Assignment08 Analysis

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

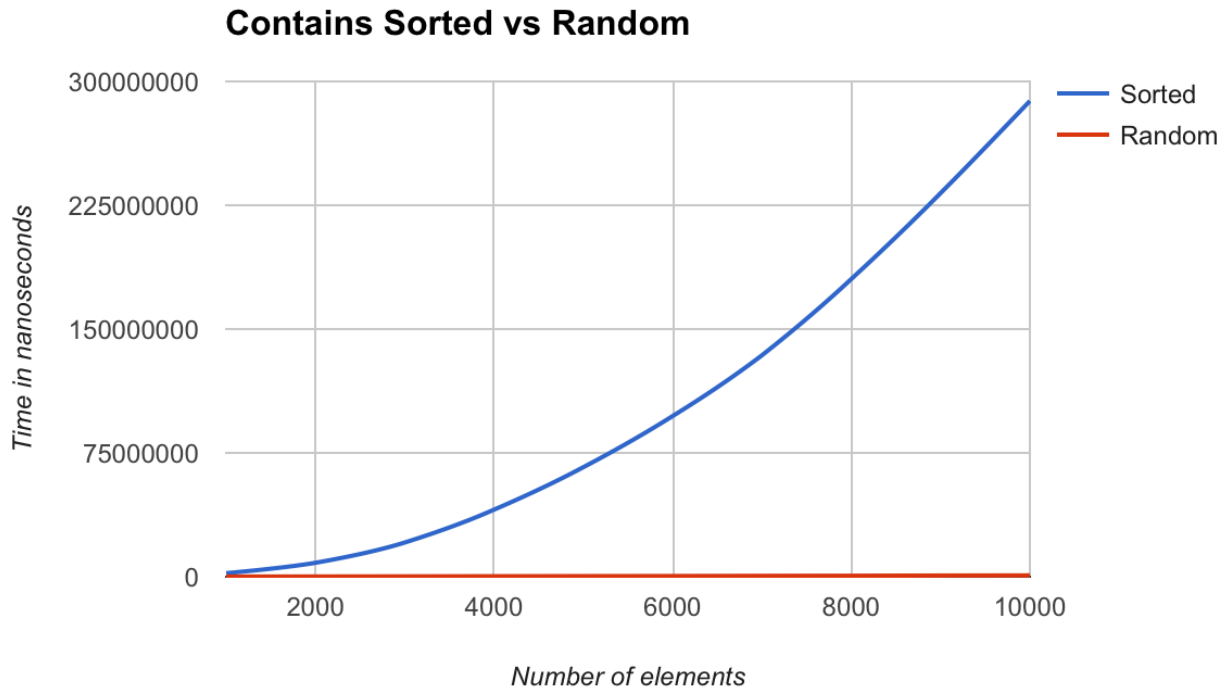
Andy Dao is my programming partner, he submitted the source code.

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

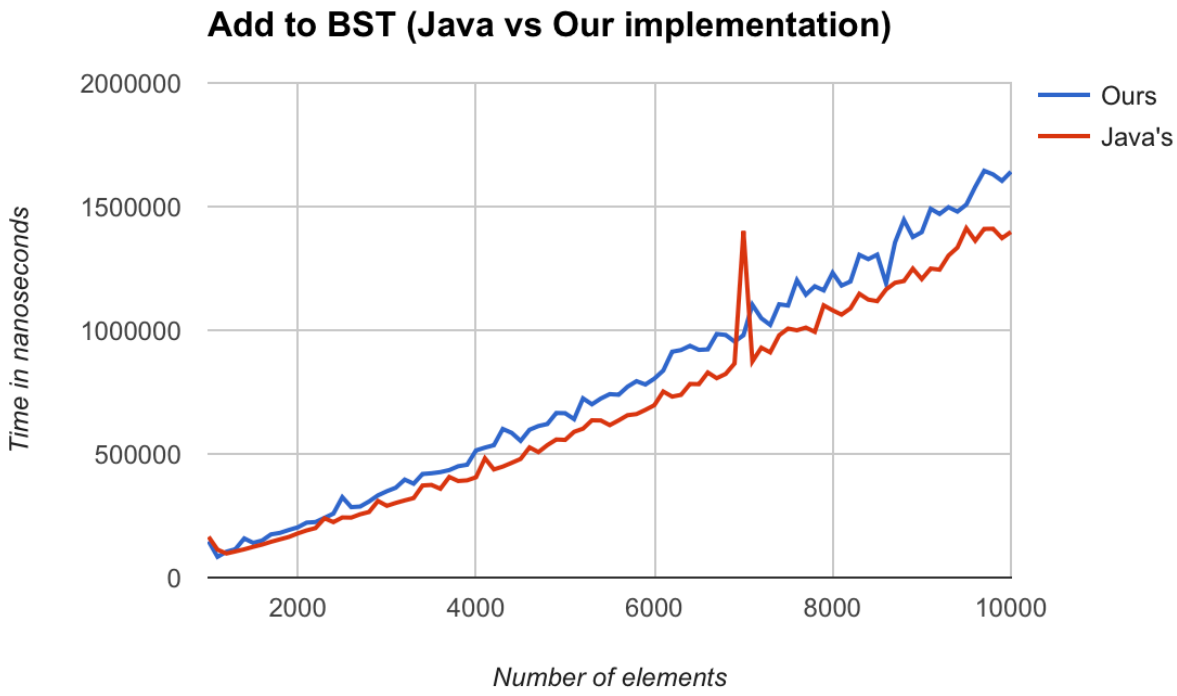
He is a great programming partner, I would definitely work with him again.

3. Design and conduct an experiment to illustrate the effect of building an N-item BST by inserting the N items in sorted order versus inserting the N items in a random order.

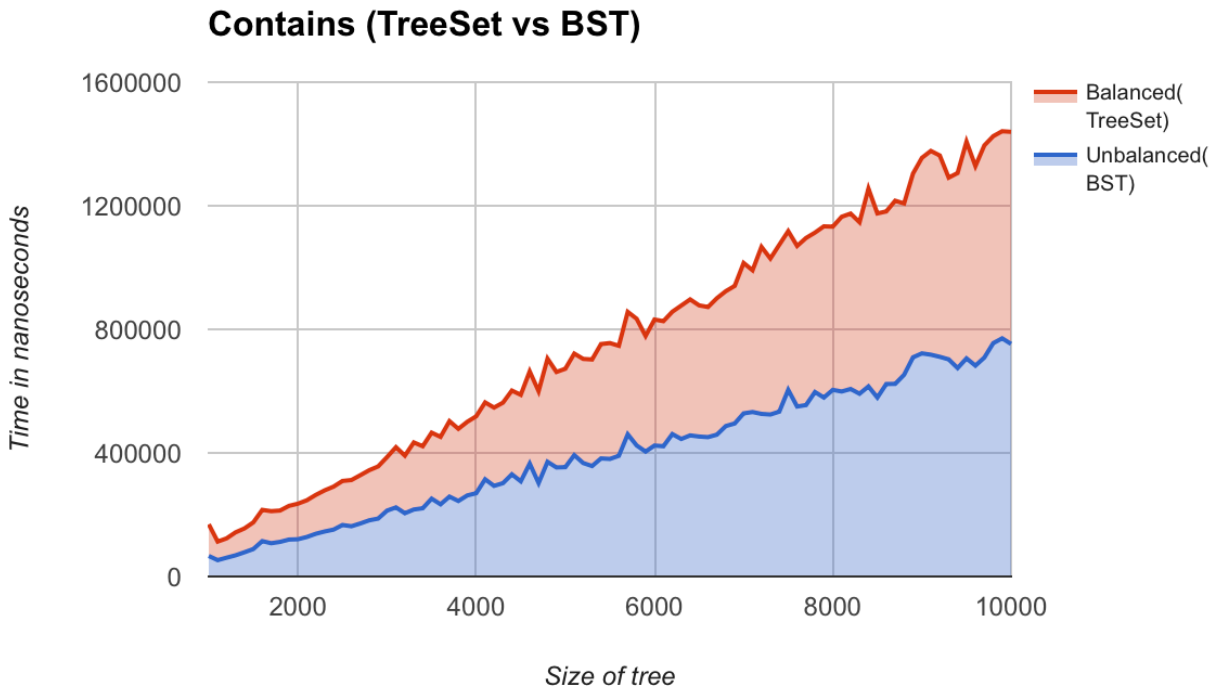
We added N items to a BST in sorted order, then recorded the time it took to traverse the tree and find the element at the end of the tree (the Nth element). This behavior was O(N) because it had to traverse every element in the tree, because a sorted tree is right heavy, effectively becoming a linked list. After, we added N items to a new BST, but in a random order, and then recorded the time it took to find the Nth element. This behavior was O(logn) because the tree was not sorted. This is apparent in the graph shown below as you can see the predicted behavior.



4. Design and conduct an experiment to illustrate the differing performance in a BST with a balance requirement and a BST that is allowed to be unbalanced.



For the first part of the timing experiment, we created an arraylist of integers from 0-N, randomized the arraylist then added those to Java's Tree list, and timed it. We then added that same arraylist to our BST and timed it. We repeated this step 90 times for both the TreeSet and BST, incrementing the tree size by 100. You can see the behavior in the graph above, reflecting that the performance time is almost identical, but Java's is slightly faster, probably due to the fact that the tree is balancing itself as it goes, making for a cleaner tree.



For the second part of the timing experiment, we then called the contains method on the trees that had been created, beginning the timer when we called the method and stopping the timer when the element was found, this was again repeated 90 times (1000-10000). It took our implementation (BST) slightly longer than Java's implementation (TreeSet) although both preformed with O(LogN) behavior. The reason for TreeSet preforming faster than ours is because Java's implementation of TreeSet balances itself as elements are added to the set, guaranteeing that behavior on a TreeSet will always be O(LogN).

5. Many dictionaries are in alphabetical order. What problem will it create for a dictionary BST if it is constructed by inserting words in alphabetical order? Explain what you could do to fix the problem.

To fix the problem, I would insert the word at the middle of the dictionary first. After, I would insert the word between the beginning of the dictionary and middle of the dictionary on the left side of the tree, and inserting the word between the middle of the dictionary and end of the dictionary on the left side of the tree.

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

I spent about 15 hours on this assignment.