Richie Frost
CS 2420
2/5/14
Assignment 3 Analysis

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

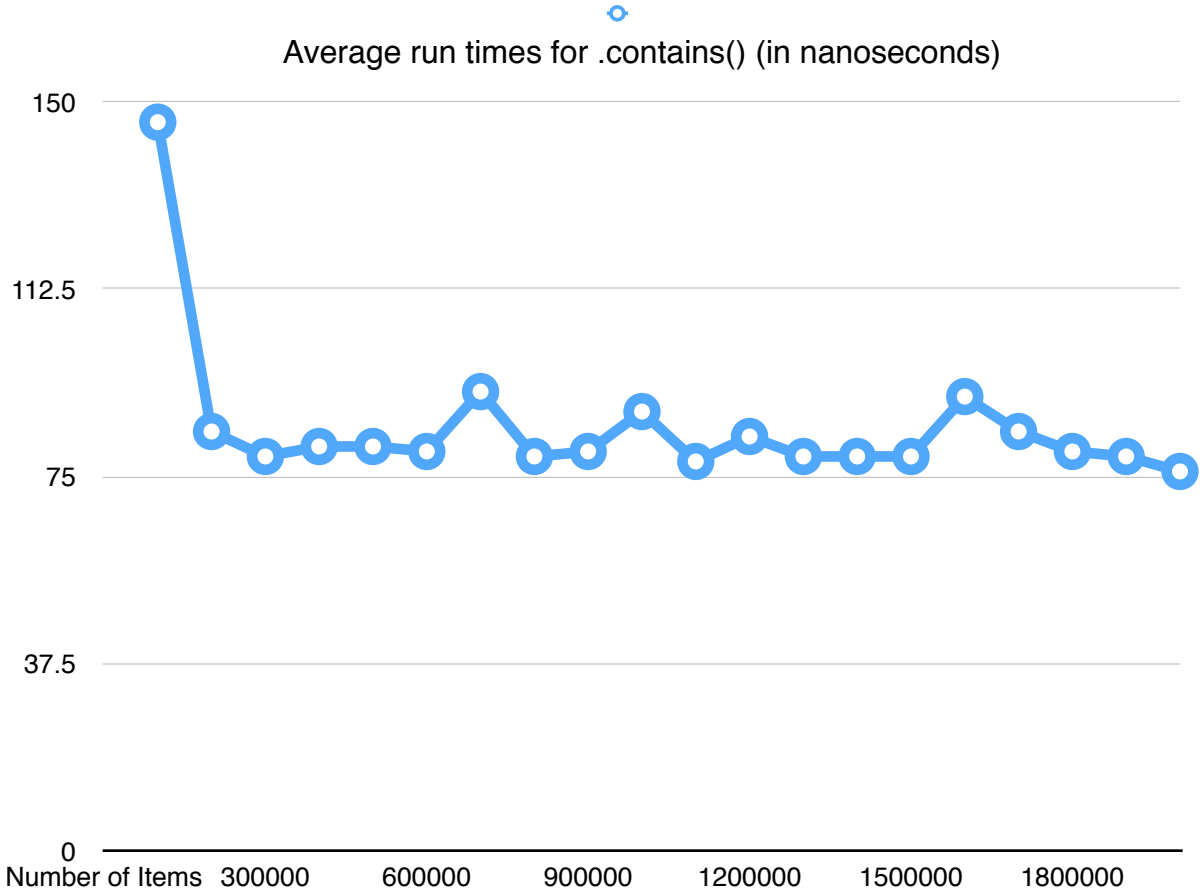
2. How often did you and your programming partner switch roles? Would you have preferred to switch less/more often? Why or why not?
We didn't switch very often, but I think I'd like to try navigating next time. I like not switching very much since it helps us get into a groove, then it's hard to get back in again if we switch too often.

3. Evaluate your programming partner. Do you plan to work with this person again?
Ryan was impressive and really helpful both as a navigator and as a driver. We put in some long hours on this project and it really helped to have a second set of eyes on the project. He's dedicated and committed to getting things done early, so that made our job a lot easier. I'll probably work with him again in the future.

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.)
I think it would have been different with a Java List because we would've needed to use an iterator of *everything*, and it wouldn't have been as easy to add items where we wanted and have the same level of control as we had with the basic array. It wasn't as easy, but it does add a little of a performance boost because there aren't as many layers to get through and you can search any item in the array. This makes it so you can do a binary search more easily, whereas the List is more for O(N) searching. The primary reason I would've used a Java List is that it would've been much easier to program, so that wouldn't have taken nearly as much time.

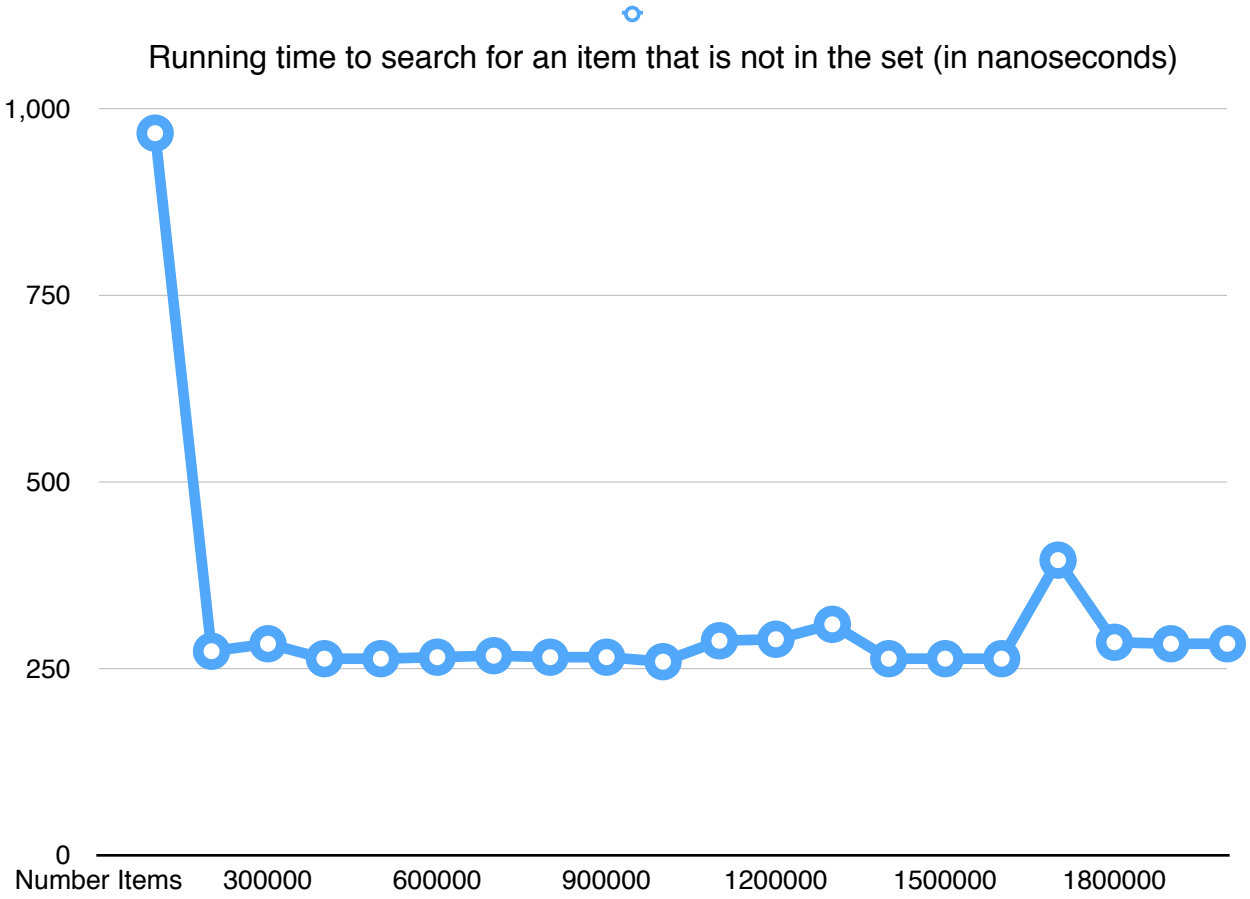
5. What do you expect the Big-O behavior of MySortedSet's contains method to be and why?
I expect the time complexity to be O(logN), because we're using a binary search to find the item in question.

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?
Since the sizes are basically linearly changing in this set (from iteration to iteration), the progression looks basically linear (even though it's actually logarithmic). It closely resembles the predictions I made in question 5. It's crazy how much faster binary search is!


Average run times for .contains() (in nanoseconds)
150
112.5
75
37.5
0
Number of Items
300000
600000
900000
1200000
1500000
1800000

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 scenario, it would take O(LogN) to find the position to add an element (since we're using binary search).



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

I spent a total of 22.5 hours on this assignment. Most of that time was spent writing unit tests (we had 78 in total) and making sure they worked ok. The majority of the remainder of my time was spent researching generics and figuring out how to handle the Comparable interface being used (or not) by items being passed in.