

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

**Mina Kim is my partner, and she will submit this assignment.**

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

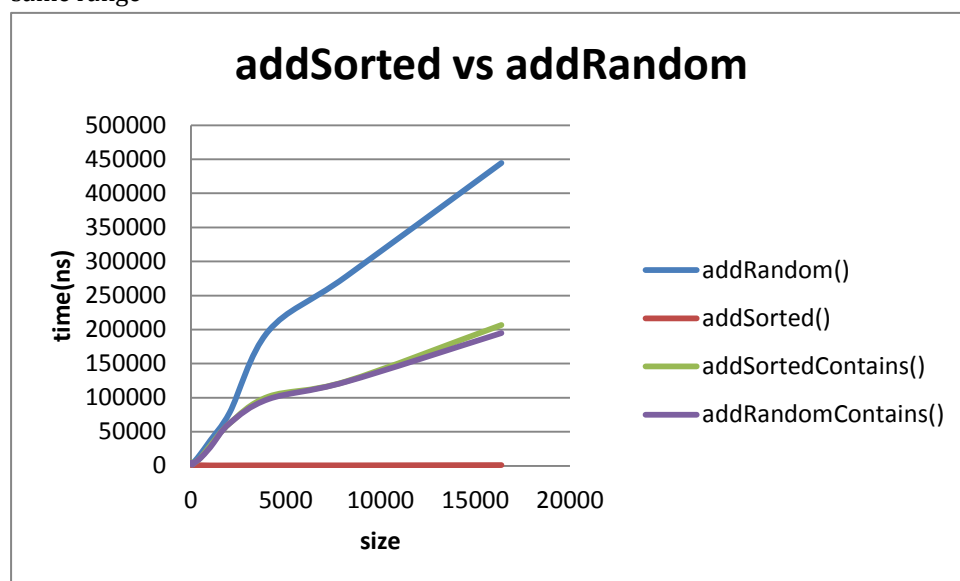
**My partner did great job I plan to work with her again.**

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. 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.)

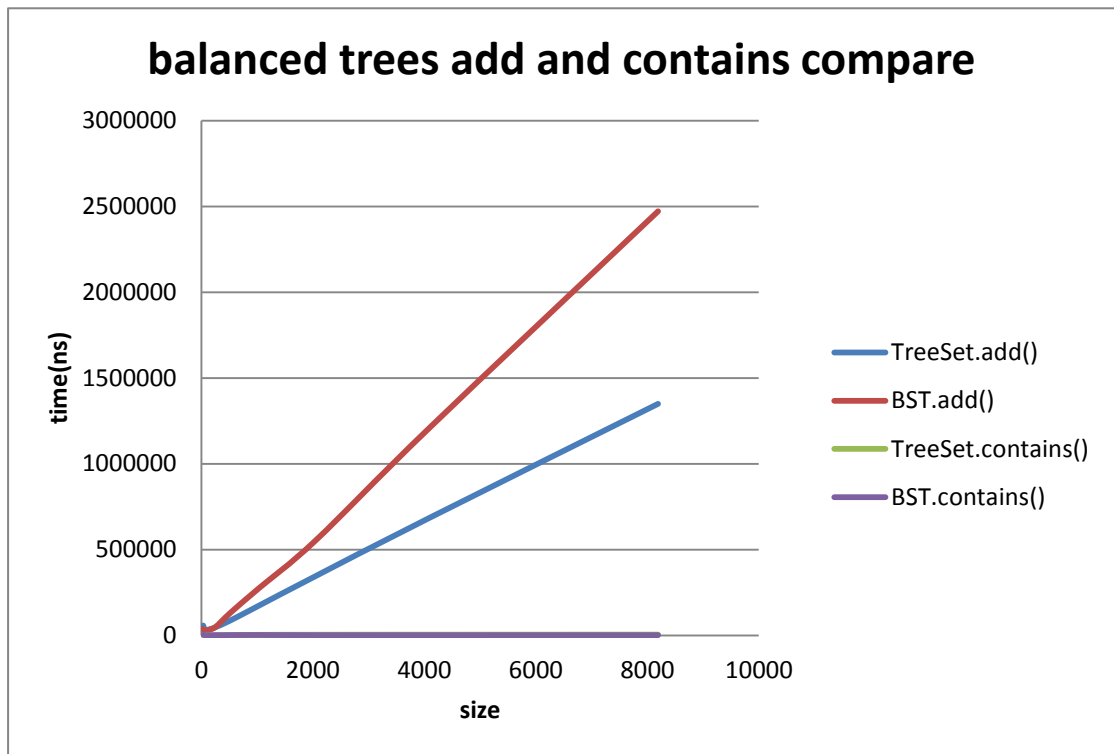
- Let one line of the plot be the running times found in #1 for each N in the range [1000, 10000] stepping by 100. (Feel free to change the range, as needed, to complement your machine.) Let the other line of the plot be the running times found in #2 for each N in the same range



Above plotting shows that add a sorted order items into the BinarySearchTree is  $O(1)$  because it will always keep add the element to the right side of the tree. And For a random order items, it will just show the average case behavior of the add which is

**$O(\log(N))$ . Otherwise, for both ways their `contains()` behave exactly same, which is  $O(\log(N))$ .**

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 `BinarySearchTree` 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.)



**This plotting shows that for a balanced tree, their complexity of add, contains methods are  $O(\log(N))$ .**

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.

**If I add words in alphabetical order, the tree would be unbalanced. Because it is a sorted order, so that it will always go to the right side of a node and its children's right. The best way to fix this is by adding them in a random order like shuffle the words first then insert to the BinarySearchTree.**

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

**8 hours**