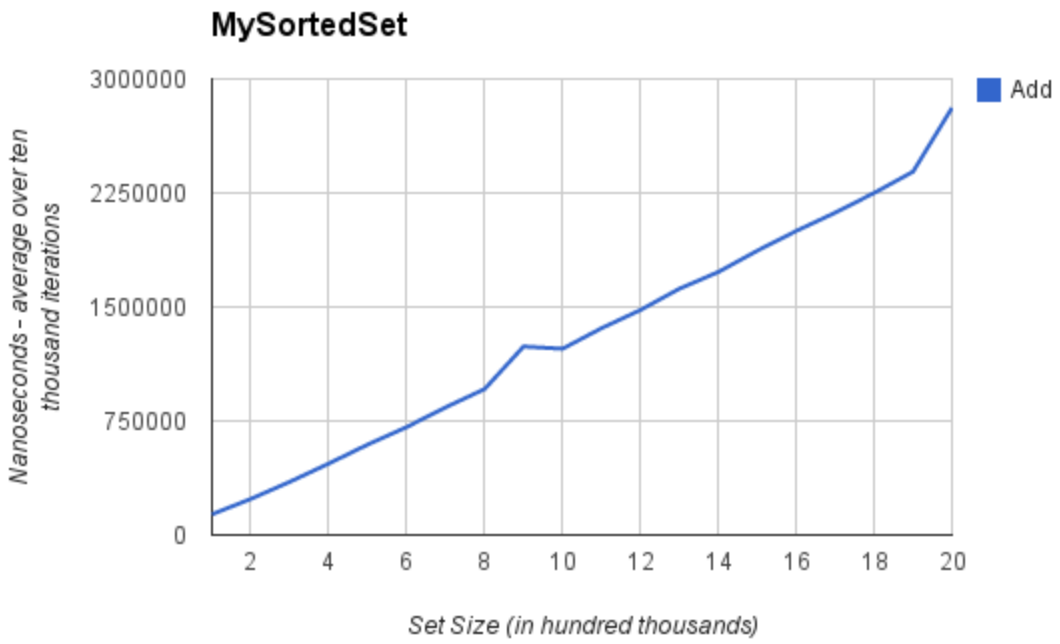
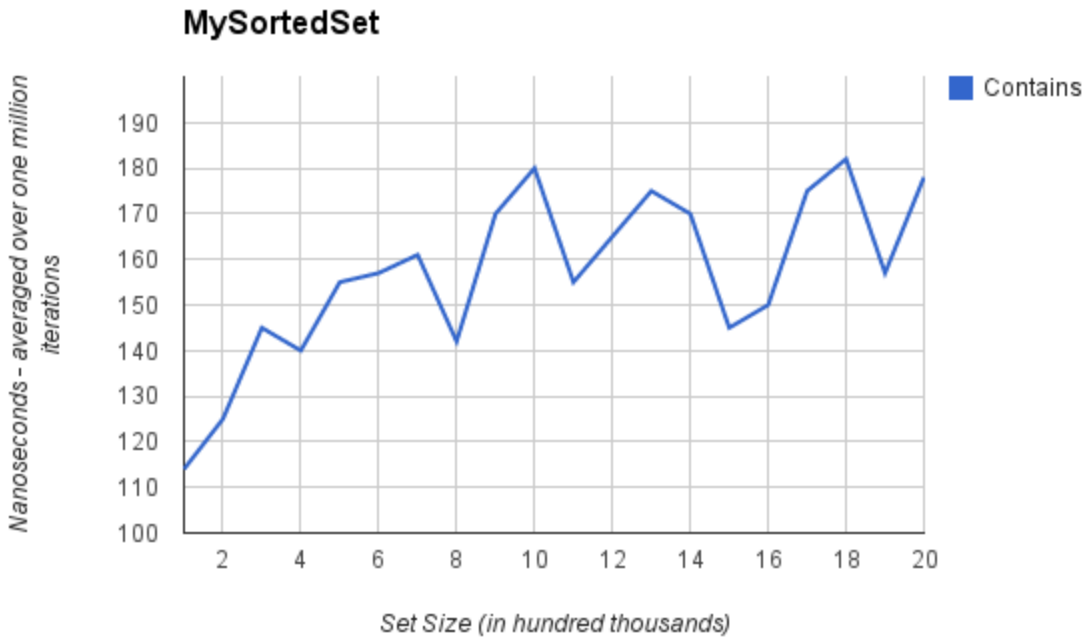
1. Who is your programming partner? Which of you submitted the source code of your program?
-Domas Piragas, and he submitted the source code.
2. How often did you and your programming partner switch roles? Would you have preferred to switch less/more often? Why or why not?
-Fairly frequently. Less, but not by much. Our switching was natural and fluid.
3. Evaluate your programming partner. Do you plan to work with this person again?
-We worked together really well. Usually one of us could come up with an answer to a problem when the other couldn't. I do plan on 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.)
-We wouldn't have had to make sure that the array was large enough to handle things because the List would have taken care of that for us. That alone would have made at least the programming development time more efficient. The List itself would have had it's own contains and remove methods, so realistically the only major amount of programming is the add method to make sure it was sorted. Because of this, both running time and development time would have been more efficient.
5. What do you expect the Big-O behavior of MySortedSet's contains method to be and why?
-N log N, since that is the behavior of binary searches.
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?
-Yes
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)?
-N
8. How many hours did you spend on this assignment?
-16


MySortedSet
Add
Nanoseconds - average over ten thousand iterations
3000000
2250000
1500000
750000
0
2
4
6
8
10
12
14
16
18
20
Set Size (in hundred thousands)


MySortedSet
Contains
Nanoseconds - averaged over one million iterations
190
180
170
160
150
140
130
120
110
100
2
4
6
8
10
12
14
16
18
20
Set Size (in hundred thousands)