Neeka Ebrahimi
u0671080

Assignment 3 Analysis

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

- Stacey Kirby. I 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 switched pretty regularly depending on who felt more comfortable with the method being worked on at the moment. I wouldn't have preferred it any differently. We seemed to have worked efficiently this way.

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

- Stacey is a great partner to work with. We are able to collaborate well together. I definitely plan on working with her 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.)

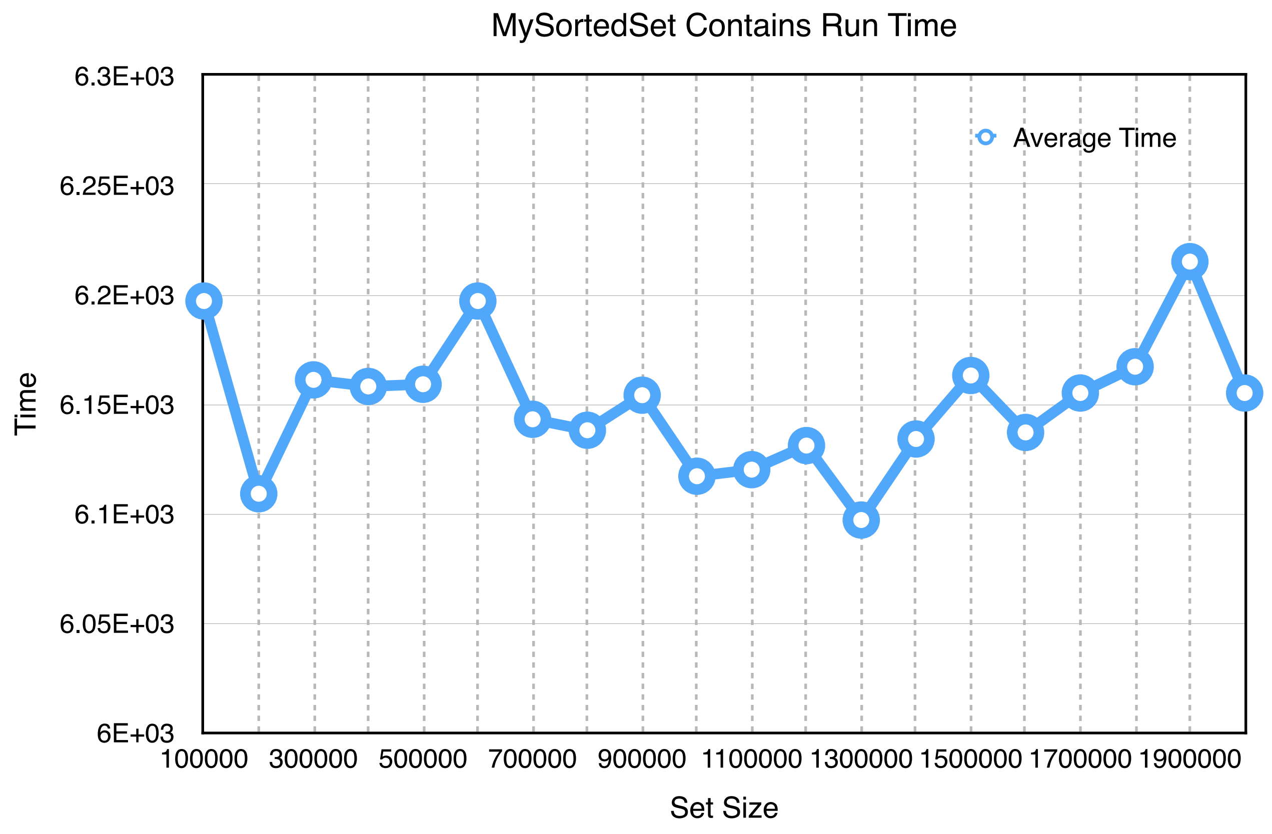
- Using a Java List seems like it would be more efficient for both program development time and run time. A Java List is an ordered collection that has control of where each item in the list is inserted, therefore we wouldn't have had to figure out where to insert during implementation. The only problem with List is that it allows duplicates. It is a quick fix to not allow duplicates by searching for the item in the array before adding it, but unfortunately this adds a lot to the running time. Regardless, this would still be faster than MySortedSet because we have to search for duplicates in that as well.

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

- I expect that the Big-O behavior for MySortedSet's contains method with be O(N) for the worst case scenario and O(constant) for the best case scenario. It would be O(N) if the item to be found is at the very end of the list since we would have to loop through each item. If the item is at the very beginning of the set then it would be O(constant) since we would just need to look at the first item.

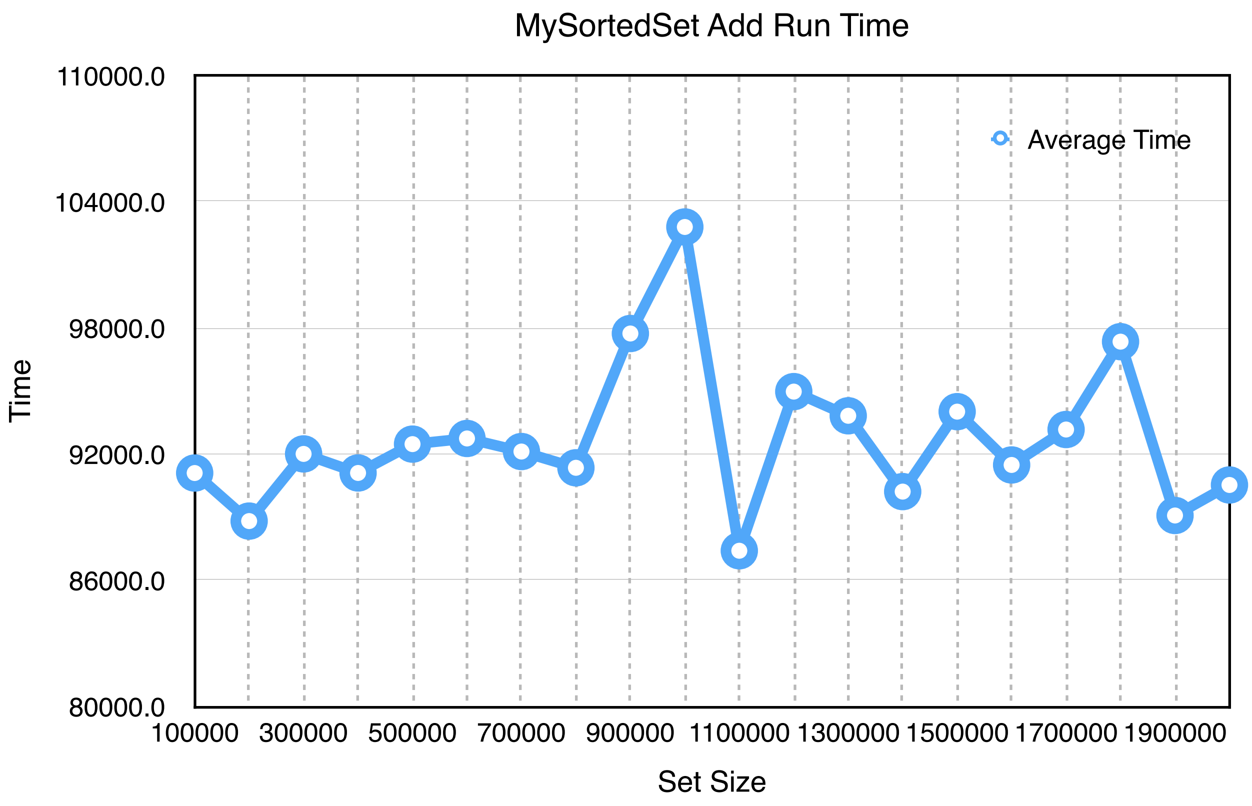
6. Plot the running time of MySortedSet's contains method for sets of sizes 100000 to 2000000 by steps of 100000. Use the timing techniques demonstrated in Lab 1. Be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Include your plot in your analysis document. Does the growth rate of these running times match the Big-oh behavior you predicted in question 5?

  - The growth rate seems to be linear like predicted.



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)?

- The Big-O behavior for the worst case seems to be O(N^2).



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

- I'm not exactly sure. Around 8 or 9.