Steve Corey
u0110170
Assignment 3

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

2. How often did you and your programming partner switch roles?
We were about evenly distributed to both roles.

Would you have preferred to switch less/more often? Why or why not?
The switching worked out nicely. We mainly switched when we were feeling tired, and the switch gave an opportunity for a brain refresh.

3. Evaluate your programming partner.
Rebecca is excellent to work with. She stays upbeat even when we're getting frustrated with the assignment. We were able to help each other when we got stuck.

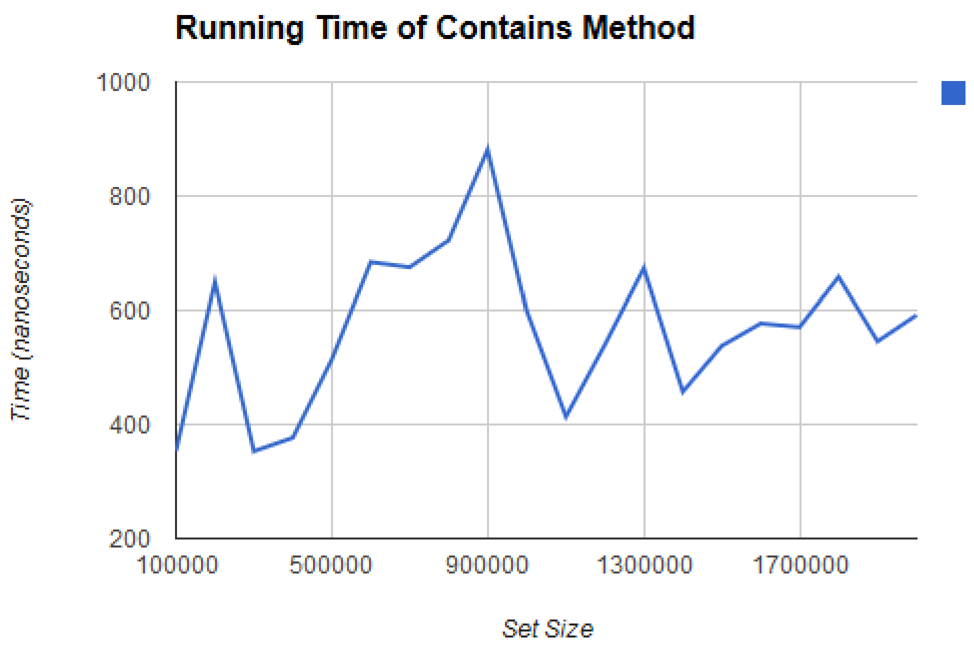
Do you plan to work with this person again?
Absolutely.

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.
Looking at the Java API, it appears that many of the methods that we had to write would already be taken care of in the List, like an iterator. But it also appears that the built-in search would probably not be as efficient.

Do you expect that using a Java List would have more or less efficient and why?
More efficient to code, less efficient in running time. See above response as to why.

5. What do you expect the Big-O behavior of MySortedSet's contains method to be and why?
O(logN) due to the set maintaining its sort status, and the search being used halving the number of items to potentially search through during each iteration of the search.

6. Plot the running time of MySortedSet's contains method for sets of sizes 100000 to 2000000 by steps of 100000.



Does the growth rate of these running times match the Big-oh behavior you predicted in question 5? Yes. It's a very noisy plot, but you can see a logN plot in there.

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)? O(N log N)

8. How many hours did you spend on this assignment? About 20.