Cameron Pickle

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

My partner is Gage Glenn. I will submit the code this week.

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

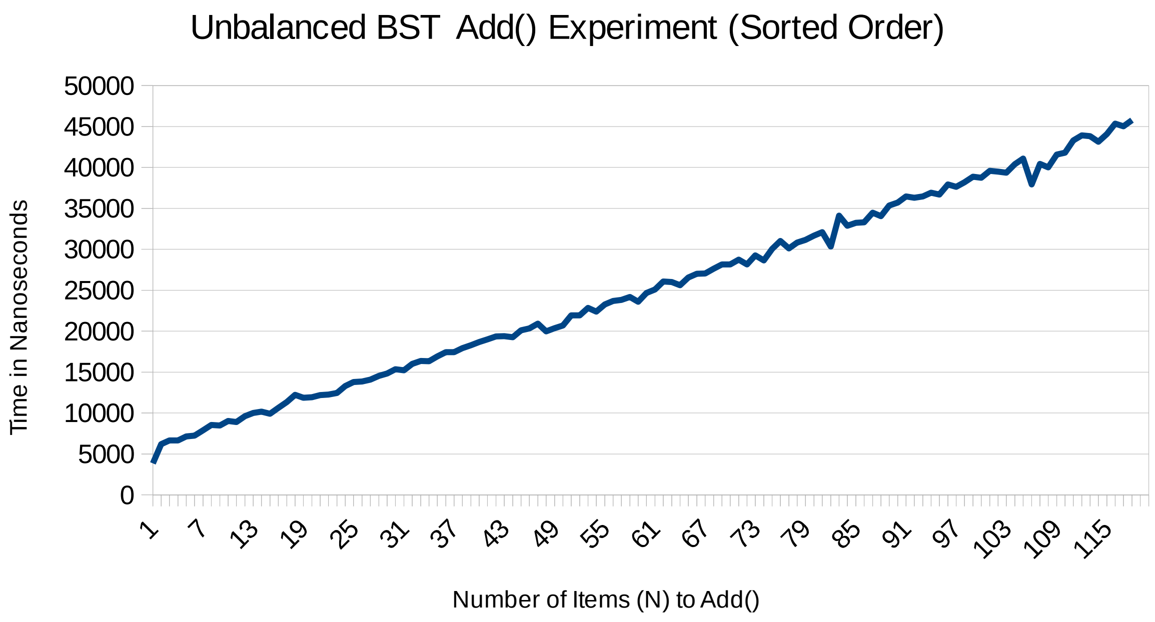
He is an excellent programmer. I have learned a lot with him during the time that we have been programming together.

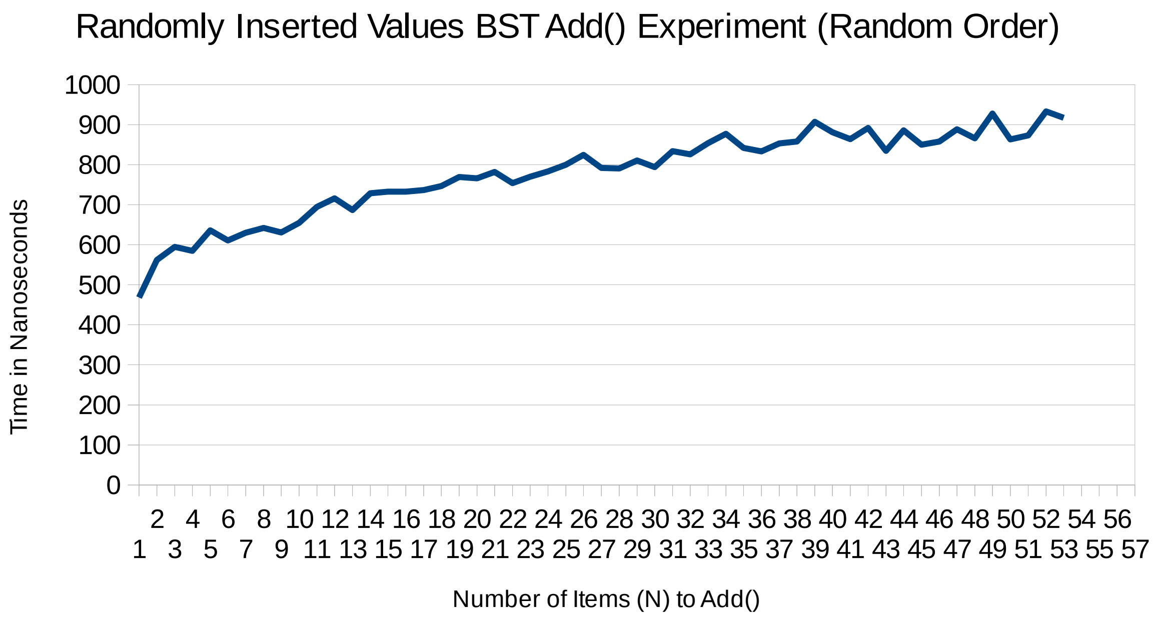
3. Design and conduct an experiment to illustrate the effect of building an N-item BST by inserting the N items in sorted order versus inserting the N items in a random order. Carefully describe your experiment, so that anyone reading this document could replicate your results. Submit any code required to conduct your experiment with the rest of your program and make sure that the code is well-commented. Plot the results of your experiment. Since the organization of your plot(s) is not specified here, the labels and titles of your plots(s), as well as, your interpretation of the plots is critical.
One suggestion for your experiments is:

- Add N items to a BST in sorted order, then record the time required to invoke the contains method for each item in the BST.

- Add the same N items to a new BST in a random order, then record the time required to invoke the contains method for each item in the new BST. (Due to the randomness of this step, you may want to perform it several times and record the average running time required.)

- Let one line of the plot be the running times found in #1 for each N in the range [1000, 10000] stepping by 100. (Feel free to change the range, as needed, to complement your machine.) Let the other line of the plot be the running times found in #2 for each N in the same range.





For the sorted order experiment we added values starting at 1000 that grew exponentially so that the number would always come after the preceding one causing a right heavy tree. In doing this we achieved our expected results of o(N). The graph clearly shows a linear growth.

For the Randomly inserted values experiment we used a java random object with a seed to make the results repeatable. We achieved once again the expected results of O(logN) as seen in the graph.

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. Use Java's TreeSet (http://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) as an example of the former and your BinarySearchTree as an example of the latter. Java's TreeSet is an implementation of a BST which automatically re-balances itself when necessary. Your BinarSearchTree class is not required to do this. Carefully describe your experiment, so that anyone reading this document could replicate your results. Submit any code required to conduct your experiment with the rest of your program and make sure that the code is well-commented. Plot the results of your experiment. Since the organization of your plot(s) is not specified here, the labels and titles of your plots(s), as well as, your interpretation of the plots is critical.
One suggestion for your experiments is:

- Add N items to a TreeSet in a random order and record the time required to do this.

- Record the time required to invoke the contains method for each item in the TreeSet.

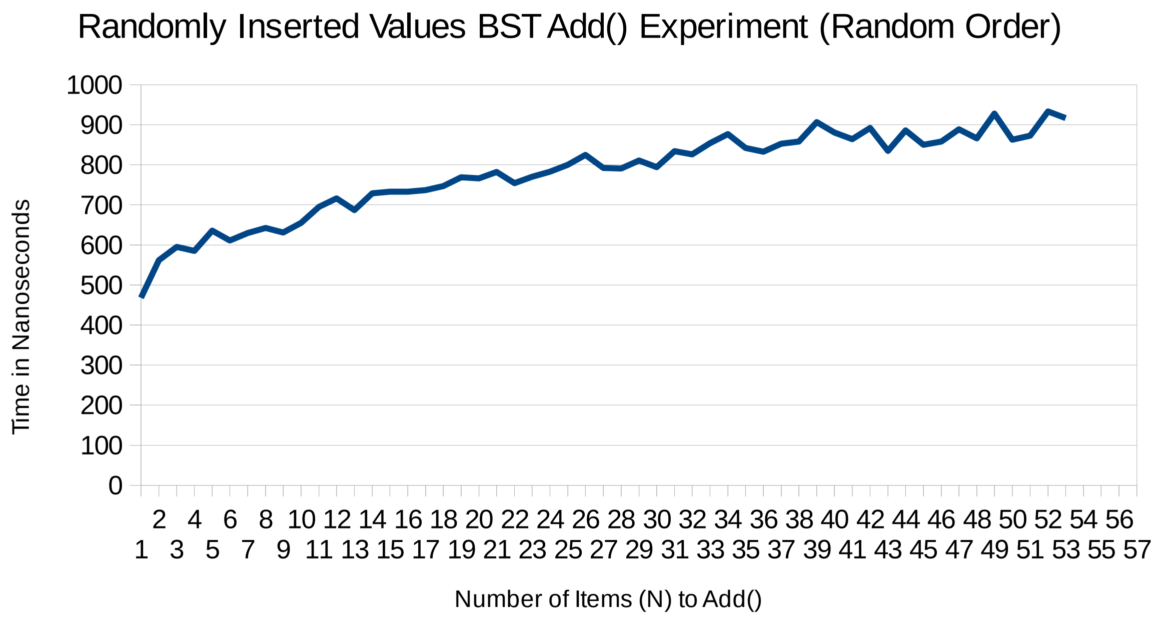
- Add the same N items (in the same random order) as in #1 to a BinarySearchTree and record the time required to do this.

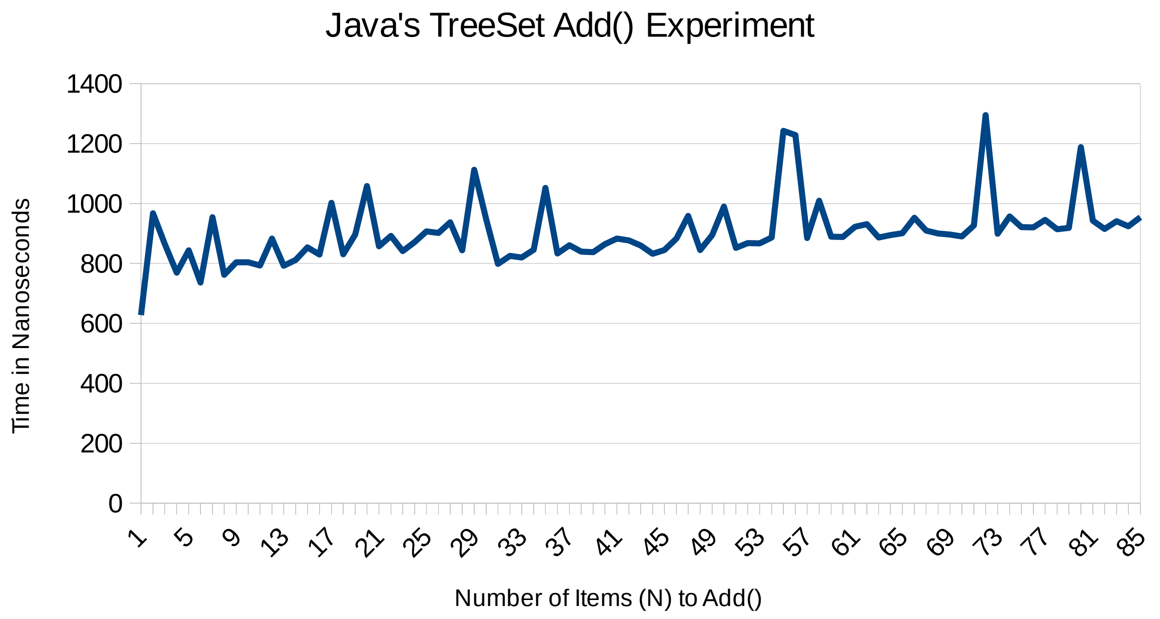
- Record the time required to invoke the contains method for each item in the BinarySearchTree.

- Let one line of the plot be the running times found in #1 for each N in the range [1000, 10000] stepping by 100. (Feel free to change the range, as needed, to complement your machine.) Let the other line of the plot be the running times found in the #3 for each N in the same range as above.

- Let one line of a new plot be the running times found in #2 for each N in the same range as above. Let the other line of plot be the running times found in #4 for each N in the same range.

(You can combine the plots in the last two steps, if the y axes are similar.)





For the Randomly inserted values experiment we used a java random object with a seed to make the results repeatable. We achieved once again the expected results of O(logN) as seen in the graph.
For the Java TreeSet we implemented Java's TreeSet and we got a very slow logarithmic growth. The spikes in the graph can be attributed to the redistribution of the graph when it becomes unbalanced. As you can see it is faster than our randomly sorted Add() and extremely faster than the ordered Add().

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.

It would make the tree become right heavy. This would make the search time become the worst case for the BST of O(N). In order to fix this problem you could take the list and find the middle value as the first and then find the middle value from the both partitions created and the middle value of those partitions etc. By doing this it would create a balanced tree.

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

We spent eight hours completing this assignment.