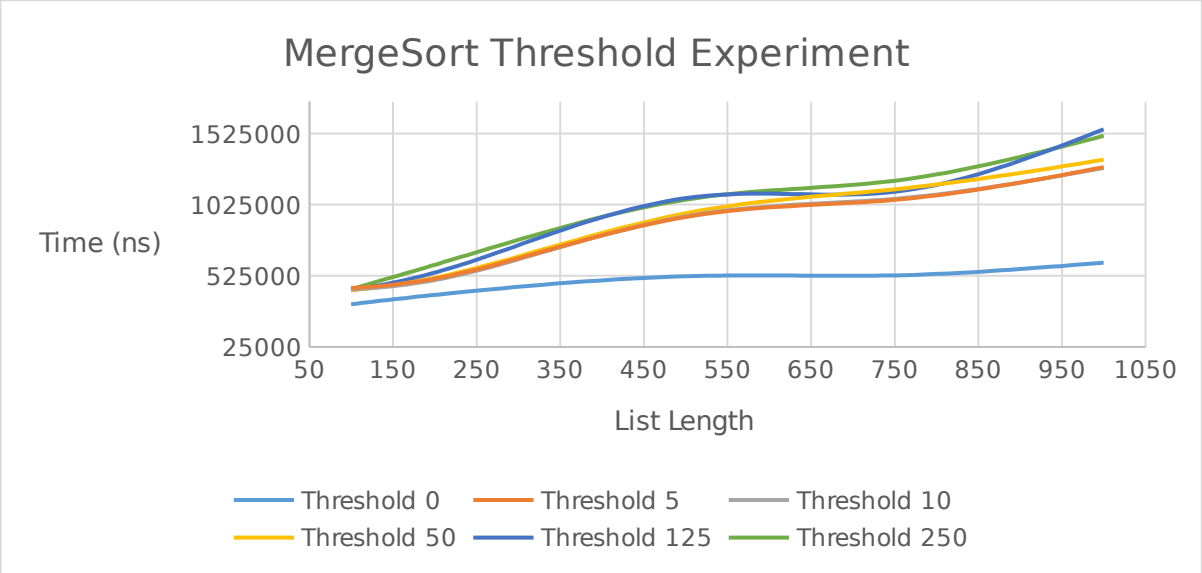
Cameron Pickle
1. Who is your programming partner? Which of you submitted the source code of your program?

Skyler Jayson. I will submit the assignment.

2. Evaluate your programming partner. Do you plan to work with this person again?

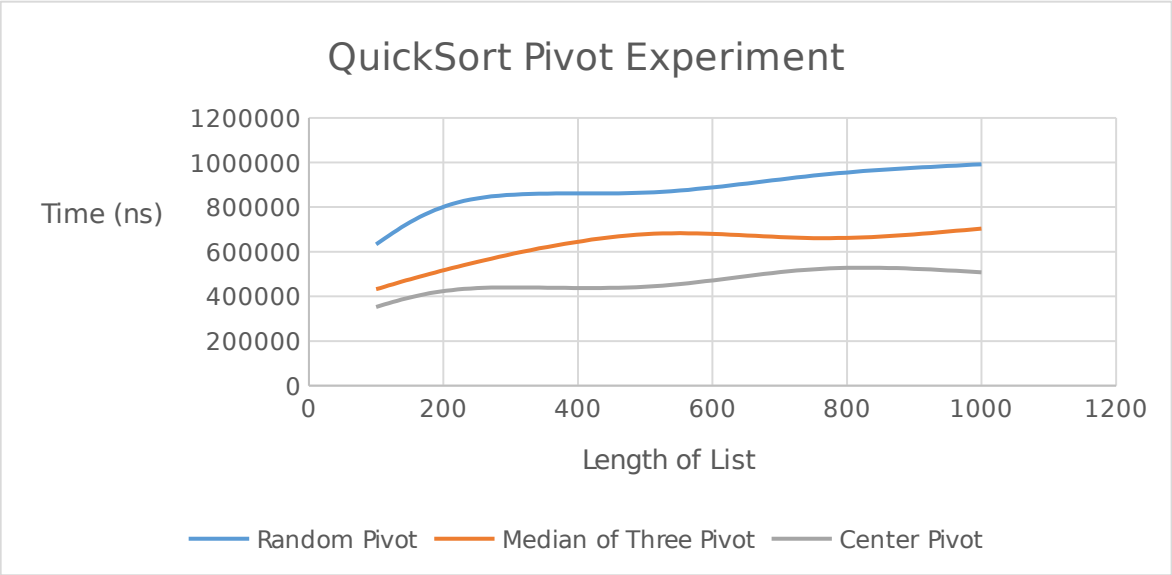
He works really hard and does a good job. We will not be working together on the next project because the professor has stated that we need to now find a new programming partner.

3. Mergesort Threshold Experiment: Determine the best threshold value for which mergesort switches over to insertion sort. Your list sizes should cover a range of input sizes to make meaningful plots, and should be large enough to capture accurate running times. To ensure a fair comparison, use the same set of permuted-order lists for each threshold value. Keep in mind that you can't resort the same ArrayList over and over, as the second time the order will have changed. Create an initial input and copy it to a temporary ArrayList for each test (but make sure you subtract the copy time from your timing results!). Use the timing techniques demonstrated in Lab 1 and be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Note that the best threshold value may be a constant value or a fraction of the list size.
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).



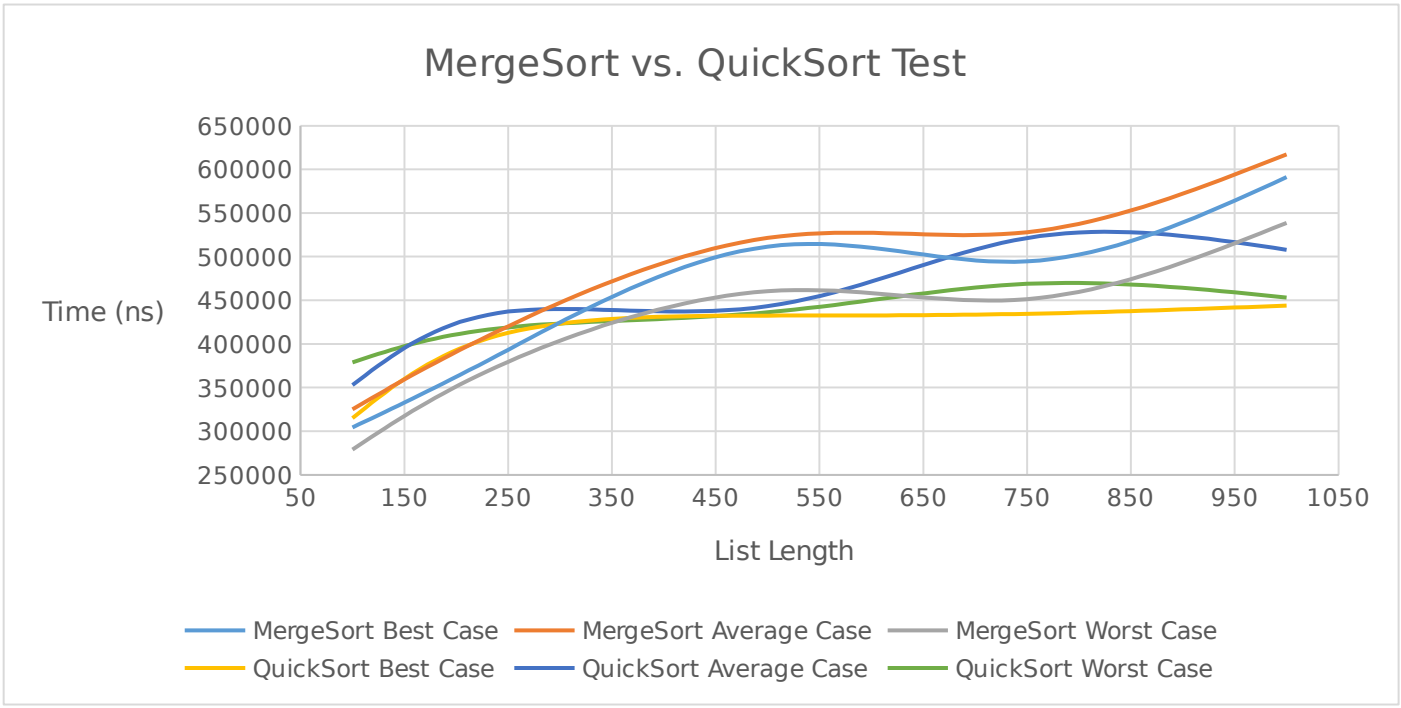
We determined that a threshold of 0 was the best.

4. Quicksort Pivot Experiment: Determine the best pivot-choosing strategy for quicksot. (As in #3, use large list sizes, the same set of permuted-order lists for each strategy, and the timing techniques demonstrated in Lab 1.) Plot the running times of your quicksort for three different pivot-choosing strategies on permuted-order lists (one line for each strategy).



We determined that a Center Pivot was the best.

5. Mergesort vs. Quicksort Experiment: Determine the best sorting algorithm for each of the three categories of lists (best-, average-, and worst-case). For the mergesort, use the threshold value that you determined to be the best. For the quicksort, use the pivot-choosing strategy that you determined to be the best. Note that the best pivot strategy on permuted lists may lead to O(N^2) performance on best/worst case lists. If this is the case, use a different pivot for this part. As in #3, use large list sizes, the same list sizes for each category and sort, and the timing techniques demonstrated in Lab 1. 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).



6. 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, both mergesort and quicksort show a logrithmic increase. We expected to see a O(NlogN) for all cases of Mergesort. We expected O(NlogN) for the best and worst-case of the quicksort and a O(N^2) for quicksort. The case that is least logrithmic O(NlogN) is the quicksort worst-case which is supposed to be closer to o(N^2) as we predicted. The mergesort was a very logrithmic sort for all the values as we expected. The results reflected the run times that we were expecting to see.

7. How many hours did you spend on this assignment?
We spent 10hours on this assignment.