Name: Mina (Min) Kim
U ID: 1054673

Note that if you use the same seed to a Java Random object, you will get the same sequence of random numbers (we will cover this more in the next lab). Use this fact to generate the same permuted list every time, after switching threshold values or pivot selection techniques in the experiments below. (i.e., re-seed the Random with the same seed)

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

My programming partner was Melody Chang. She submitted the source code for the program.

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

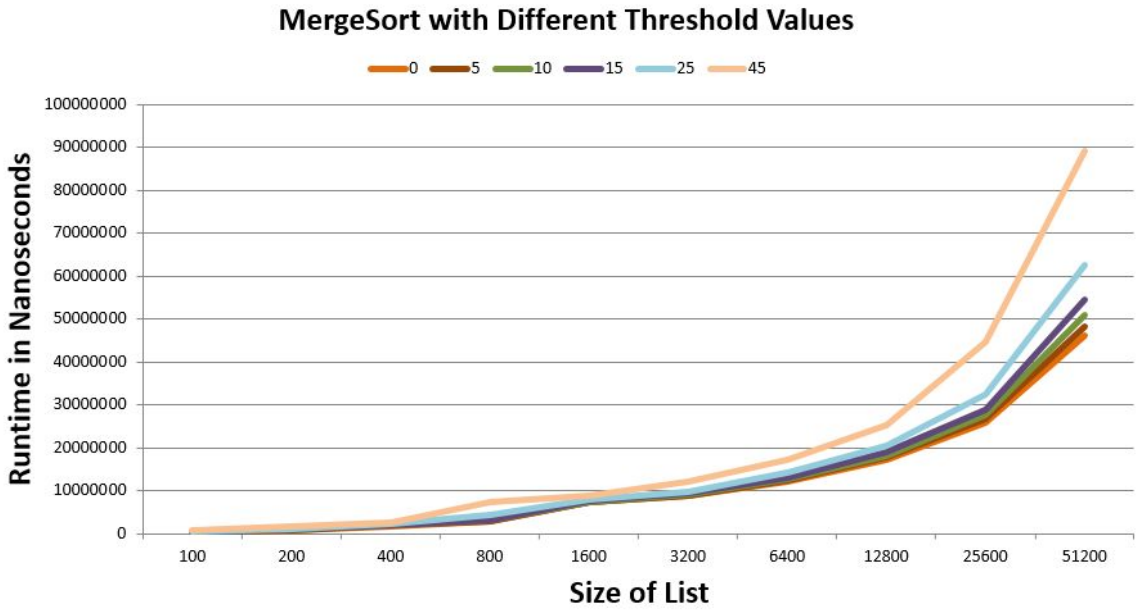
Melody was great! She totally knew what she was doing. She helped me figure out how to properly use the debugger to catch bugs, and it was thanks to her that our quicksort and mergesort both work properly. She also helped tremendously with the timing. Yes, I would love to work with her again.

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?

I really liked how we had a second pair of eyes to catch bugs in the program, and I liked that we had a helping hand for when we got stuck, rather than having to wait hours to days until a TA was available. What didn't work out so well was that even when I thought up an idea that I was certain would work, we couldn't go ahead and use that method until I'd managed to convince my partner, if at all, which was sometimes a bit of a challenge. For future assignments, I might try to be a better partner by looking over the assignment and reviewing the powerpoint slides in advance so that I'll be less of a deadweight.

4. Mergesort Threshold Experiment: Determine the best threshold value for which mergesort switches over to insertion sort.

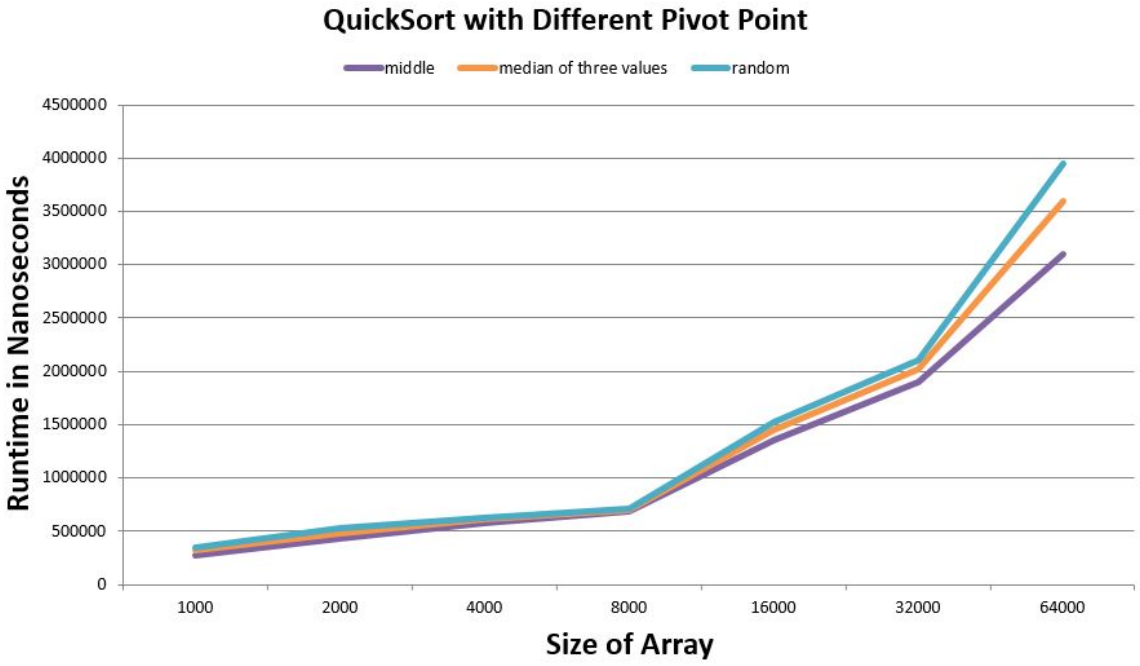
For our code, we determined that the threshold worked better if it was lower -- in fact, from the numbers we tested, it worked best when it was at zero for some reason. We accordingly decided to use 0 as our threshold.



---

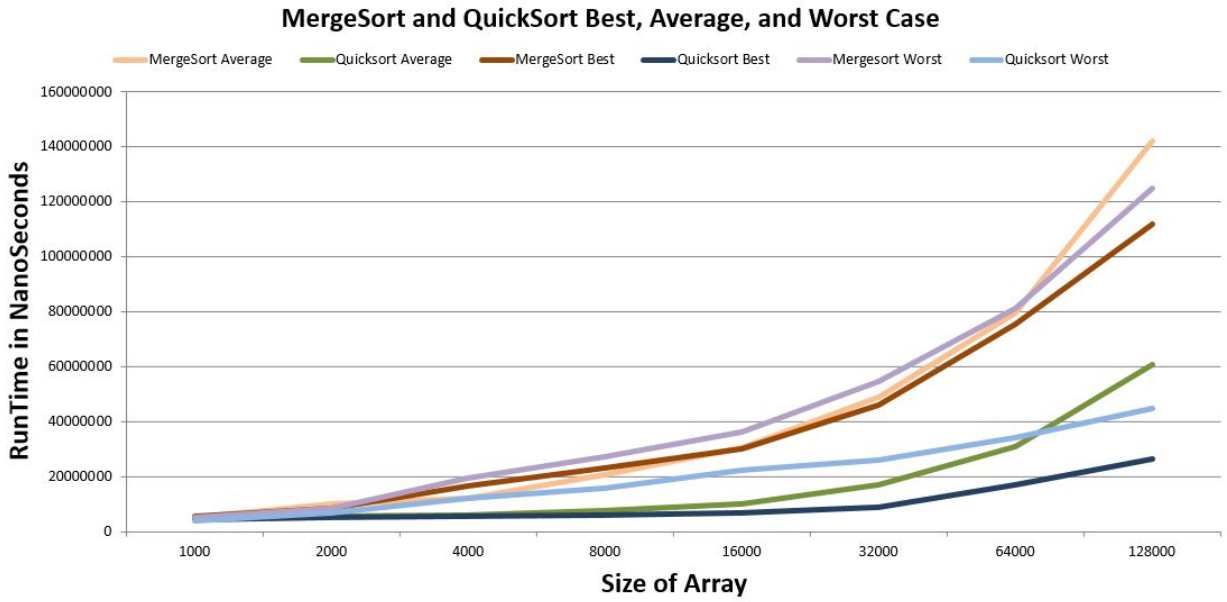
5. Quicksort Pivot Experiment: Determine the best pivot-choosing strategy for quicksort

For our quicksort, we determined that the best pivot-choosing strategy was the pivotMiddle method, which consists of using whatever element was in the middle of the array as our pivot. We thought this was strange, as we expected the “median of three values” pivot to be the most efficient.



6. Mergesort vs. Quicksort Experiment: Determine the best sorting algorithm for each of the three categories of lists (best-, average-, and worst-case).

For our code, the Quicksort proved superior in the best case, the average case, and the worst case. We thought this was strange, since we expected them to be relatively similar & for quicksort to be less efficient in the worst case.



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.

No, the actual running times of my sorting methods did not quite exhibit the growth rates I expected to see. I expected the running times for quicksort and mergesort to be relatively the same for the best and average case, and I expected quicksort to be less efficient in the worst case. However, the lines, barring that of Quicksort Worst, do look relatively like they have a big-o complexity of n log n, which was what I expected. I'm surprised by how linear the line for quicksort worst looks, since I expected the worst case to be o(n^2), and I expected the worst case to be more inefficient than the average case. This also applies to the merge sort, where the average case time complexity eventually hikes up and becomes less efficient than the worst case. I feel like quicksort is more reliant on the pivot selection than the actual content of the array, while mergesort is more reliant on the threshold selection than the actual content of the array. Perhaps this could be one of the factors that contributed to the discrepancies between our data and our expectations.

In retrospect, I also wonder if we should have used significantly larger array sizes. The array sizes we used seemed sufficiently large back when we were testing, but the graphs don't match up to our expectations as well as they could, and I feel like the array size would be the most obvious reason why. Otherwise, we tried testing the content behind our graphs multiple times. While the fluctuations were significant enough that sometimes the order of the lines would change (e.g., in some cases, the worst case time complexity would be noticeably more efficient than the average case time complexity), the rough idea and shape of the lines remained consistent.

8. How many hours did you spend on this assignment?

We spent from around fifteen to twenty hours on this assignment.