Joshua Shipley
CS2420
Assignment 3 Analysis

1. Who is your programming partner? Which of you submitted the source code of your program?
    - My partner is Daniel, and he will be turning in the code for this assignment.

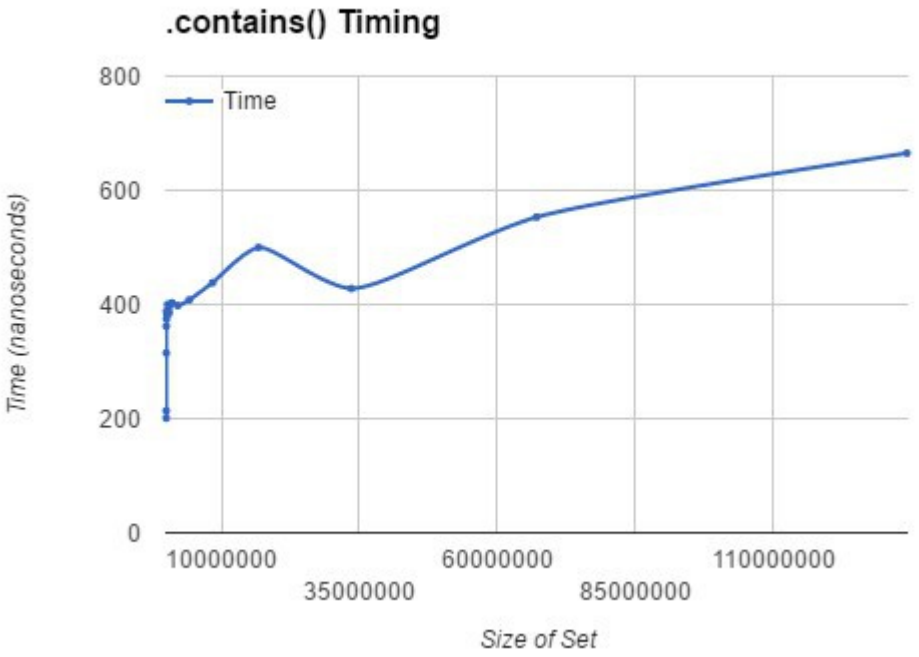
2. How often did you and your programming partner switch roles? Would you have preferred to switch less/more often? Why or why not?
    - We switched a few times, but for the most part when we were together he ended up being the one typing down most of it since the majority of the code was written down on his computer, which he was more used to than I was of course. I sort of liked how we did things simply because I feel like constantly switching out roles can confuse the situation rather than having a “designated driver”.

3. Evaluate your programming partner. Do you plan to work with this person again?
    - Daniel was a good partner to work with. He understands what's going on and is helpful when I need a hand, and when we're both stuck he does a good job with bouncing ideas back and forth. We do plan on working together again.

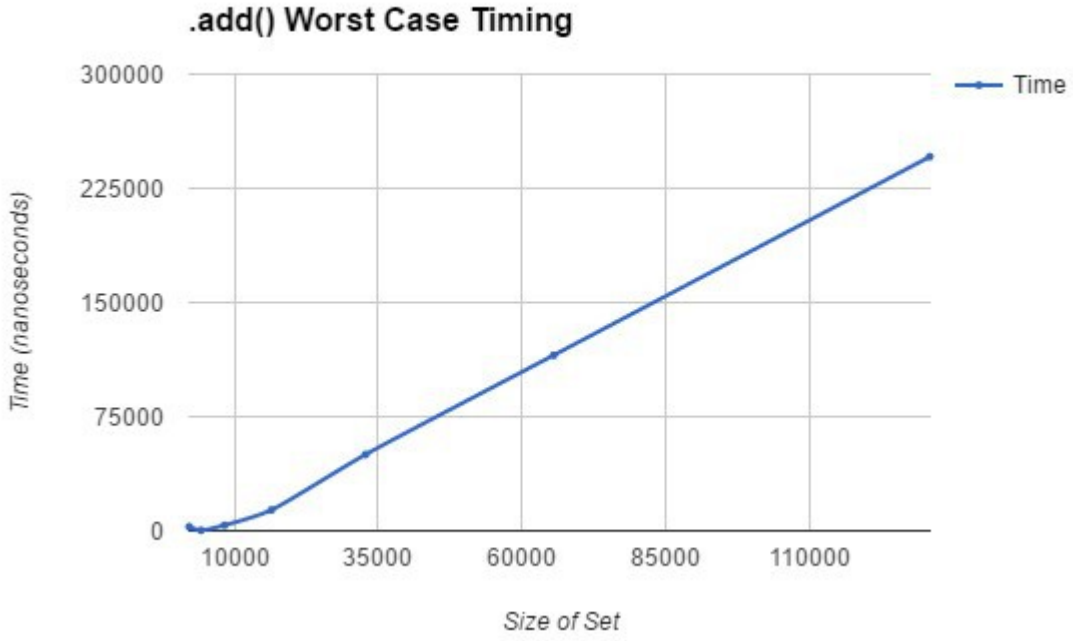
4. If you had backed the sorted set with a Java List instead of a basic array, summarize the main points in which your implementation would have differed. Do you expect that using a Java List would have more or less efficient and why? (Consider efficiency both in running time and in program development time.)
    - It would have been a lot more straight forward when it comes to resizing the list since that is pre-implemented under the hood. We could also just call Java's built in functions such as add, even though it might be less efficient if we were to just call add() then sort().

5. What do you expect the Big-O behavior of BinarySearchSet's contains method to be and why
    - We expected it to be log(N) since after every comparison it makes, if the wanted object wasn't found, it still cuts out half of the remaining set each time.

6. Plot the running time of BinarySearchSet's contains method, using the timing techniques demonstrated in Lab 2. Does the growth rate of these running times match the Big-oh behavior you predicted in question 5?
    - What we did was fill up a set of increasing size with consecutive Integers, we then found the average time of 10,000 timings dictating how long it took to look up a predetermined, randomly decided Integer. As shown by the graph below, even with the small dip partway through, the function as a whole takes on a fairly logarithmic timing path.



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. In the worst-case, how much time does it take to locate the position to add an element (give your answer using Big-oh)?



   - In the worst-case scenario, in which the element needs to be added to the front of the set, since every element that has already been added needs to move over, it is O(N). To test this case, we filled up a set with consecutive Integers of 1 to SetSize, then timed how long it took to add in a 0. As you can see in the above graph, the resulting average times

ended up being very linear.

8. How many hours did you spend on this assignment?
    - We ended up spending a total of about 11 – 12 hours working on the code for this assignment.