Tanner Marshall
CS 2420

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

Tim Dorny. He 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 did not switch very often. I was fine with the way it was. He understood more than me and help along the way.

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

Very good, yes I do.

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 easier to deal with the sizing of the array. The last() method would also have been easier because we would not have needed to keep track of the elements in the array. I think a list would have been more efficient. Lists already have a lot of the things we implements already done.

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

Log(n) because it splits the array in half every 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?

I could only get up to 400,000 sizes but my results up until then were
100000 1101 nanoseconds
200000 1165 nanoseconds
300000 1353 nanoseconds
400000 1338 nanoseconds
500000 1563 nanoseconds

After 500000 each run was taking incredibly long amounts of time
But from my 4 results it looks as though it would continue in long(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)?

Worst case: nlog(n)
Best case: n

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

10