When you are satisfied that your program is correct, write a brief analysis document. The analysis document is 30% of your Assignment 3 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?

---Jeramie Mcdonough, He submitted the code for the assignment.

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

---Very often probably every half hour we swapped roles but before moving forward we discussed our next moves together. The swap time frame was perfect because it gave us to catch errors more efficiently.

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

---Yes Absolutely.

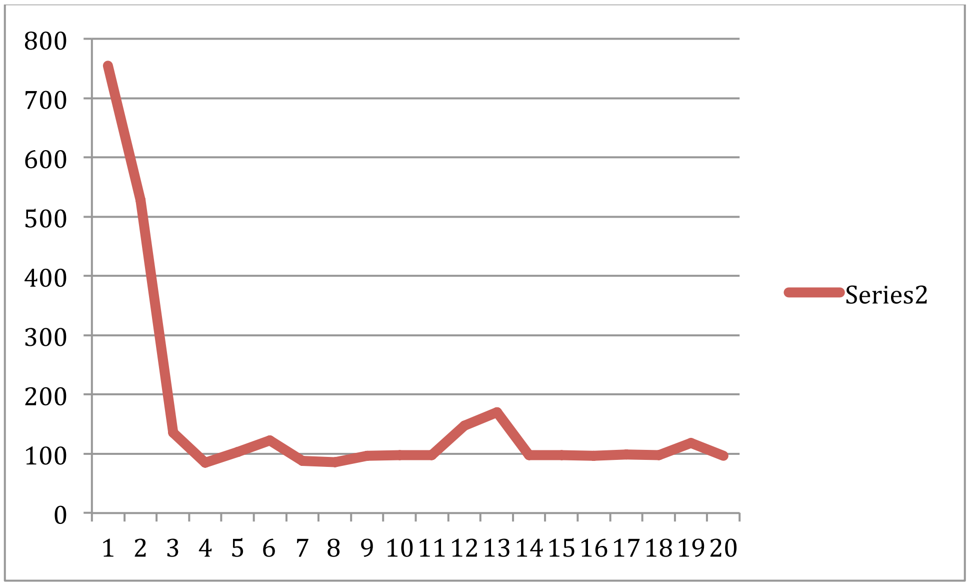
4. If you had backed the sorted set with a Java List instead of a basic array, summarize the main points in which your implementation would have differed. Do you expect that using a Java List would have more or less efficient and why? (Consider efficiency both in running time and in program development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

---Well the Java list would allow duplicates that would totally throw a wrench in the add methods. On the other hand the Java list is ordered and the set is not pre ordered. But our sort functions that utilize comparable take care of this to make the collections sorted anyway. I believe the sorted set has much more functionality and freedom to compare and sort in the manner you dictate making it more efficient in the long run.

5. What do you expect the Big-O behavior of MySortedSet's contains method to be and why?

---2N(LogN)/2 because our contains runs our Binary search to find the position which is N(logN)/2 and then comes out to grab the found element and return if it found it or not making that operation N. Overall making the whole method 2N(LogN)/2.

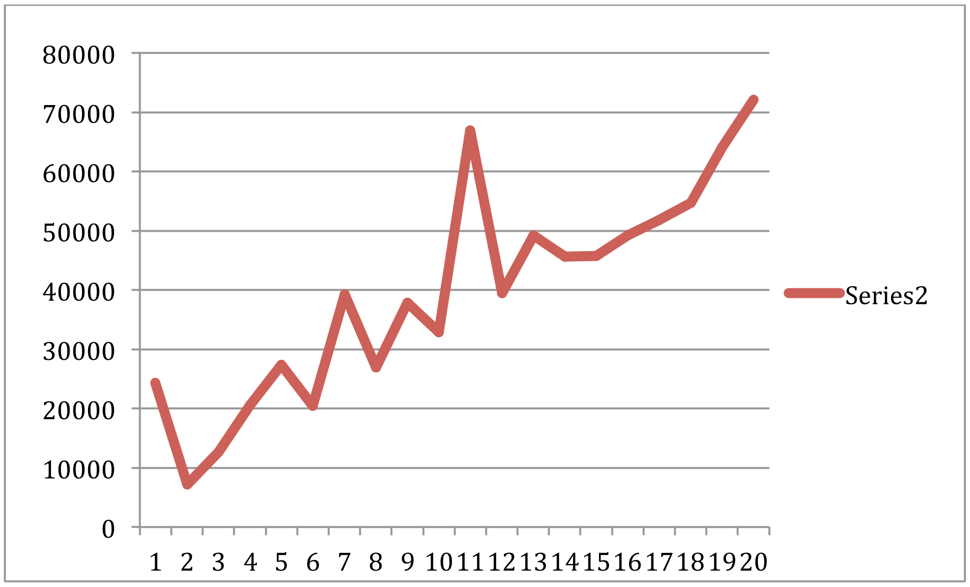
6. Plot the running time of MySortedSet's contains method for sets of sizes 100000 to 2000000 by steps of 100000. Use the timing techniques demonstrated in Lab 1. Be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Include your plot in your analysis document.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



**I had to reduce the values of increment because Eclipse was not functioning correctly with the larger values.

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)?



N is what it looks like from here as this is pretty linear in growth.

**I had to reduce the values of increment because Eclipse was not functioning correctly with the larger values.

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

---We spent about 15 to 20 hours on this assignment.

Programming partners are encouraged to collaborate on the answers to these questions. However, each partner must write and submit his/her own solutions.

Upload your document (.pdf only!) to the Assignment 3 page by 11:59pm on February 5.