Tanner Martin u0718593
CS 2420 Spring 2015

1. Who is your programming partner? Which of you submitted the source code of your program?
   **Kincaid Savoie was my partner. He is the one who submitted the source code.**

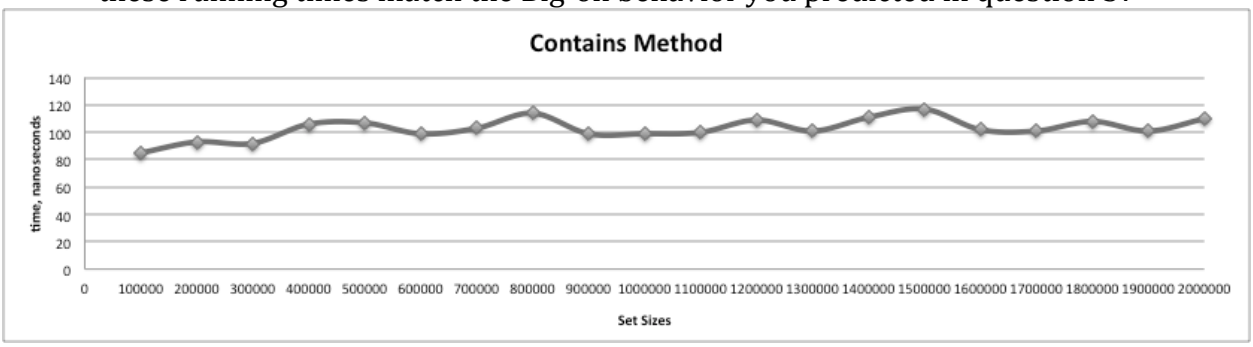
2. How often did you and your programming partner switch roles? Would you have preferred to switch less/more often? Why or why not?
   **We switched quite a few times. I think what we did worked fine.**

3. Evaluate your programming partner. Do you plan to work with this person again?
   **I really like working with Kincaid, he is a great programmer and I will definitely be working with him again.**

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 used one of the various Java lists to back our sorted set we wouldn't have had to worry about copying over and making new arrays to be able to hold and remove date. Lists already implement these methods so we could just reuse them.**

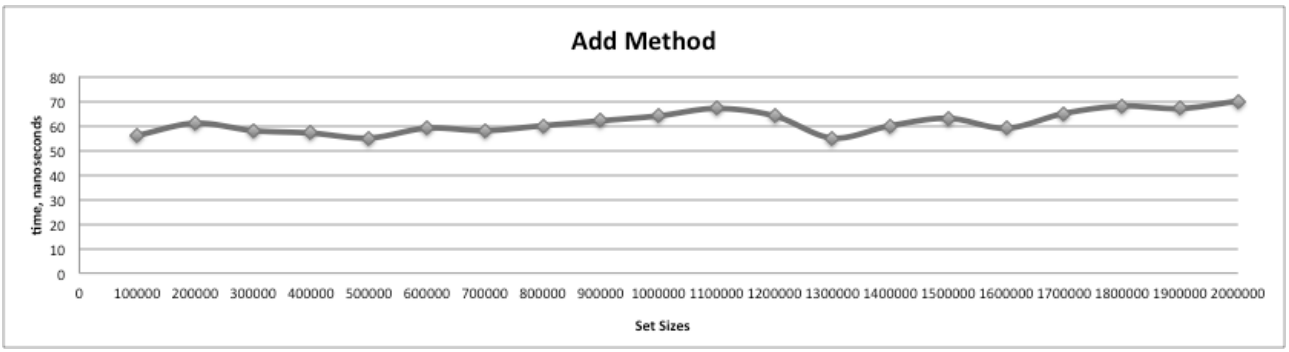
5. What do you expect the Big-O behavior of MySortedSet's contains method to be and why?
   **I would say log N because we are using a binary search.**

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?




**Some of the timing for my computer was a bit scattered, but it does mostly look like O(log N).**

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)?




**This also looks like an O(log N) graph.**

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

**About 6.**