Assignment 8 Analysis

Brayden Wright
u0895942

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

My programming partner for this assignment was Abdulaziz Aljanahi (u0901606). I am the submitter of our program's source code for this assignment.

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

I think this partnership was fine and would not mind working with Aziz on the next assignment. It would be helpful to determine if we want to go that route before the Monday before the assignment is due though.

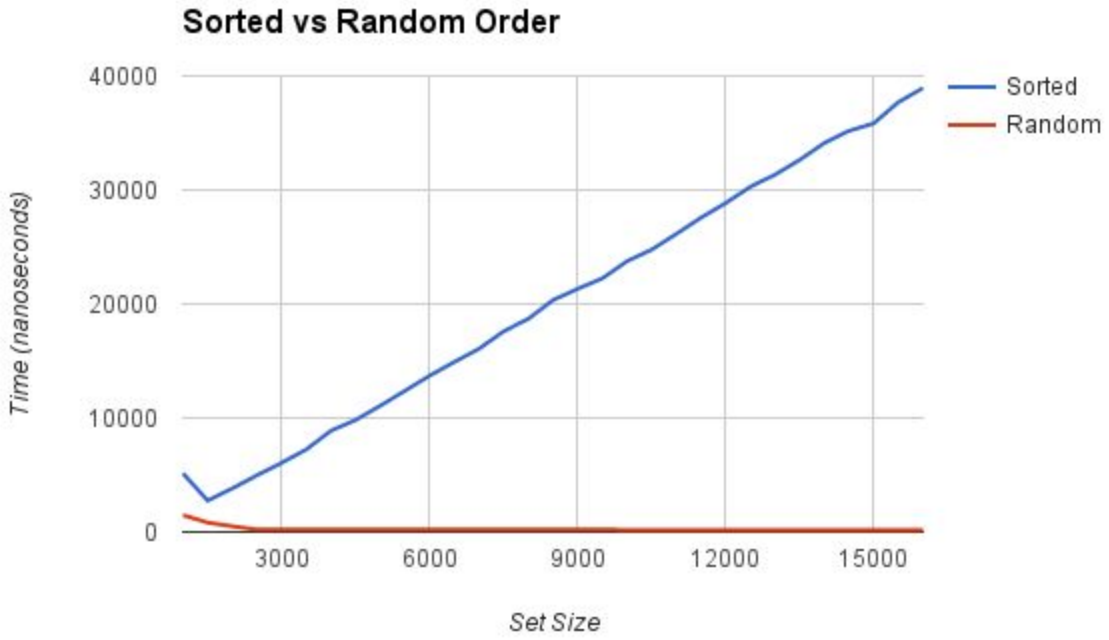
3. **How does building a Binary Search Tree by inserting in sorted order compare to building one by inserting in random order? Conduct a reasonable experiment, detail your steps, and plot your results.**

When building a Binary Search Tree by inserting item that are in sorted order, the Binary Search Tree become extremely lopsided. So much so that it essentially becomes a Linked List. If instead the Binary Search Tree is built by inserting in random order, there is a much greater chance of having something relatively close to a balanced Binary Search Tree structure. Note that there is no guarantee that the tree is actually balanced.

To conduct the timing tests, we used the suggested experiment as an outline. Now, we did our timing experiments separately, so the following numbers might differ a bit. Both variations started by first throwing a number of items (N) into an Array List, shuffling the items beforehand for the random order test. N started at 1,000 and incremented by 500 until the set size hit 16,000. I then tested checking for existing items within our Binary Search Tree in by shuffling the original Array List and checking if every item within the Array List is contained within our Binary Search Tree.

The graph produced from these timing experiments is on the next page (I was unable to get it to fit here without making it tiny and unreadable). Using a sorted order tree produced a linear graph, as was expected. In comparison, using a random order tree seemed to be almost constant time, which shouldn't happen. The running times actually grew shorted the larger the set

became for some reason. A random order tree should have produced a *log(n)* graph, which leads me to believe that there is some sort of issue with how this was timed.

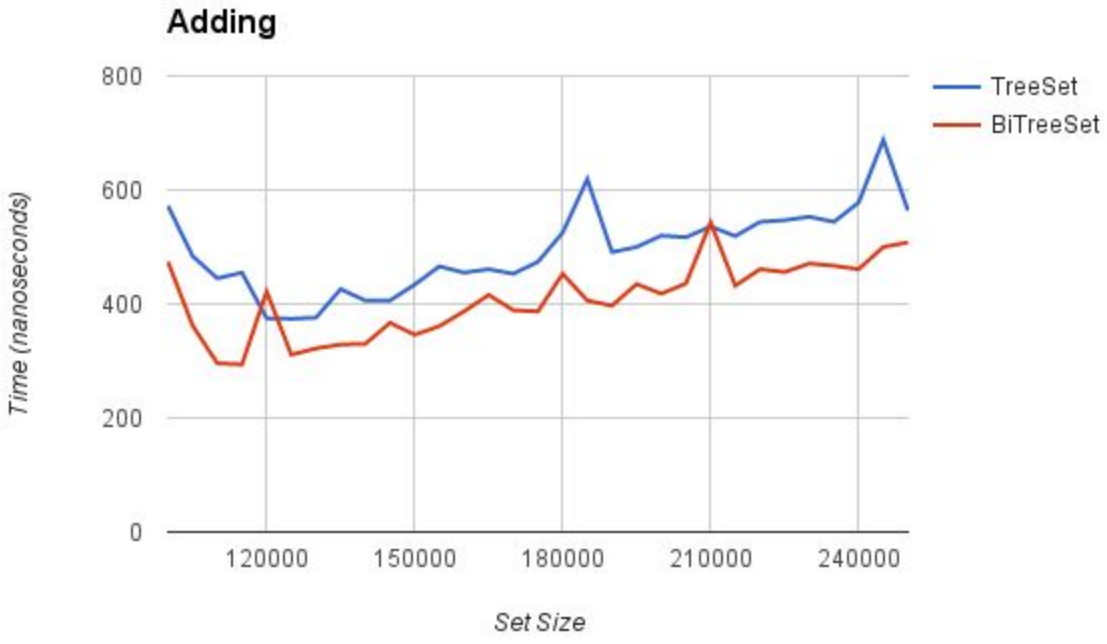


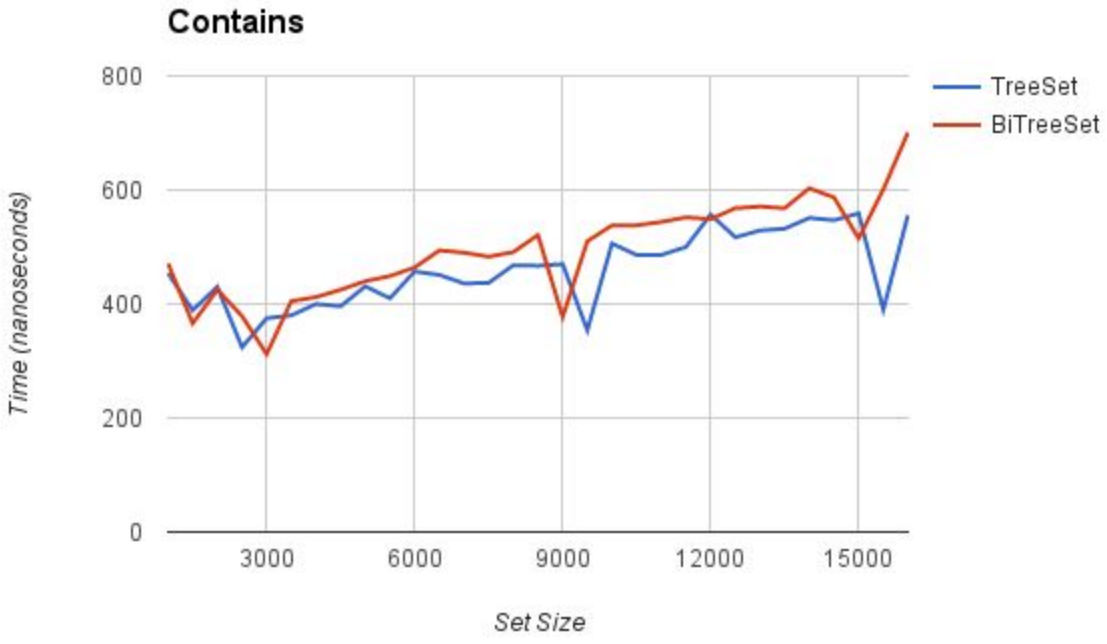

4. **Compare performance of a Binary Search Tree that requires balance to one which does not. Conduct a reasonable experiment, detail your steps, and plot your results.**

This experiment was performed in about the same manner as the experiment done in question 3. Both structures were timed adding N number of items in random order to them and averaged out. N goes from 100,000 to 250,000 in increments of 5,000 for these tests. The result of these timing tests are shown in the below graph titled *Adding*. The lines turned out as about what I was expecting; relatively slow growth with jaggedness due to randomness throwing off the average times. The *Contains* is a result of doing the timing in a similar manner to how adding was tested, described a couple sentences ago. Again, the results are about what I would expect, with the same reasoning behind its looks as described for the *Adding* graph.

Graphs are again on the next page because I could not get them to fit on this page without hurting their visual quality and making them fairly difficult to parse.

Brayden Wright
u0895942




5. **Many dictionaries are in alphabetical order. What problem will it create for a dictionary Binary Search Tree if it is constructed by inserting words in alphabetical order? Explain how this problem might be solved.**

I would probably go the easy route and just create a small symbol structure that stores the symbol, line, and column for any checked symbol and throw that structure into the stack instead of just the symbol. That way I would be able to just grab the line and column numbers of the unmatched opening symbol directly from the stack.

**6. How many hours did you spend on this assignment?**

I would say we spent somewhere around ten to fourteen hours in total on this assignment.