When you are satisfied that your program is correct, write a detailed analysis document. The analysis document is 40% of your assignment 5 grade. Ensure that your analysis document addresses the following.

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

   Noah Goetz is my partner. I, Zhangtuoming Zhao, will submit the source code.

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

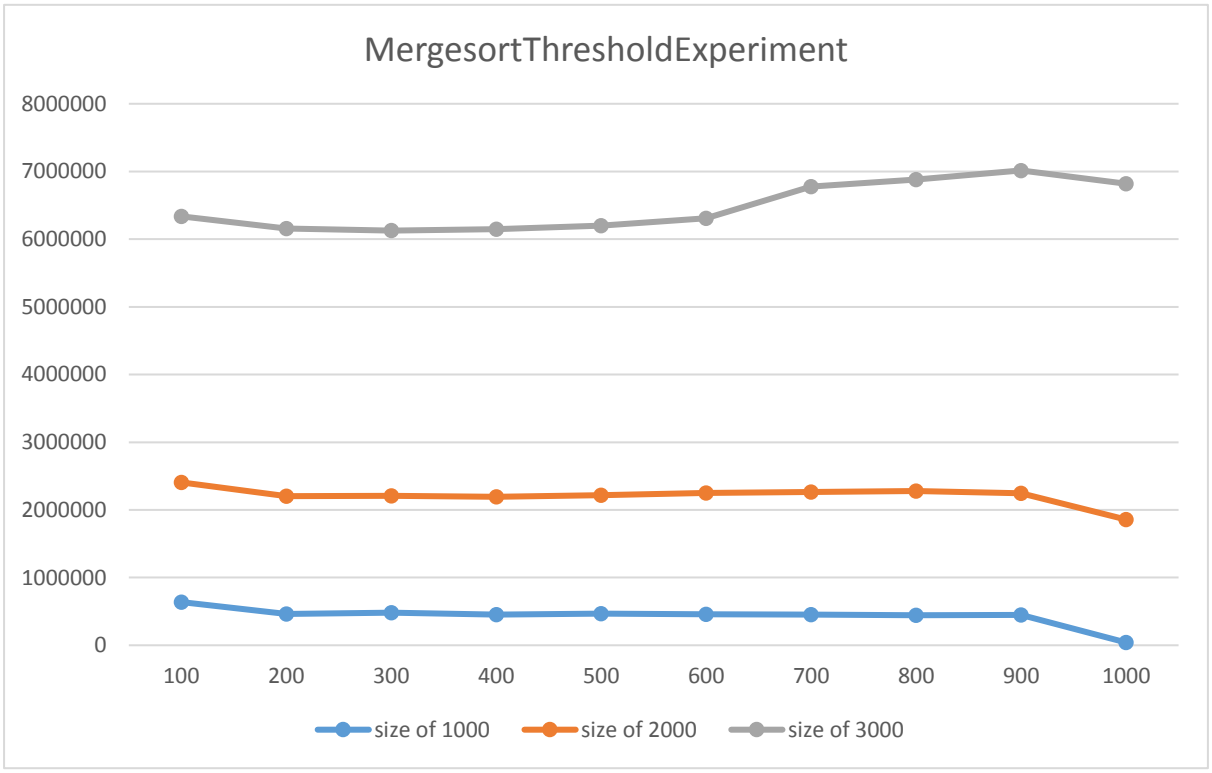
   He is a very good programmer. I will plan to work with him again.

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 think the communication is the most important thing during the programming. I really need to improve this part. I have to say that I can write the code, but I cannot present what I want to show to my partner. And sometimes, I cannot make the work more efficient. I will challenge myself to make more improvement.

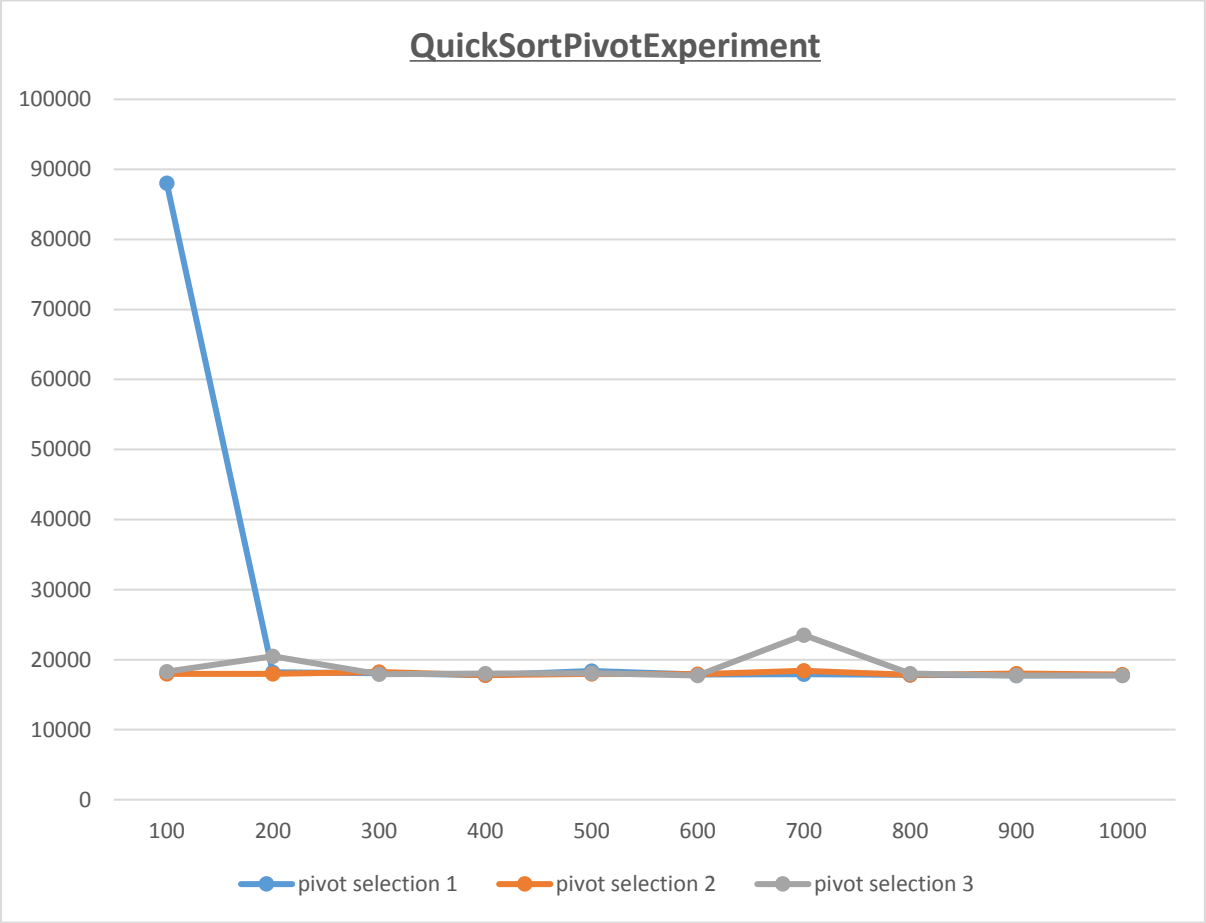
4. **Mergesort Threshold Experiment: Determine the best threshold value for which mergesort switches over to insertion sort. Your list sizes should cover a range of input sizes to make meaningful plots, and should be large enough to capture accurate running times. To ensure a fair comparison, use the 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. â ¨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).



5. **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).

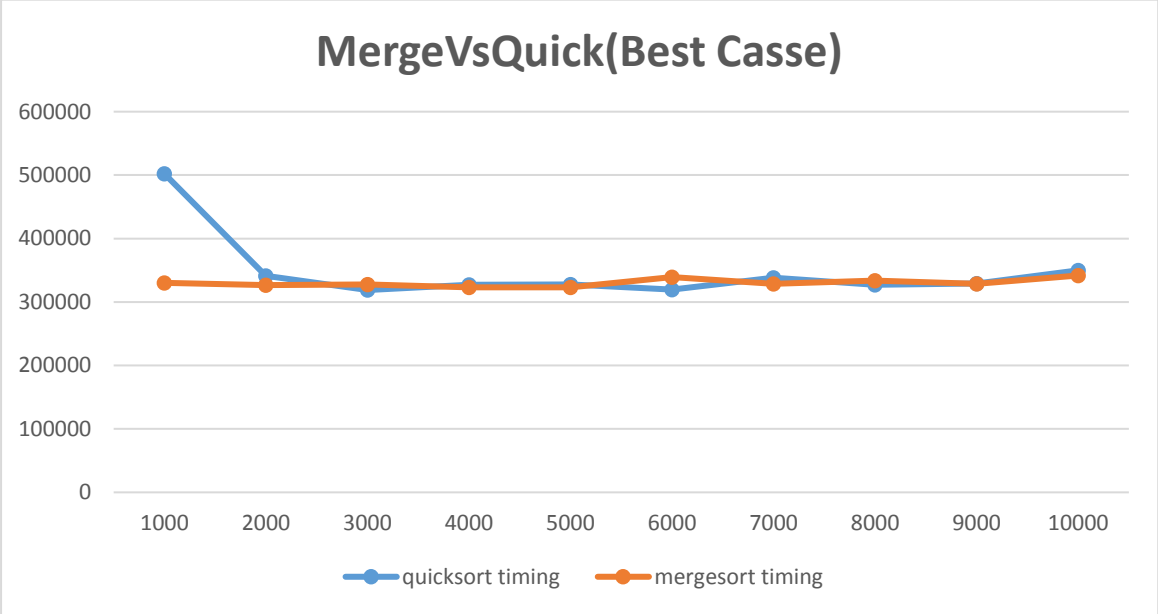


As we can see the pivot 2 is the most stable case in those 3 situations. To conclude that, the best option to choose the pivot is to choose the middle element of the arrayList that need to be sorted.

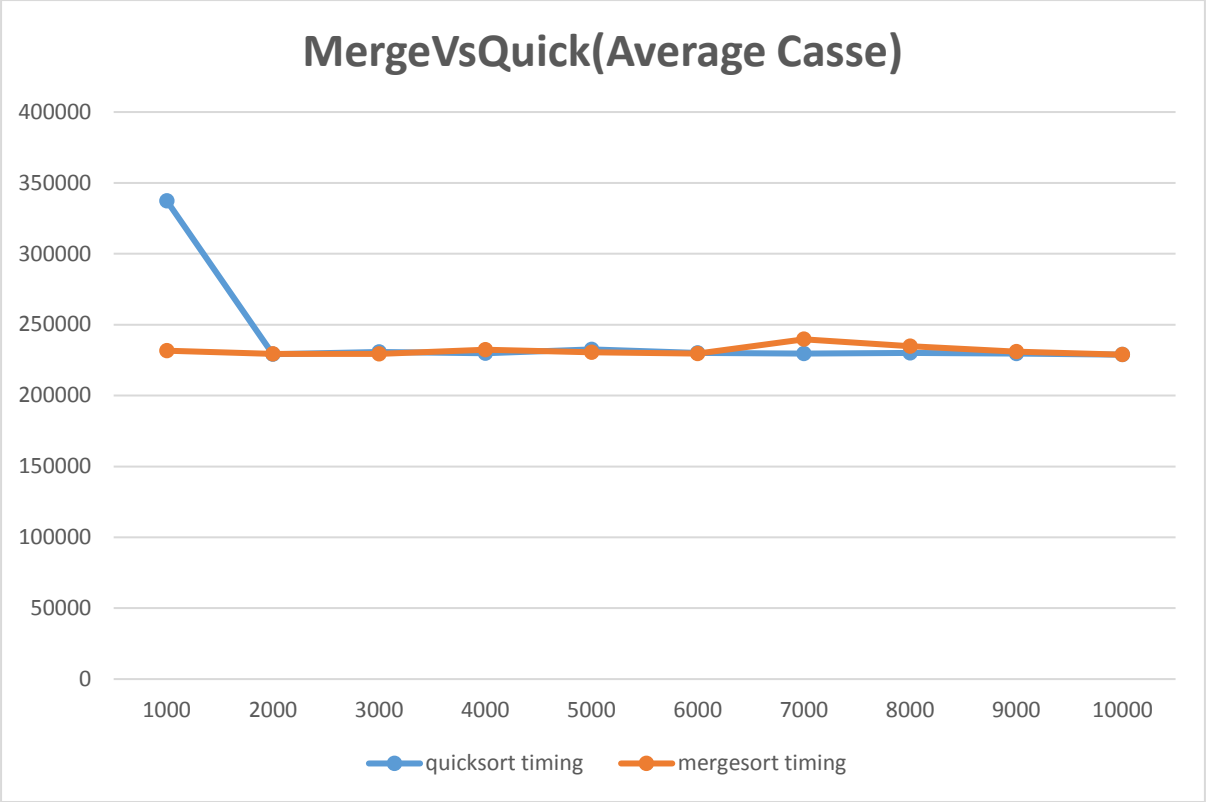
6. **Mergesort vs. Quicksort Experiment: Determine the best sorting algorithm for each of the three categories of lists (best-, average-, and worst-case). For the mergesort, use the threshold value that you determined to be the best. For the quicksort, use the**

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).

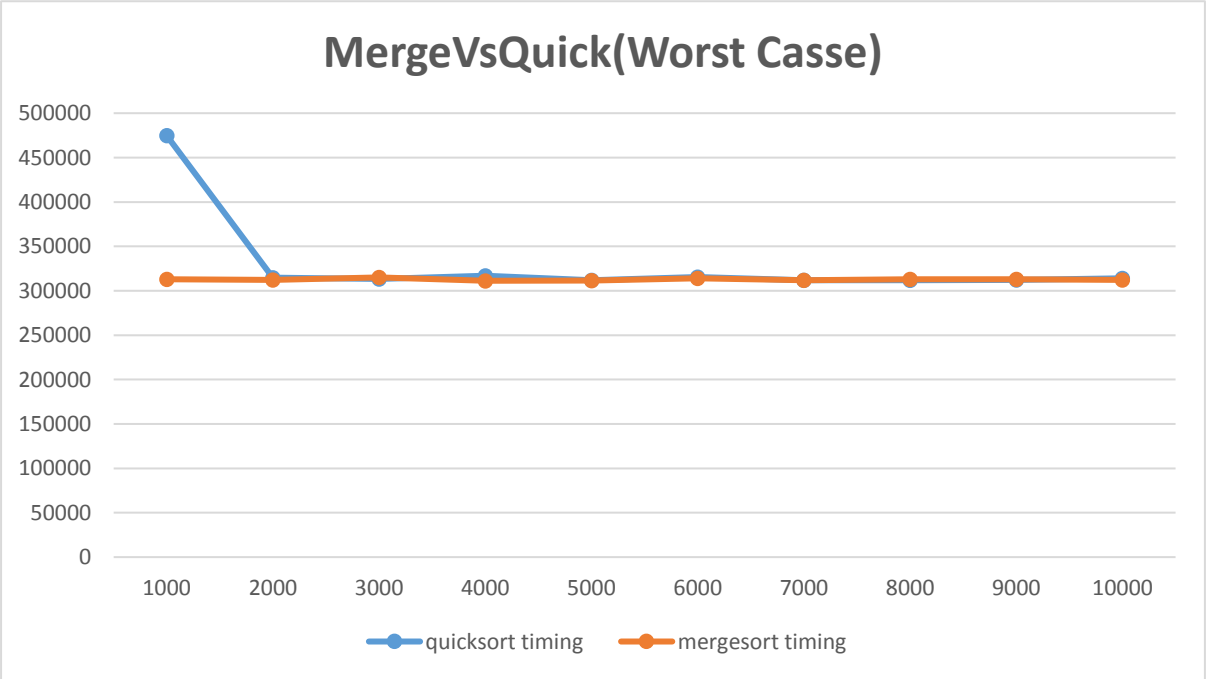
Best case:



Average case:



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.

As we can see in the last 3 graphs, when the size is growing, the sorting time is kind of a stable growth. The volatility is very limited. The growth rate is more like a O(N) instead of O(NlogN). However, when the size is small, the performance of quicksort is not very good.

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

   About 25 hours.