Sierra Allred u0740878

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

Ajmal Esa, and he is submitting our 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 got Saros and synced our laptops. So we basically sat next to each other and both typed and collaborated at the same time. It was very effective.

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

Yes he works hard and does his share of the work.

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.)

Using a List would have been a lot more efficient. It would have made iteration a lot more efficient, and I wouldn't have needed to create a binary search, which not only took a long time to develop, but it's running time is probably pretty slow. It also has built in methods for add and remove, which would have saved me a lot of time.

5. What do you expect the Big-O behavior of MySortedSet's contains method to be and why?
I think it would be O(logN) because we call on binary search and that splits the list in half each time as it looks for the correct index.

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's hard to tell, but it looks like I predicted correctly. If I were to draw a line through the center of the plot it wouldn't have a slop great enough to be O(NlogN) but it is definitely growing.

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)?

At the very worst I would say it could take up to O(N2) time , but the trend in the graph seems to show a growth of O(NlogN). I didn't feel like our add method was very efficient at all, so this didn't surprise me.

8. How many hours did you spend on this assignment?   15 hours.