Name: Yixiong Qin

Uid: u0900071

CS2420

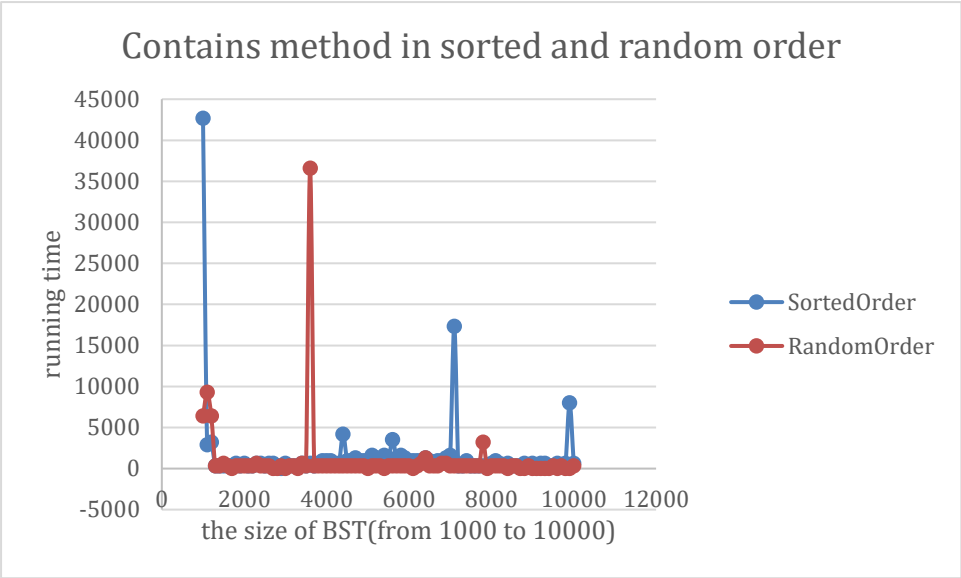
Analysis for Assignment 8

1. My partner is Yuhong Lin, and he is going to submit this assignment.

2. He is really good as my duo. We worked together to draw some pictures in order to get a deeper understanding for the process of code working in Binary Search Tree. And I'm planning to work with him again.

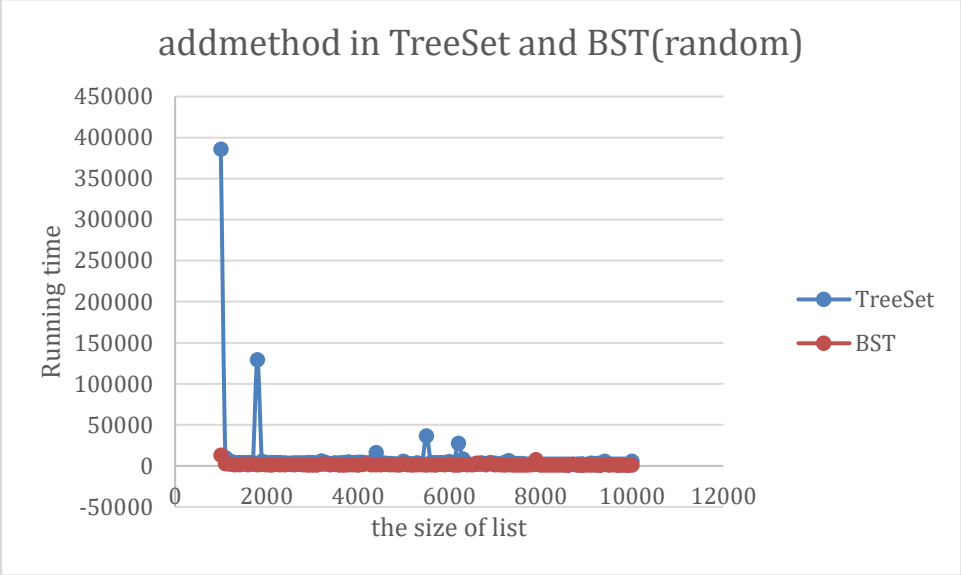
3. For this experiment, I made the size of BinarySearchTree to change from 1000 to 10000, and then add elements by the size of the tree. Firstly, I add 1000 to 10000 Integer elements in ascending order and set up the testing time for it. It will just run constantly if the run time less than the testing time. The last step is just calculate average time by expression: averageTime = ((midpointTime - startTime) - (stopTime - midpointTime))/ timesToLoop; Actually we used a 'for' loop to make BinarySearchTree checking whether

the current tree contains every element.

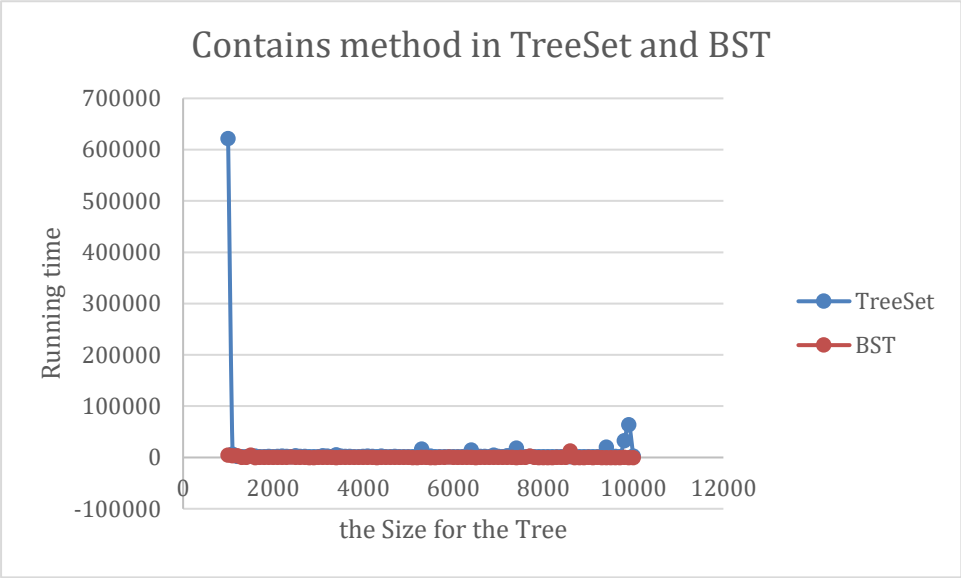


The picture is used to express the behavior for contains method of BinarySearchTree in different situations of SortedOrder and RandomOrder. At the SortedOrder, the Big-O complexity is O (N) for contains method. It needs to compare with every element in the current tree until the item is founded. By comparison, we can find that invoking contains method in RandomOrder is a bit faster than in SortedOrder. That's because it's possible for us to pick a better random element to be the first in a BinarySearchTree, and then it will make the tree looked more balance instead of a tree just has one side. So it will help us to save some time when we tried to invoke contains method. And It actually just needs check half nodes.

4. For this experiment, it is about the add method test. The usage of ArrayList is to store the random numbers, we used a ‘while’ loop for the operation to avoid duplication, and then I made separately operation to add elements in ArrayList into TreeSet and BinarySearchTree to test their running time, and below is the graph. Regarding to contains method testing, I made the add operation first, and then test the contains method.



The complexity of Add method for both two kinds of tree is O (logN). It needs to compare with the data in root node and then to decide which direction should go for inserting instead of compare with all elements that exist in the tree when we wanted to insert a new element. At the beginning to add elements to tree, the first element is really important for a unbalanced tree.



Tis graph is the comparison of contains method in TreeSet and BinarySearchTree. The complexity Big-O is O (logN) for both of these two classes. And it seems that they are almost same tendency in the graph. I guess it mainly because the random order . It's also possible to make BinarySearchTree to be balanced if we choose a really good start element.

5. The complexity of Big-O would be O (N) if we wanted to insert the words in a dictionary to BST. The words are in sorted by alphabetical order in a dictionary. So it is going to conduct the BST looks like a line that every thing in right child of node. A way to solve this problem is to usie a list to insert words in dictionary by a random order, and then invoke addAll method in BST to insert elements.

6. I almost spent 6 hours on this assignment.