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?

Ethan Anderson. Ethan 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?

Very often, less than 10 minutes. I think this is about the best way to code because we each get a shot at editing the code.

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

He's a good guy. I like working with him. I hope we will work together 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.)

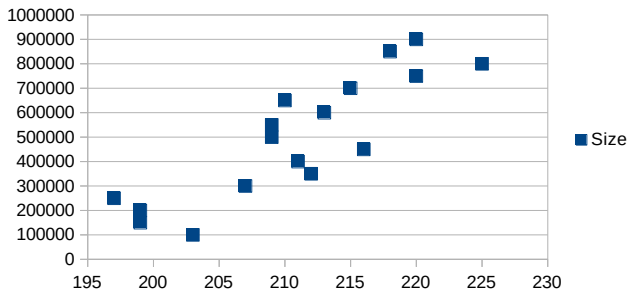
It would have been really nice to back it into a java list because it always would have been sorted. The binary search/sort method was really tough because it would search where an element needed to be, which is not traditional binary search. With a list it would be sorted always.

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

O(Log(n)) because it is only calling the binary search and returning true or false.

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?

It does match the big-Oh behavior because the average of the points exhibits log-like behavior. We performed tests from 100,000 to 1,00,000, which Jason said advised us to do. Here are the plotted results:



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)?

For an element that isn't in the set, this means worst case situation which would be O(Log(n)). We need to find where the element goes, and we need to place it. Then we need to move all of elements over, which is constant. If it were worst case scenario, then it would be sorted as the very last element or the very first. This is the cool thing about binary search, it starts in the middle and either makes its way to one end or the other. So it's still just O(log(n)).

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

We spent about 15 hours. We were stuck on binary search for a long time.

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.