CSCI 2270 - Data structures and algorithms
Instructor: Hoenigman
Assignment 9
Due March 30 by 3pm

Red-black trees

In this assignment, you are asked to manipulate a red-black tree by adding and removing nodes from the tree. There are a few questions on moodle on red black trees that you need to complete, similar to the question you completed for this week's recitation.

In addition to the moodle questions, answer the following two questions and upload your answers to the Assignment 9 link on moodle. Your answers should provide a specific example to support your reasoning.

Question 1: Does inserting a node into a red-black tree, re-balancing, and then deleting it result in the original tree?

Question 2: Does deleting a node with no children from a red-black tree, re-balancing, and then re-inserting it with the same key always result in the original tree?

Red-black algorithms for insert and delete

For your reference, the insert and delete algorithms are provided here.

Insert algorithm

```
redBlackInsert(value){
        x = insert(value) //add a node to the tree as a red node
        while(x != root and x.parent.color == red){
                If(parent == x.parent.parent.left){
                        uncle = x.parent.parent.right
                        if(uncle.color == red){
                                x.parent.color = black
                                uncle.color = black
                                x.parent.parent.color = red
                                x = x.parent.parent
                        }else{
                                if(x == x.parent.right){
                                        x = x.parent
                                        leftRotate(x)
                                }
                                x.parent.color = black
                                x.parent.parent.color = red
                                rightRotate(x.parent.parent)
                        }
                }else{
```

```
                    //x.parent is a right child. Swap left and right for algorithm
                }
        }
        root.color = black
}
```

Delete algorithm

```
redBlackDelete(value){
        node = search(value)
        nodeColor = node.color
        if(node != root){
                if(node.leftChild == nullNode and node.rightChild == nullNode){ //no children
                        node.parent.leftChild = nullNode
                        x = node.leftChild
                }else if(node.leftChild != nullNode and node.rightChild != nullNode){ //two children
                        min = treeMinimum(node.rightChild)
                        nodeColor = min.color //color of replacement
                        x = min.rightChild
                        if (min == node.rightChild){
                                node.parent.leftChild = min
                                min.parent = node.parent
                                min.leftChild = node.leftChild
                                min.leftChild.parent = min
                        }else{
                           min.parent.leftChild = min.rightChild
                           min.rightChild.parent = min.parent
                           min.parent = node.parent
                           node.parent.leftChild = min
                           min.leftChild = node.leftChild
                           min.rightChild = node.rightChild
                           node.rightChild.parent = min
                           node.leftChild.parent = min
                        }
                        min.color = node.color //replacement gets nodes color
                }else{ //one child
                         x = node.leftChild
                         node.parent.leftChild = x
                         x.parent = node.parent
        }else{
                  //repeat cases of 0, 1, or 2 children
                  //replacement node is the new root
                  //parent of replacement is nullNode
        }
        if (nodeColor == BLACK){
```

```
            RBBalance(x)
        }
        delete node
}
```

Red-black rebalancing after delete

```
RBBalance(x){
        while (x != root and x.color == BLACK){
                if (x == x.parent.leftChild){
                        s = x.parent.rightChild
                        if (s.color == RED){  //Case 1
                                s.color = BLACK
                                x.parent.color = RED
                                leftRotate(x.parent)
                                s = x.parent.rightChild
                        }
                        if (s.leftChild.color == BLACK and s.rightChild.color == BLACK){ //Case 2
                                s.color = RED
                                x = x.parent
                        }else if(s.leftChild.color == RED and s.rightChild.color == BLACK){ //Case 3
                                s.leftChild.color = BLACK
                                s.color = RED
                                rightRotate(s)
                                s = x.parent.rightChild
                        }else{
                                s.color = x.parent.color  //Case 4
                                x.parent.color = BLACK
                                s.rightChild.color = BLACK
                                leftRotate(x.parent)
                                x = root
                        }
                }else{
                        //x is a right child
                        //exchange left and right
                }
        }
        x.color = BLACK
}
```