Project 1

CS 241 – 02

Minwoo Soh

[msoh@cpp.edu](mailto:msoh@cpp.edu)

Tuesday, January 30, 2018

Section 1. Project Description:

Your program should read from the standard input a sequence of integer values, with each value separated by a space. Your task is to:

• Build a binary search tree using these values in the order they are entered. • Print 3 traversals: pre-, in-, and post-order.

• Allow the user to insert/delete a value. Once a new tree is generated, print it in-order. • Find predecessor of a given value. The predecessor is the node that appears right before the given value in an in-order traversal.

• Find successor of a given value. The successor is the node that appears right after the given value in an in-order traversal.

In your BST implementation, the add and delete methods must be implemented using recursion. You will lose major points for using a non-recursive implementation. Note that no duplicates are allowed in this BST.

Section 2. Project Specification:

The project required us to create a Binary Search Tree with most of the default methods and few additional functions. Default methods included basic functions such as having a method of preorder, in-order, and post-order traversal. Additional functions were primarily getting the predecessor and successor of a specified node as mentioned in the task description. Furthermore, functions for finding a node and finding its parents were also added to make other methods, such as remove, getPredecessor, and getSuccessor, much easier to implement. A main program, named Project1, was made for interactive interface, and a BinaryNode class was used in conjunction with the BinarySearchTree class.

Section 3. Testing Methodology:

Testing to see if the code worked correctly came down to trying out many different tree cases. Tree variety ranged from full trees, balanced trees, unbalanced trees, trees with children only on one side, single node trees, small trees with 2 or 3 nodes only, empty trees, and more. Different tree structures were used in the main program and tested with all Binary Search Tree methods to determine if there were any errors within the in-order list output. Errors were then analyzed along with the resulting tree to determine the correct solution. These scenarios were primarily used to test methods such as remove, getPredecessor, and getSuccessor, which required separate implementations for each of their numerous cases.

Section 4. Lesson Learned:

Encountering abundant amount of errors during testing helped me realize the additional cases that the methods can encounter. For example, nodes were not getting deleted whenever the delete command was made, and the tree stayed exactly same as before. The remove method continued to cause errors despite implementing the 4 cases of removing a node as mentioned in the lecture. I later understood that the parent node of the specified node had to rearrange their references to point at the node’s child or null to make sure the node was dereferenced and therefore effectively removed from the tree.

Section 5. Analysis of Output:

After fixing all errors and making sure the program did not crash on any occasion, every input and commands ran correctly. The program was tested with the provided data set in the project description, and the output and format were perfectly identical to its own results.