

## Analysis

U0950512

YUHONG LIN

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

Yixiong Qin is my partner. I will submitte our program.

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

He is good. I' d like to 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. 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.

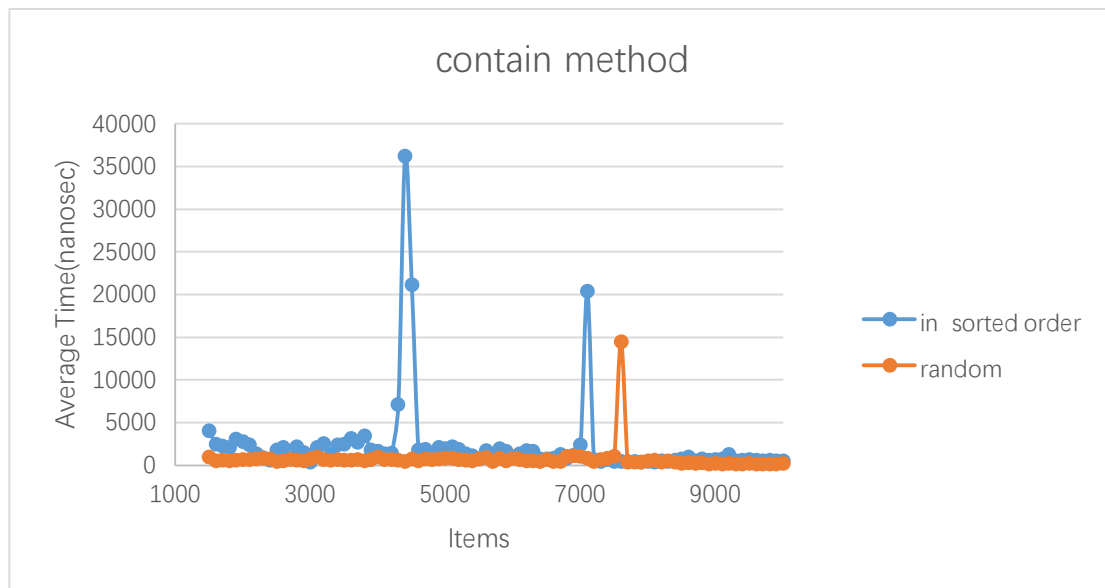
One suggestion for your experiments is:

- 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.



According to the graph, we can easily find out in small amount of item, use random order to insert can make contains() method more efficient, and in large amount of item, there is no big different between using sorted order and random order for contains() method.

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.

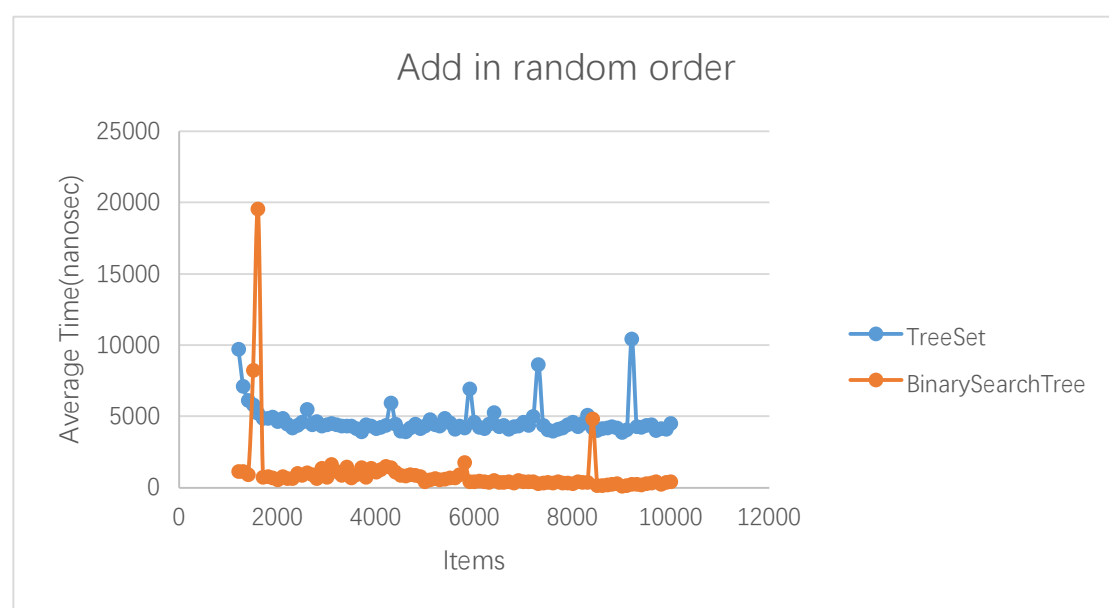
- Add the same N items (in the same random order) as in #1 to a BinarySearchTree and record the time required to do this.

- 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.

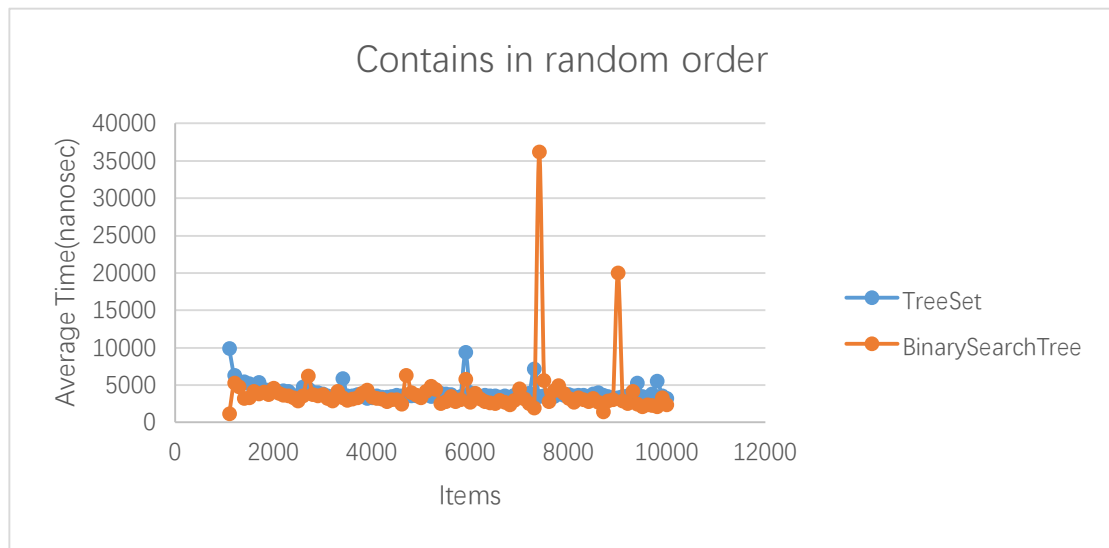
- Let one line of a new plot be the running times found in #2 for each N in the same range as above. Let the other line of plot be the running times found in #4 for each N in the same range.

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



According to the graph, TreeSet and BinarySearchTree both grow as  $O(1)$ . But I

used the same data to insert, and apparently BinarySearchTree use less time than TreeSet.



According to the graph, TreeSet and BinarySearchTree both go  $O(1)$ , and they use similar time to finish their job.

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 we insert words in alphabetical order, the tree will look like a line, because all the word we insert, always smaller than next inserted word. And it is easy to solve this problem, which is using random order to insert words.

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

About 5 hours.