Kincaid Savoie
Analysis Document
Partner: Tanner Martin

When you are satisfied that your program is correct, write a brief analysis document. The analysis document is 30% of your Assignment 3 grade. Ensure that your analysis document addresses the following.

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

My partner was Tanner Martin. 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 maybe every hour or so, whenever we got a method or test implemented. I think it worked pretty well, just because switching between methods seems like a good stopping point.

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

Tanner was great to work with. He was efficient, a competent coder, and offered valuable insight on how we completed the project. I'll definitely work 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.)

Looking at the documentation for java.util.List, it seems like List is an interface that a class would implement. Looking at the Java classes that implement this interface, I would probably use ArrayList, since it offers a few methods that basically do exactly what some of my methods in the assignment did.

The main things it would change about my assignment are:
-The add method. I would no longer have to copy all the data from one array to another, or shift them over myself. I'm guessing Java does something very similar to what we did in the background, but they probably implemented it more efficiently, which would lead to a decrease in runtime. I would still need to locate the index where I need to insert the object, and check for duplicates, however. The remove method is also implemented in a similar manner, so I wouldn't have to worry about shifting things left.

-ArrayList has built-in addAll and removeAll methods, so I could just use those instead of my own. They are probably more efficient than mine.

-The contains method would probably just use the ArrayList's indexOf method, which would speed up runtime a bit.

-The built-in toArray method would be used to return the list as an array.

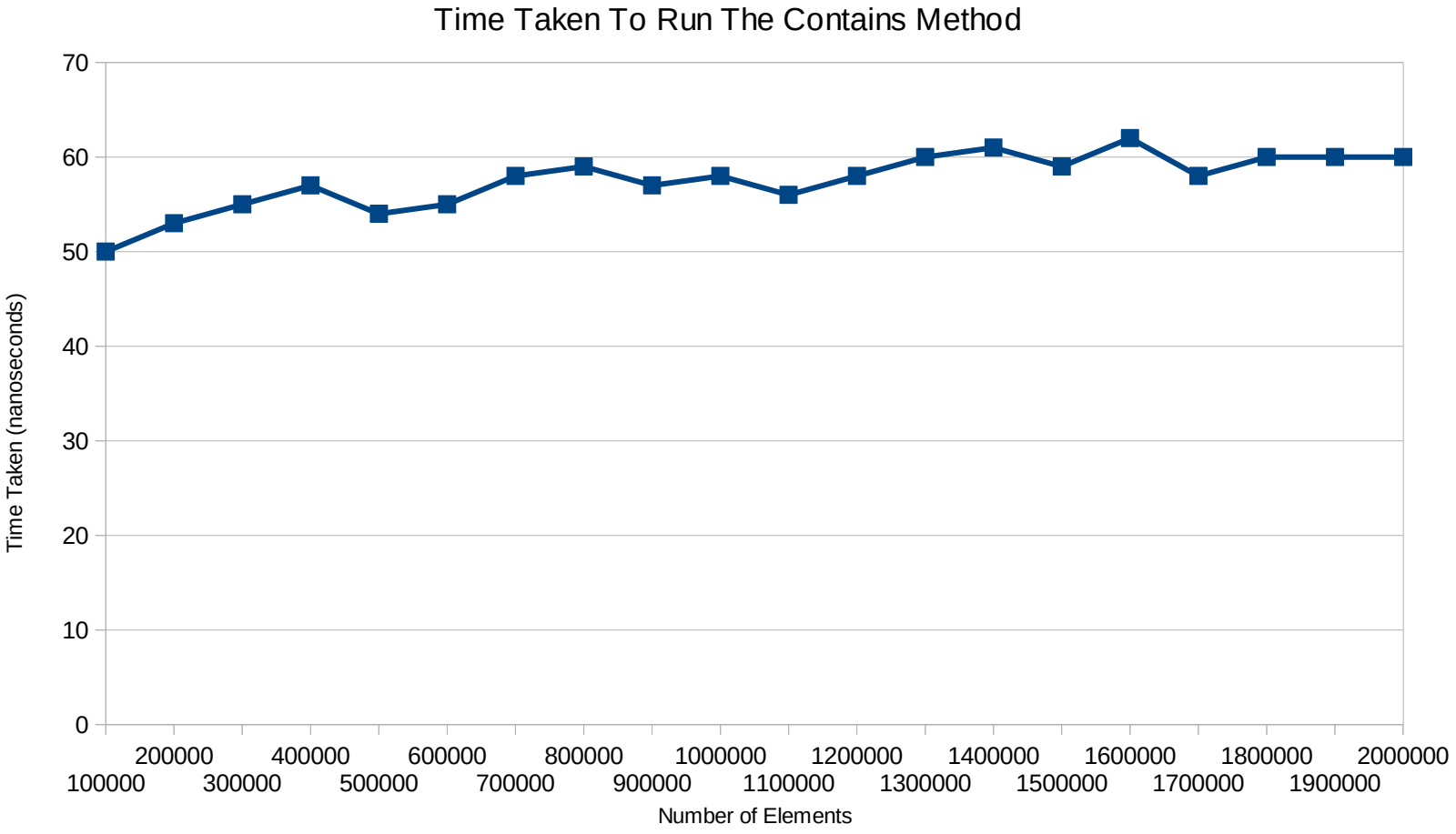
-There is a built-in clear method that I would also use, though the operation is so simple I doubt it would speed up runtime very much, or at all.

Basically, I would just take advantage of some methods that are implemented as a part of the list interface, that would probably be implemented better than I could, so it would be more effiecient.

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

Since it uses a binary search algorithm, it is probably O(log N).

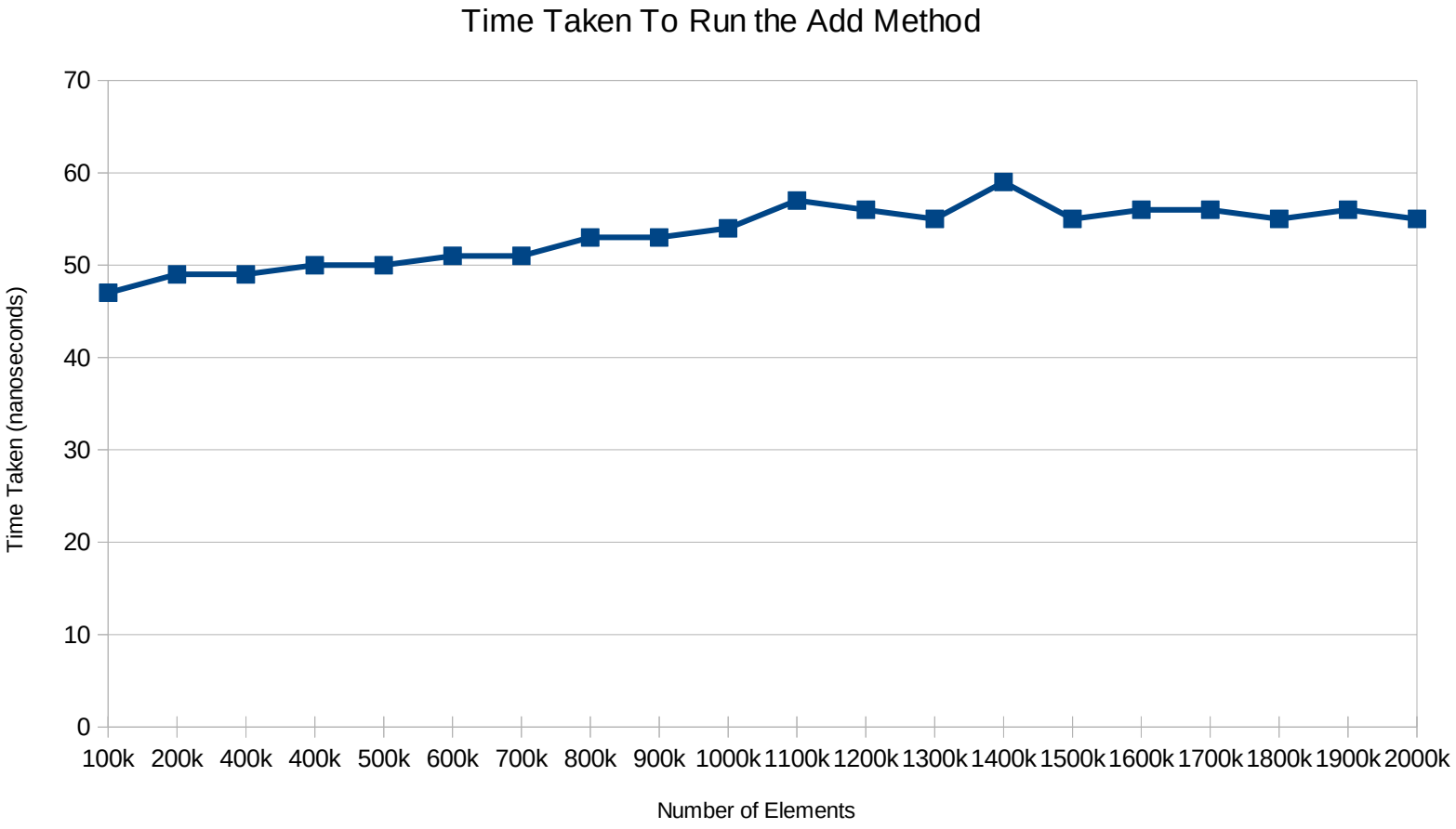
6. Plot the running time of MySortedSet's contains method for sets of sizes 100000 to 2000000 by steps of 100000. Use the timing techniques demonstrated in Lab 1. Be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Include your plot in your analysis document.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



Pictured above is a graph of the time it took to run the contains method (with a parameter of 100) for a MySortedList with the specified number of integer elements. It took roughly between 55 and 60 nanoseconds, and the time increased very slowly. I noticed there were some points where the time taken jumps a bit, but I think this might be due to my timing methods. Overall, I think it fits what I predicted, although starting from zero and going to a higher number of elements, as well as using a better timing method, would give me a better idea.

NOTE: My timing class was a modified version of the final example provided in Lab 1.

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)?



Pictured above is a plot of the times it took to add an element 1 larger than the largest element in the set of integers using the add method. It looks pretty much the same as the contains method, which makes sense considering they both use a binary search algorithm to locate the appropriate index.

Because it is dictated by a binary search algorithm, like the contains method, the add method has a big-oh behavior of O(log N).

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

About 6.

Programming partners are encouraged to collaborate on the answers to these questions. However, each partner must write and submit his/her own solutions.

Upload your document (.pdf only!) to the Assignment 3 page by 11:59pm on February 5.