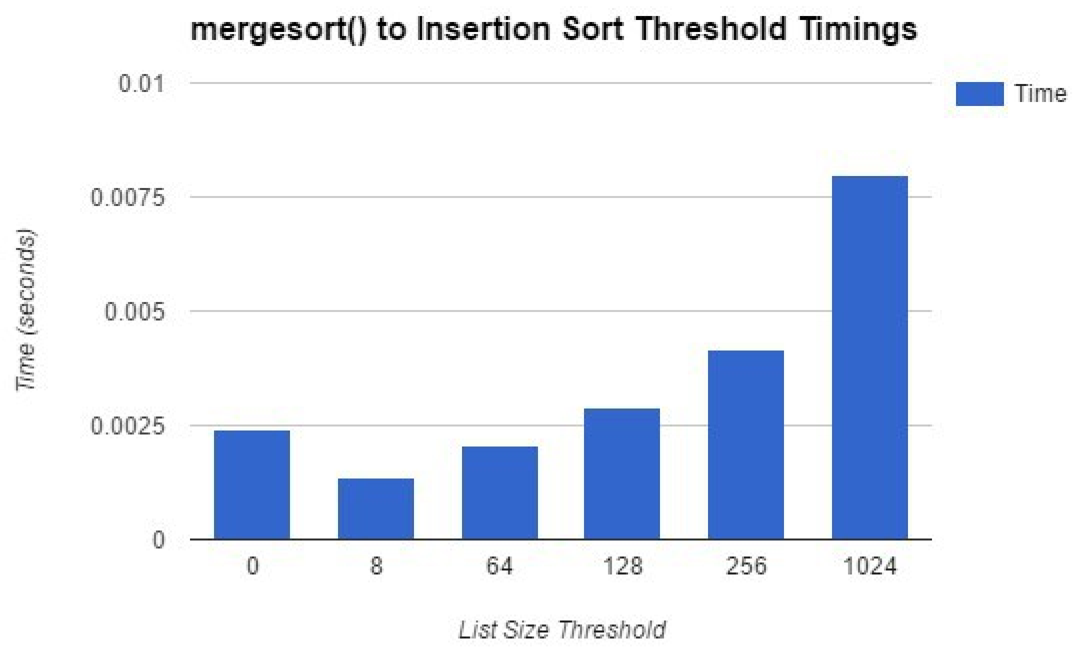
Josh Shipley
CS2420
Assignment 05 Analysis

1. Who is your programming partner? Which of you submitted the source code of your program?
- My partner is Daniel Garcia and he will be submitting the code.

2. Evaluate your programming partner. Do you plan to work with this person again?
- Despite some scheduling issues, he is a good partner to work with. He helps with the concepts and code that I have trouble understanding.

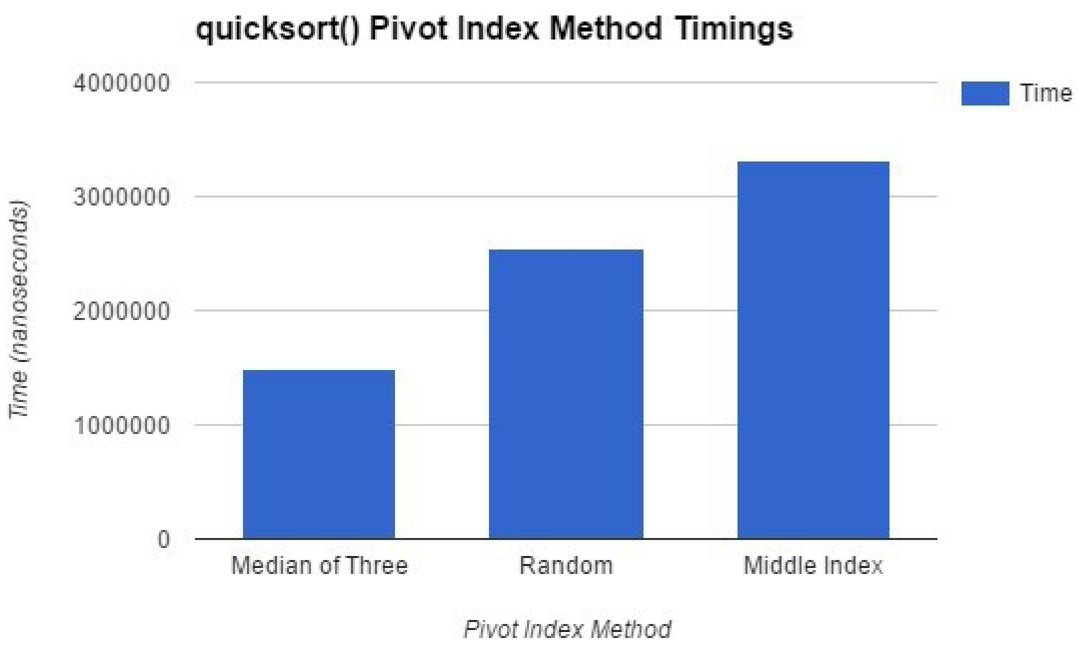
3. Evaluate the pros and cons of the the pair programming you've done so far. What did you like, what didn't work out so well? You'll be asked to pair on three more of the remaining seven assignments. How can you be a better partner for those assignments?
- Scheduling can be a bit tough sometimes, but when we do get together the assignment goes by a lot quicker when we have two heads working on the one assignment. I can have some trouble explaining my thoughts as we go though, so I do need to work on that some.

4. Mergesort Threshold Experiment: Determine the best threshold value for which mergesort switches over to insertion sort. Plot the running times of your threshold mergesort for five different threshold values on permuted-order lists (one line for each threshold value). In the five different threshold values, be sure to include the threshold value that simulates a full mergesort, i.e., never switching to insertion sort (and identify that line as such in your plot).



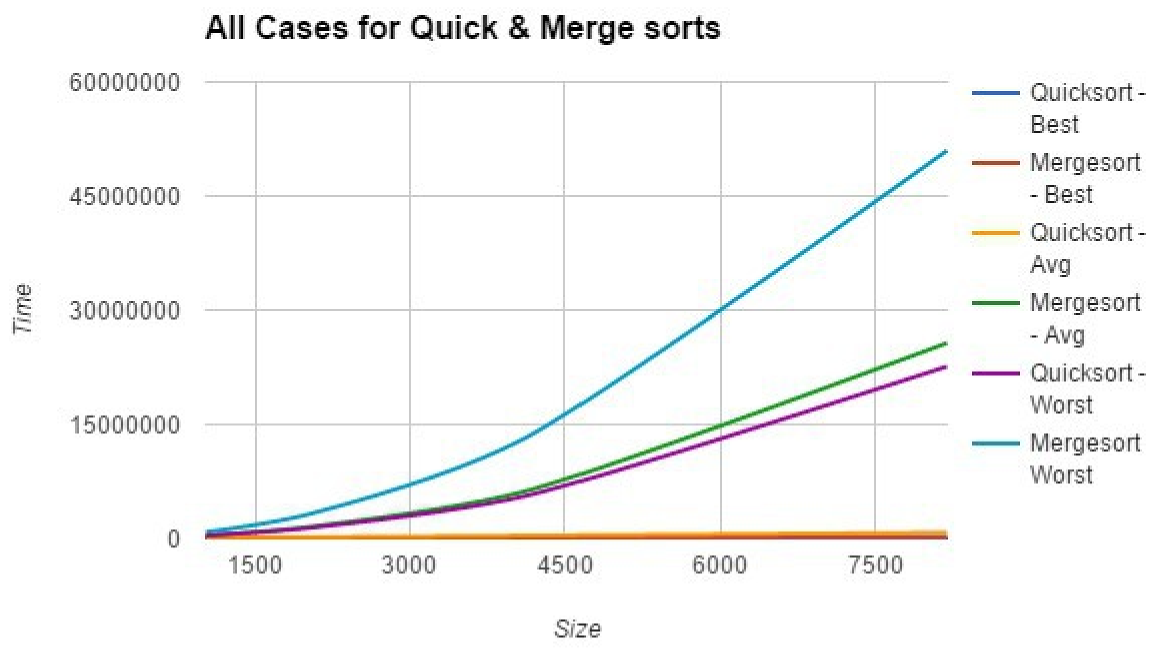
- What we did was for a large array size, we ran mergesort with the threshold of when it switches to insertion sort steadily increasing, with 0 indicating a full mergesort and doubling the threshold from there. As shown by the graph above, we found that 8 was the best threshold to switch between merge and insertion sort. This makes sense since although insertion is typically N^2, the array size is so relatively small that it's closer to the best case scenario of N timing, whereas a full mergesort is constantly Nlog(N) all the way down.

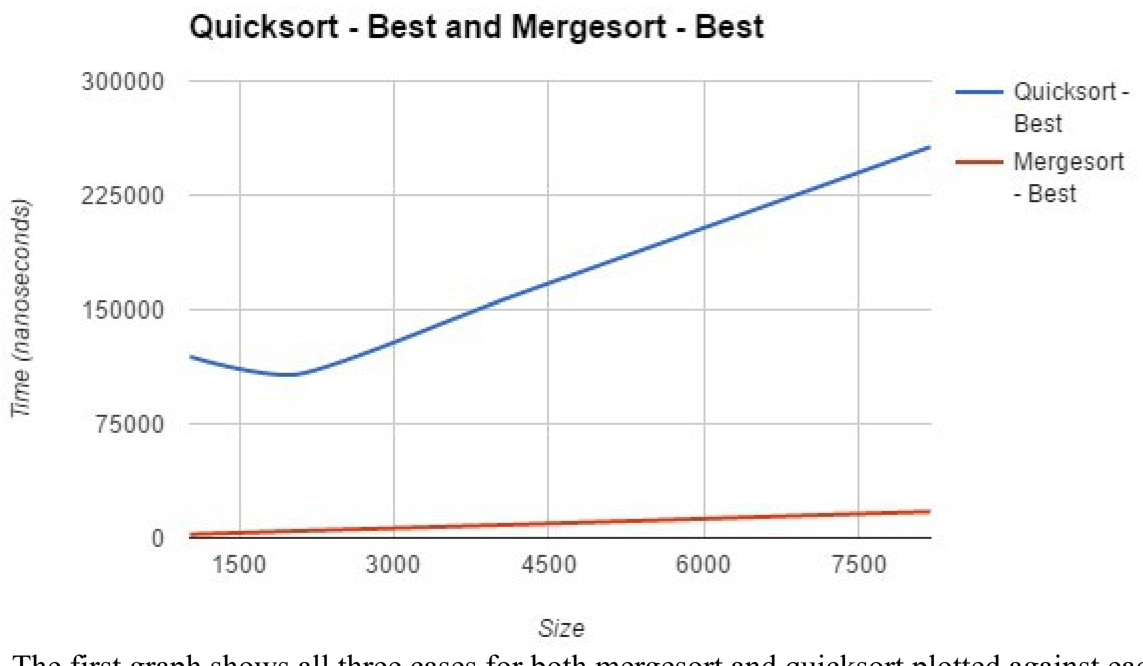
5. Quicksort Pivot Experiment: Determine the best pivot-choosing strategy for quicksort. Plot the running times of your quicksort for three different pivot-choosing strategies on permuted-order lists (one line for each strategy).



- This was run similarly to the previous experiment, however, this time after each set of tests, we simply set it to automatically switch to the next pivot-choosing method. As you can see, the best method we found was that of finding the median value of three from within the array. This makes sense as this is the only option we had that guaranteed that as long as the list size was larger than two, it would not select the overall maximum or minimum value, which would lead to a worst-case scenario.

6. Mergesort vs. Quicksort Experiment: Determine the best sorting algorithm for each of the three categories of lists (best-, average-, and worst-case). Plot the running times of your sorts for the three categories of lists. You may plot all six lines at once or create three plots (one for each category of lists).





- The first graph shows all three cases for both mergesort and quicksort plotted against each other. The second graph rescales down to show only the best cases for each as these numbers were so relatively low and hard to see when viewed against worst-case scenarios.

7. Do the actual running times of your sorting methods exhibit the growth rates you expected to see? Why or why not? Please be thorough in this explanation.

- Yes. Starting with the best cases, it makes sense that mergesort is notably faster since it doesn't have to do as many comparisons, calculations, and swaps as quicksort does when the list is already ordered. Next, the average cases. It was a little shocking to see how much faster the quicksort runs on average than mergesort, but I suppose it makes sense that, with the pivot strategy, it can get easier to stay closer to the best case scenario as long as you don't pick an extreme value for the pivot. Finally, we have the worst-case scenarios. These are rather self explanatory, as mergesort runs in a constantly Nlog(N) time while quicksort has to go up to N^2 time since it has to swap each and every element each time, no matter what the pivot chosen is.

8. How many hours did you spend on this assignment?
   - We spent around 13 – 14 hours on this assignment.