Name： Pingchuan Ma
UID：u0805309

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

2. Evaluate your programming partner. Do you plan to work with this person again?
She is a hard worker with good schedule. She pays attention to the details sometimes I ignore. I plan to work with her again.

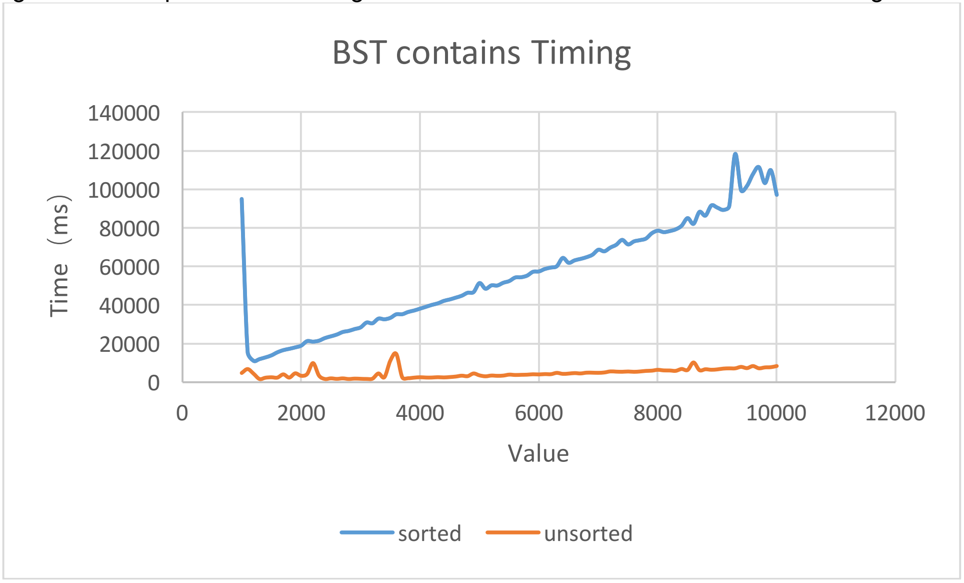
3. Design and conduct an experiment to illustrate the effect of building an N-item BST by inserting the N items in sorted order versus inserting the N items in a random order. Carefully describe your experiment, so that anyone reading this document could replicate your results. Submit any code required 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.

Building an N-item BST by inserting the N items in sorted order, the performance of contains method is O(N), the code runs very slowly. Because it is right heavy tress as a linked list, which performance for contains is O(N).
Building an N-item BST by inserting the N items in a random order, the tree will be much more balanced, as a balanced tree the performance of contains method is O(logN).
So for the contains method it is more efficient to build an N-item BST by inserting the N items in a random order.

Experiment is:
- Add N items(1-10000) to a BST in sorted order, then record the time required to invoke the contains method for the biggest item in the BST.

- Add N items(1-10000) to a new BST in a random order(shuffle the collection of the items in #1), then record the time required to invoke the contains method for the biggest item in the new BST. (Due to the randomness of this step, I perform it 10 times and record the average running time required.)

- Blue line of the plot is the running times found in #1 for each N in the range [1000, 10000] stepping by 100. Orange line of the plot is the running times found in #2 for each N in the same range.



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.

For the add method, BST and TreeSet both have O(logN) performance. But as N grows bigger BST cost more time than TreeSet. It means each time inserting an item to a much more balanced tree (TreeSet) is much more efficient, as the performance of a balanced tree is O(logN).
For the contain method, BST and TreeSet both have O(logN) performance. But as N grows bigger BST cost more time than TreeSet. It means each time search an item in a much more balanced tree (TreeSet) is much more efficient, as the performance of a balanced tree is O(logN).

The experiments is:

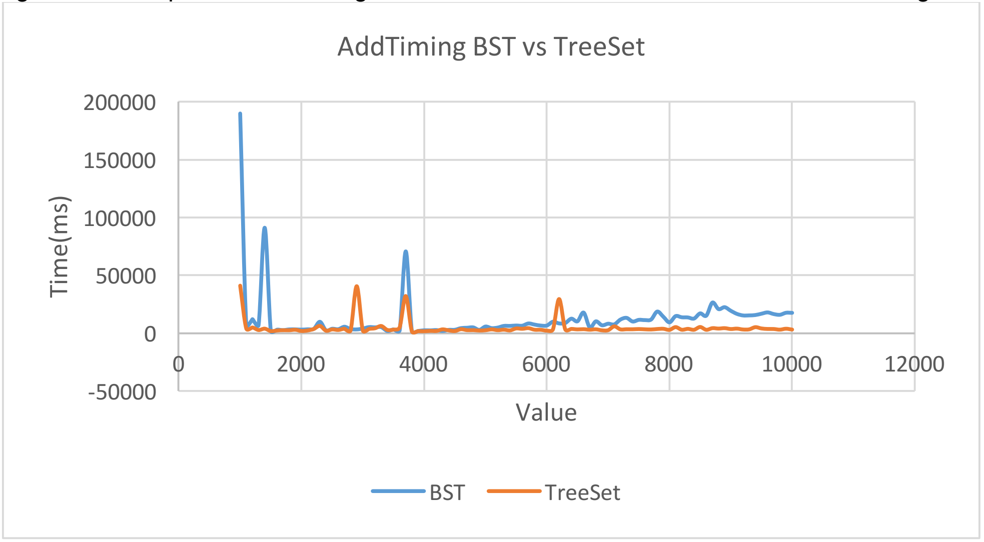
- Add 10000 items(1-10000) to a TreeSet in a random order and record the time required to add the item N+1.

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

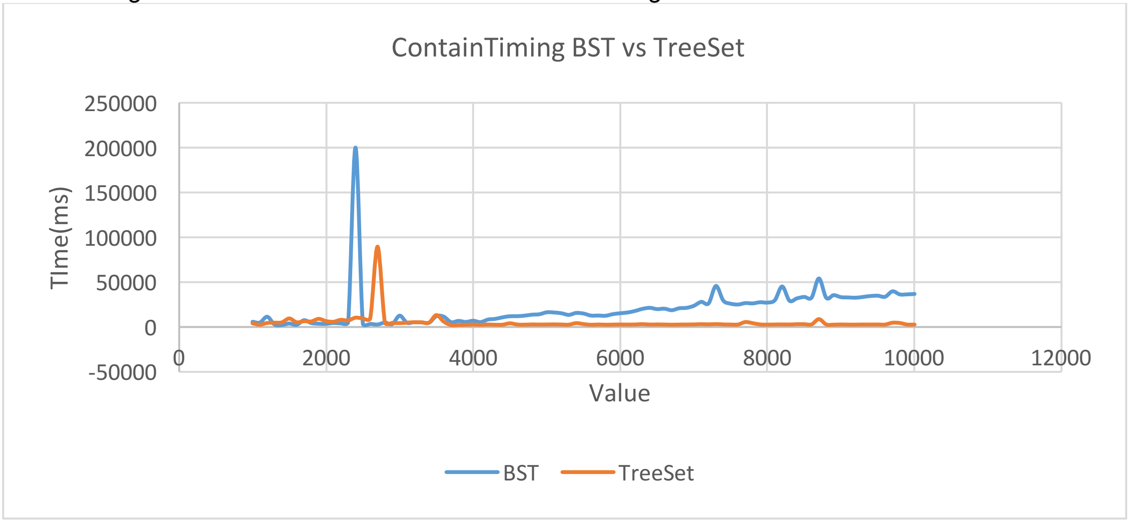
- Add 10000 items(1-10000) (in the same random order) as in #1 to a BinarySearchTree and record the time required to add the item N+1 .

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

- Blue line of the plot is the running times found in #1 for each N in the range [1000, 10000] stepping by 100. Orange line of the plot is the running times found in the #3 for each N in the same range as above.



- Blue line below is the running times found in #2 for each N in the same range as above. Orange line of the plot is the running times found in #4 for each N in the same range.



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.

When the words are in alphabetical order, the tree will be very unbalanced and come into a linked list with n nodes. The difference in performance between the balanced and unbalanced tree may be enormous as I discussed above. The add and contain will be O(N) for an balanced tree, while O(logN) for a balanced one. To solve this problem, each time we can choose the middle of the words to insert. For example, first we choose the middle one to insert it as root, so the list is separated to two sides. Then choose the middle from the left side to insert and it will be root's left child, then choose the middle from the left side to insert as root's right child. Do recursion like this until each of the words inserted.

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

18 hours