CSCI 2270 Data Structures Recitation 8
Instructors: Hoenigman/Zagrodzki/Zietz

Deleting a Node in BST

Learning Objectives:

- Delete a node in a BST

In this recitation, you will be learning how to delete a node in binary search trees. When a node is deleted from the tree, the node may need to be replaced with another node in the tree. The replacement node needs to be selected such that the BST properties are preserved. There are three cases to consider when deleting a node. Exactly one of the following conditions is true about the deleted node:

1. The node has no children.
2. The node has one child.
3. The node has two children.

The tree below shows a BST with examples of nodes with 0, 1, or 2 children. The nodes with values of 3, 9, 17, and 20 have no children. The node with a value of 7 has one child, and the nodes with values of 6, 15, and 18 have two children. The delete() algorithm to handle all three cases is below. For brevity, only the case where the deleted node is its parent's left child is shown.

```
            15
          /    \
       6          18
    /    \      /    \
  3        7   17    20
             \
              9
```

Algorithm delete(value)
Deletes the node where the value matches the node key value.

Pre-conditions
value is a valid search value whose type matches the node key type. search() algorithm exists to identify the node to delete. treeMinimum() algorithm exists to identify the minimum value in a sub-tree, which will be the replacement node for a deleted node with two children.

Post-conditions
Node with specified key value is deleted from the tree. parent, left child, and right child pointers for the deleted node and neighboring nodes are reset accordingly.

Algorithm
(Note: this is not the complete delete() algorithm. For the one- and two-children cases, only the case where the deleted node is the left child of its parent is shown. Additional cases are needed to handle when the deleted node is the right child.)

```
delete(value)
1. node = search(value)
2. if(node != root)
3.     if(node.leftChild == NULL and node.rightChild == NULL) //no children
4.           node.parent.leftChild = NULL
5.     else if(node.leftChild != NULL and node.rightChild != NULL) //two children
6.           min = treeMinimum(node.rightChild)
7.           if (min == node.rightChild)
8.                 node.parent.leftChild = min
10.                else
11.                    min.parent.leftChild = min.rightChild
12.                    min.parent = node.parent
13.                    min.right.parent = min.parent
14.                    node.parent.leftChild = min
15.                    min.leftChild = node.leftChild
16.                    min.rightChild = node.rightChild
17.                    node.rightChild.parent = min
18.                    node.leftChild.parent = min
19.          else //one child
20.              x = node.leftChild
21.              node.parent.leftChild = x
22.              x.parent = node.parent
23. else
24.     //repeat cases of 0, 1, or 2 children
25.     //replacement node is the new root
26.     //parent of replacement is NULL
```

```
27. delete node
```

Recitation Assignment:

In this recitation, you are not required to write the code for deleting a node in a BST; instead, you need to write pseudo-code on a piece of paper and hand it to your TA for the following questions.

Consider the BST in the figure above. Try to write basic procedures for deleting the three nodes:

- **Node has no children: delete(3)**
- **Node has one child: delete(7)**
- **delete(6)**

We take attendance for this lab's credit.