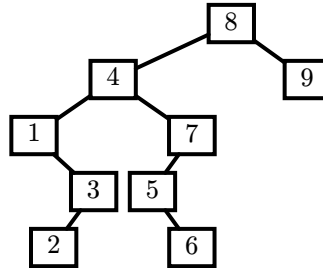
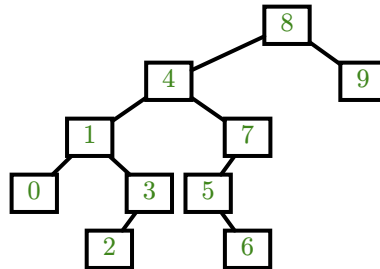


1 BST Insertion, Deletion, and Rotation (Lectures 17, 19)

- (a) Suppose we have the BST shown below. Insert 0.

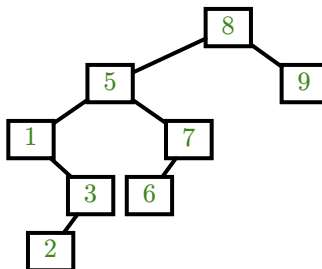


We insert 0 by traversing left from 8, left from 4, left from 1. Since 1 has no left child, 0 becomes the left child of 1.



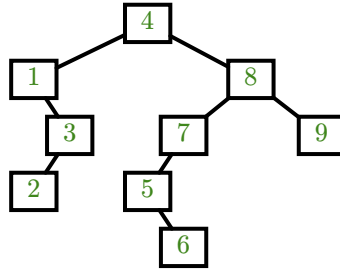
- (b) Now, starting from the original tree (before the insertion of 0), delete 4 using Hibbard deletion. Draw the new tree below. Check your answer with your TA's answer key.

To delete 4 using Hibbard deletion with the successor, we can use successor of 4, which is the smallest element in 4's right subtree. The right subtree of 4 is rooted at 7, and the smallest element in that subtree is 5. We replace 4 with 5, and then remove 5 from its original position. Since 5 had a right child (6), 6 takes 5's place as the left child of 7.



- (c) Now, starting from the original tree (before the insertion of 0), rotate 8 to the right. Draw the new tree below. Check your answer with your TA's answer key.

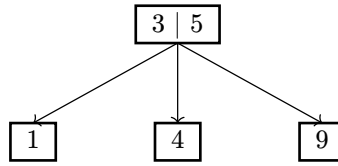
When we rotate 8 to the right, 4 becomes the new root. 8 becomes 4's right child. 4's old right subtree (rooted at 7) moves to become 8's new left child. 8's right child (9) is unchanged.



- (d) To better understand tree rotations, go to https://hugocode.com/61b/bst_rotation and balance the tree. Suggestion: play until you're able to do it without thinking too hard.

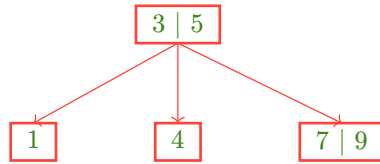
2 2-3 Tree Insertions (Lectures 18, 19)

- (a) Draw what the following 2-3 tree would look like after inserting 7, 6, 2, 8, and 8.5.

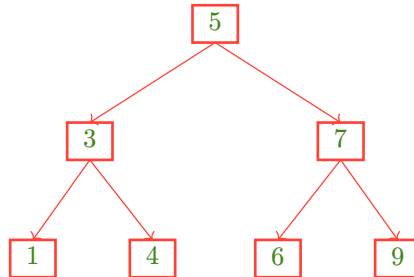


Solution:

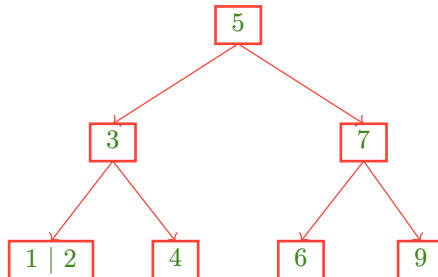
Adding 7:



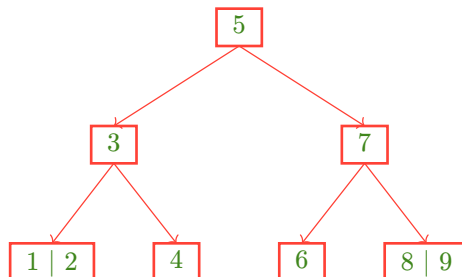
Adding 6: the leaf overflows to 6 | 7 | 9 and splits, sending 7 up. The root then overflows to 3 | 5 | 7 and splits, sending 5 up.



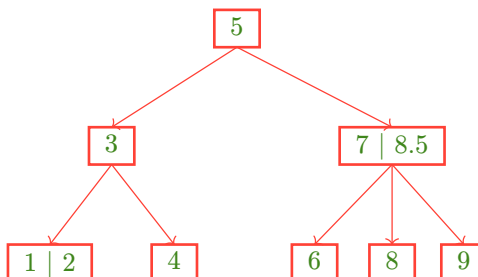
Adding 2:



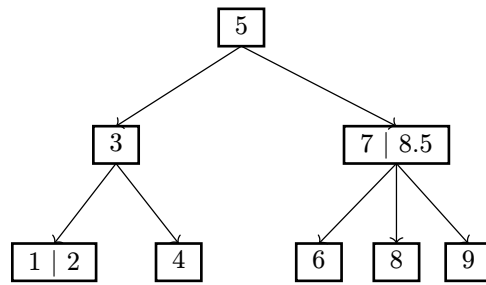
Adding 8:



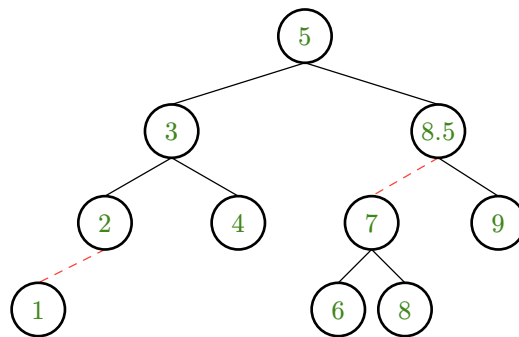
Adding 8.5: the leaf overflows to 8 | 8.5 | 9 and splits, sending 8.5 up into the 7 node.



(b) Convert the 2-3 tree below to a left-leaning red-black tree.

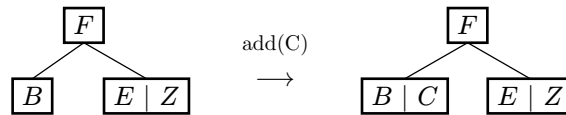


Solution:

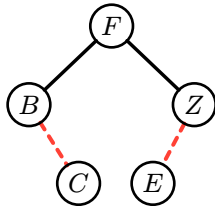


3 Deriving the LLRB Rotation Rules (Lecture 19)

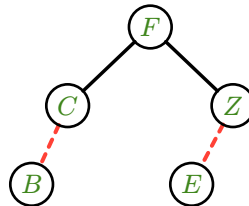
- (a) In 2-3 tree world, if we add C to the tree on the left, we get the tree on the right:



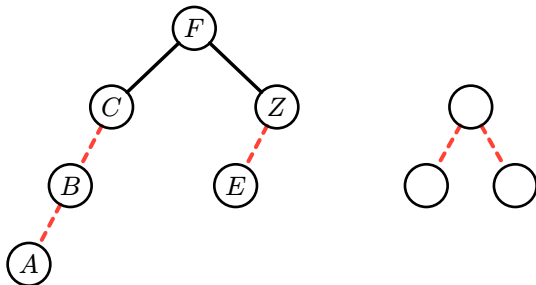
By contrast, in LLRB world, if we add C with a red link but don't perform any rotations, we end up with the tree below. What rotation do we perform to restore the proper 1-1 correspondence? Draw the entire tree after this rotation. Draw red links with a dashed line or label the line with "red".



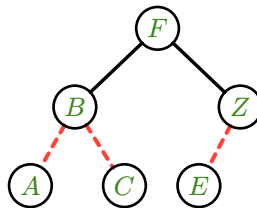
rotateLeft(B). The red link $B \rightarrow C$ leans right, which is not allowed in an LLRB. Rotating left makes C the parent, with a red link down-left to B. (A 2-3 node $B \mid C$ is always represented with the red link to its left child.)



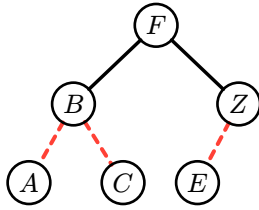
- (b) If we now insert A into the tree you just rotated, we get the tree on the left. We see that A, B, and C are the wrong shape to be a temporary 3-item node: the correct shape is shown on the right. What rotation do we perform? Draw the entire tree after this rotation.



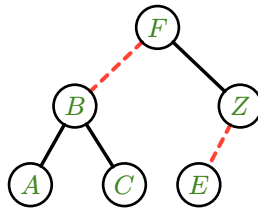
rotateRight(C). Two red links in a row on the left ($C \rightarrow B \rightarrow A$) means the middle item, B, should be the parent of the temporary 3-item node, with red links to both A and C.



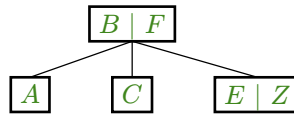
- (c) After the rotation, we have the tree below. If this were a 2-3 tree (i.e. a B-tree where the limit is 2 items), we'd call **split(B)**. On an LLRB, we instead call **colorFlip(B)**. Draw the result after the color flip.



The color flip turns B's two red links black and turns the link from B's parent red, which is exactly "push the middle item up" in 2-3 world.

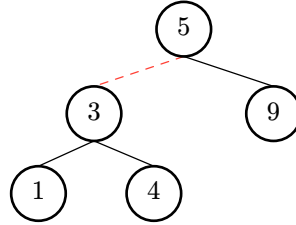


- (d) Lastly, convert this LLRB into its equivalent 2-3 tree.



4 LLRB Insertions (Lecture 19)

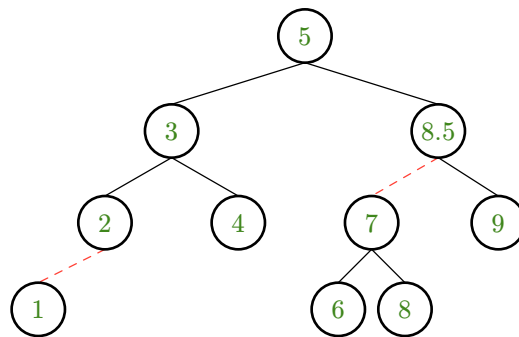
Given the LLRB below, perform the following insertions and draw the final state of the LLRB. In addition, for each insertion, write the balancing operations needed in the correct order (**rotateRight**, **rotateLeft**, or **colorFlip**). If no balancing operations are needed, write "Nothing". Assume that the link between 5 and 3 is red and all other links are black at the start.



- (a)
1. Insert 7
 2. Insert 6
 3. Insert 2
 4. Insert 8
 5. Insert 8.5 (warning: this one is a bit lengthy, you may want another sheet of paper)
 6. Final state

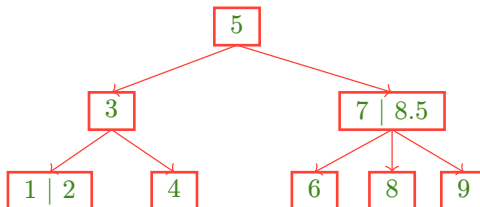
Solution: For a visualization of the process, see [here](#)

1. Insert 7:
 - Nothing
2. Insert 6:
 - rotateRight(9)
 - colorFlip(7)
 - colorFlip(5)
3. Insert 2:
 - rotateLeft(1)
4. Insert 8:
 - Nothing
5. Insert 8.5:
 - rotateLeft(8)
 - rotateRight(9)
 - colorFlip(8.5)
 - rotateLeft(7)
6. Final state



- (b) Convert the final LLRB to its corresponding 2-3 Tree.

Solution:



- (c) For more practice with LLRB insertion, try the game mode at <https://hugocode.com/61b/llrb/>.