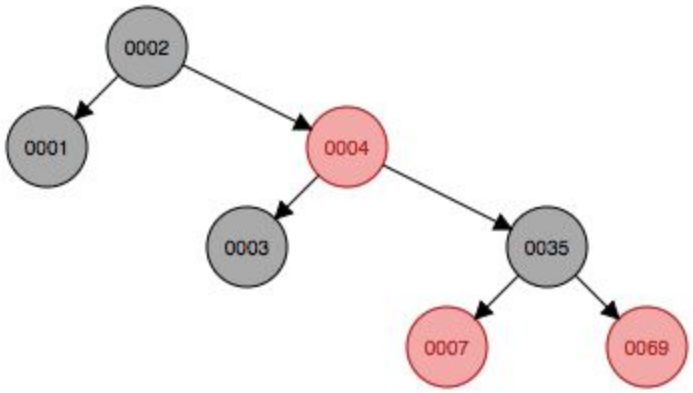
Malcolm McKellips
Data Structures
Zagrodzki
Lecture 300
Recitation 301

Assignment 7

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

Answer 1: No, inserting a node into a red-black tree, re-balancing, and then deleting the node does not necessarily result in the original tree.

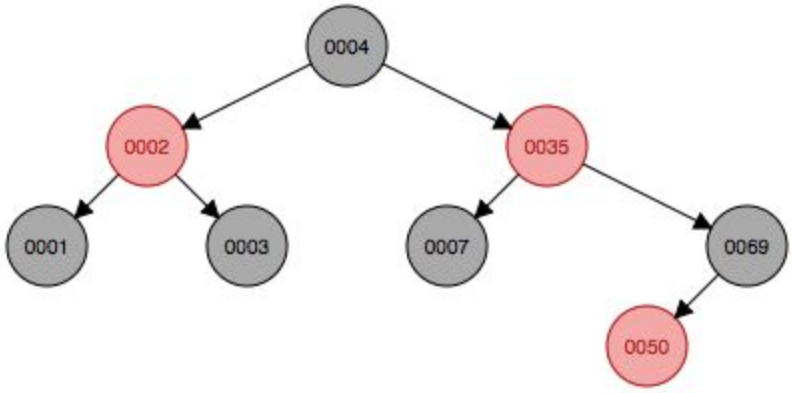
E.g.
Original Tree



Insert 50
Algorithm
-Insert 50 as if it was a normal BST. 50 Is greater than 2 so progress to the right subtree of 2. It is also greater than 4 so progress to the right subtree of 4. It is also greater than 35 so progress to the right subtree of 35. It is less than 69 and 69 points to only null leaves, so add 50 as the red left child of 69.
-50's parent 69 is the right child of 35 so the uncle of 50 is the left child of 35 which is 7
-The uncle node 7 is red, so recolor the parent node 69 and the uncle node 7 to black
-Also since the uncle node 7 was red, recolor x's grandparent 35 to red.
-Now the new node x that we will be analyzing is 50''s grandparent 35
-35's parent is red, so the tree is still not balanced and we will have to go through the rebalancing loop again
-35's parent 4 is the right child of its parent 2, so the uncle node is the left child of 2, which is 1
-This uncle is not red and 35 is the right child of 4, so x is now 4 and we rotate left around 4, and the tree appears almost balanced.

-Finally we recolor 4 to black and as 4 is now the root, the balancing is complete and we do not have to go through the rebalancing loop again.

Rebalanced Tree

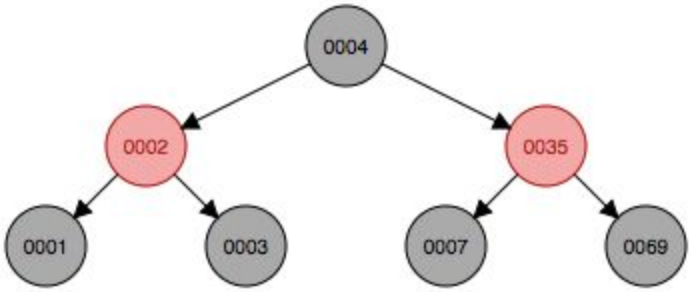


Delete 50

Algorithm

-Search for the node with the value 50 and store this as the node to be deleted.
-Store the color of 50 as red
-The node is not the root and it has no children, so set the left child of the parent of 50 to a null node
-As the node to be deleted was a red node, the tree needs no further balancing
-Delete the memory space for 50

Final Tree



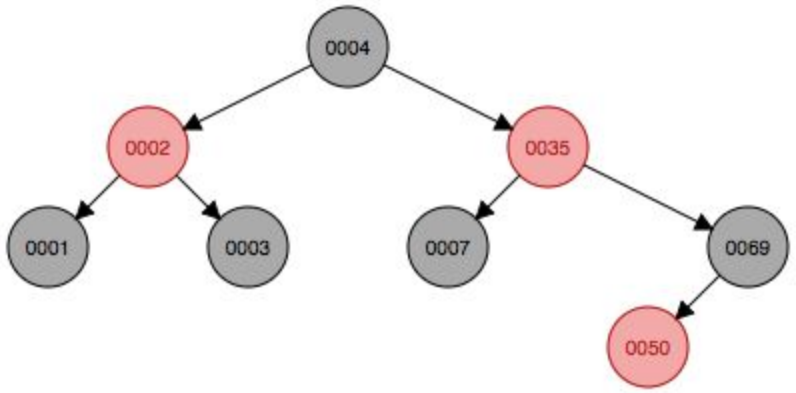
-This tree is clearly not the same as the original tree, so inserting a node and then deleting it does not always result in the original tree

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

Answer 2: No, deleting a node with no children from a red-black tree, re-balancing, and then re-inserting it with the same key does not always result in the original tree.

e.g.

Original Tree



Delete 7

Algorithm

-Store the node with the value of seven as the node to be deleted and store that its color is black.

-7 is not the root and it has no children, so set the left child of its parent 35 equal to a null node. Store the left child of seven as x.

- Since 7 is black, we have to rebalance around x

-After rebalancing, delete the memory space of 7

Rebalance after removing 7

Algorithm

-x (the null leaf that is currently 35's left child after the deletion of 7) is not the root and it is black so we need to rebalance.

-x is a left child, so set its sibling s equal to the right child of its parent 35, which is 69

-69 has a red left child, so recolor 69 to red and 50 to black and right rotate around 69.

-x's sibling s now equals the new black 50 after the rotate

-x is still black and not the root, so we still need to rebalance

-Since s (50) is black and has a red right child (69), we use case 4

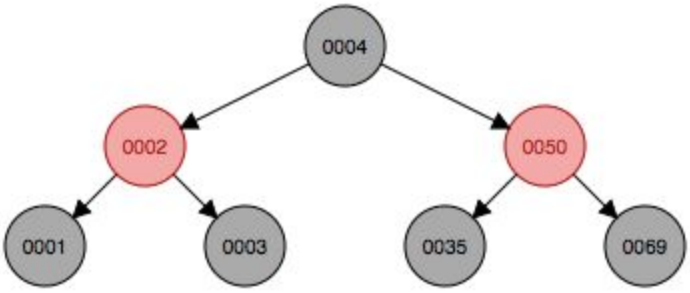
-Set 50's color to the color of x's parent 35 which is red

-Set x's parent 35 to black

-Set 69 to black again and left rotate around x's parent 35

-Balancing is now complete

Rebalanced Tree

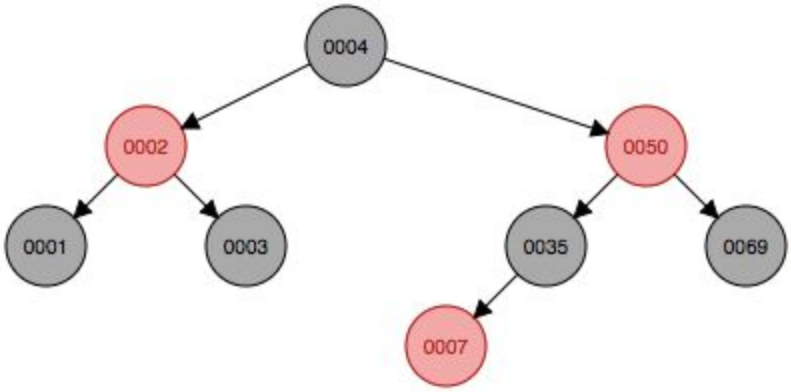


Re-Inserting 7

Algorithm

-First insert 7 as you would into a regular BST. 7 is greater than 4 so move to the right subtree. 7 is less than 50 so move to the left subtree. 7 is less than 35 and is pointing to a null node, so insert a red seven as the left child of 35.
-Set x equal to the the node containing 7.
-As the parent of x is not red, balancing is nearly complete
-Finally check that the root is black and as it is, balancing is complete.

Final Tree



-This is clearly a different tree than the original tree, so deleting a childless node, rebalancing, and then adding a node with the same key does NOT necessarily result in the exact same tree as the original.