Nicholas Lloyd – U0949261

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

Partner: Nicholas Moore – U0847478

Nicholas Lloyd submitting the .java files.

2. How often did you and your programming partner switch roles? Would you have preferred to switch less/more often? Why or why not?

We did approximately the same amount of work in each role. With my current partner, I felt comfortable in either role, as we are both very competent programmers and communicate well.

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

My partner was really great to work with. He was flexible about scheduling but always followed through on arranged meetings. He worked hard and programmed well. I plan to work with him again on the next assignment.

4. If you had backed the sorted set with a Java List instead of a basic array, summarize the main points in which your implementation would have differed. Do you expect that using a Java List would have more or less efficient and why? (Consider efficiency both in running time and in program development time.)

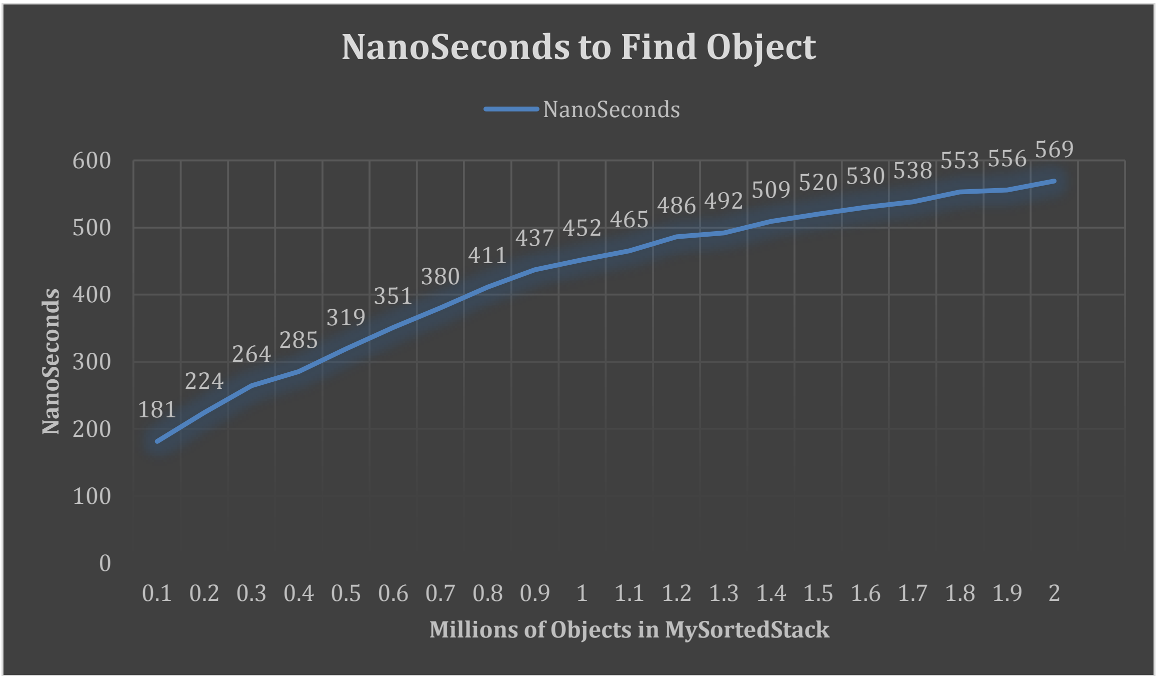
If we had implemented a List instead of an Array, the majority of the work would have been done for us. First off, we would not have had to worry about scaling the size of our structure, as this is taken care of by a list. Adding and removing Objects would have been simple calls, making sure to check for duplicates. Also, built in sorting methods would have made this much easier.

5. What do you expect the Big-O behavior of

MySortedSet's contains method to be and why?

I would expect MySortedSet's contains() method to perform at approximately O(log(N)) complexity, as doubling the size of the array only creates one extra loop.

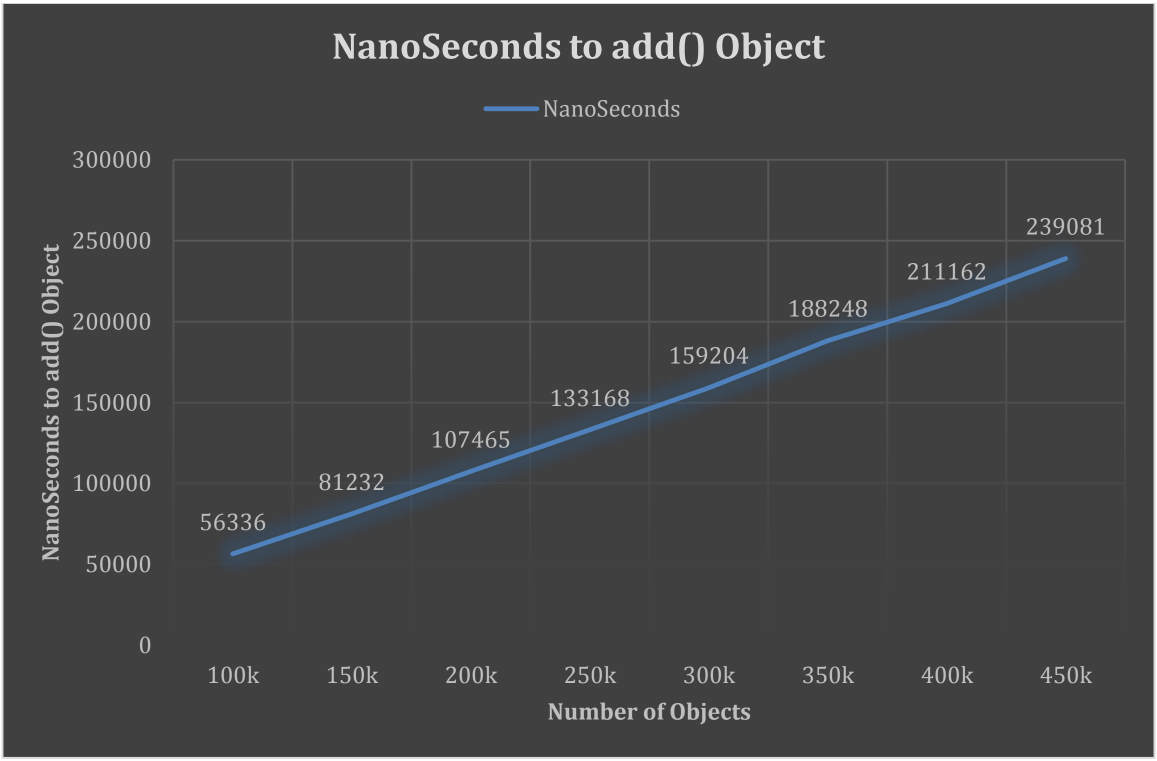
6. Plot the running time of MySortedSet's contains method for sets of sizes 100000 to 2000000 by steps of 100000. Use the timing techniques demonstrated in Lab 1. Be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Include your plot in your analysis document. Does the growth rate of these running times match the Big-oh behavior you predicted in question 5?



Each loop ran 10m times to be certain it was averaging correctly. The data does seem to be pointing at a generally logarithmic complexity.

7. Consider your add method. For an element not already contained in the set, how long does it take to locate the correct position at which to insert the element? Create a plot of running times. Pay close attention to the problem size for which you are collecting running

times. Beware that if you simply add N items, the size of the sorted set is always changing. A good strategy is to fill a sorted set with N items and time how long it takes to add one additional item. To do this repeatedly (i.e., timesToLoop), remove the item and add it again, being careful not to include the time required to call remove() in your total. In the worst-case, how much time does it take to locate the position to add an element (give your answer using Big-oh)?



The graph makes it appear that the complexity of the add() function is O(N), which makes sense since more elements would on average have to be shifted when adding in the internal array the larger the arrays size.

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

Approximately 15 hours.