AV121 – Data Structures and algorithms

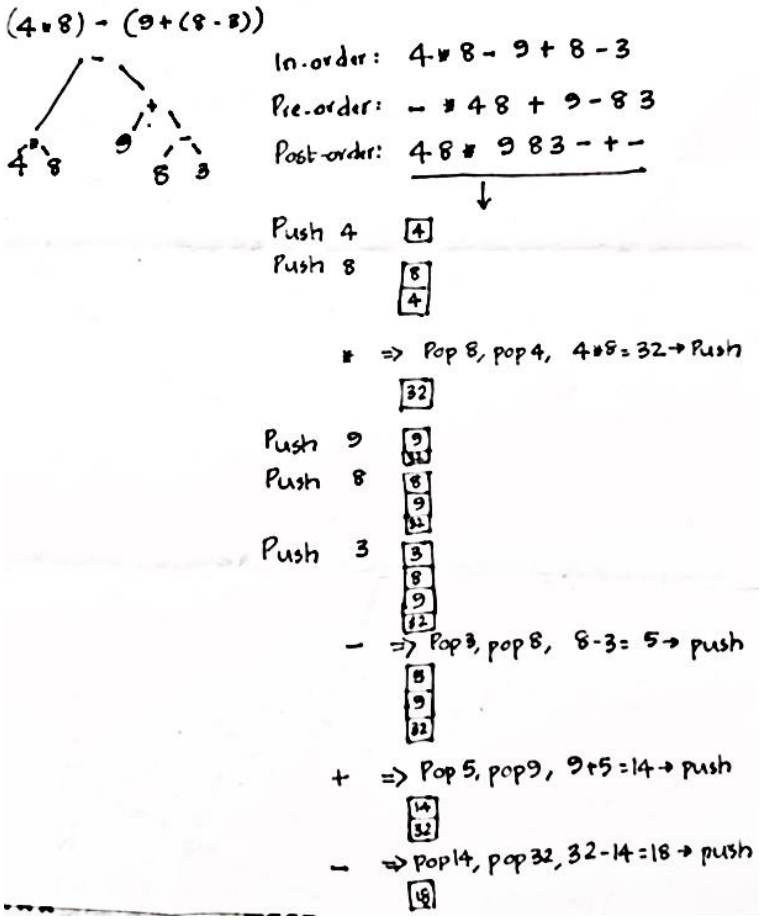
Assignment 1 – Answer Key

Part 1

1.1 Given the following mathematical expression:

(4*8) - (9 + (8-3))

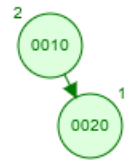
- Construct the equivalent expression tree.
- Write down the inorder, preorder and postorder traversals of the tree.
- If the postorder traversal is given as input to a computer for evaluation using a stack, list down the steps and the content of the stack after each step.

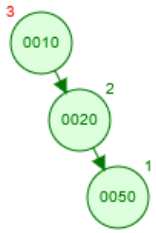


1.2 Construct an AVL tree from the following elements (in the order given below)

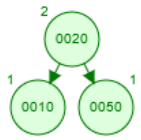
10, 20, 50, 90, 80, 5, 1

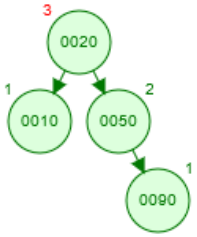
- Show the tree after inserting each number.
- After inserting the last number, what is the height of the resulting AVL tree?



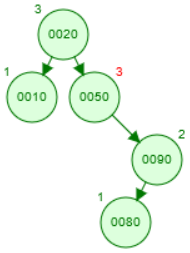


Node 10 has a balance factor 2, hence the tree must be rebalanced by doing a left rotate.

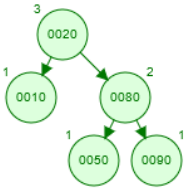




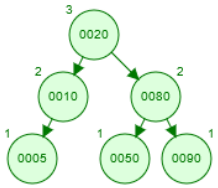
This is still a valid AVL tree. No rotations are needed here.

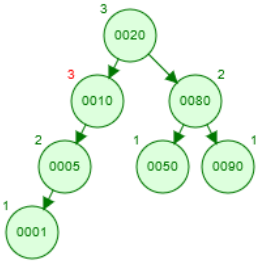


After inserting 80, there is an imbalance at node 50. This is a right-left imbalance. So, do a right rotation of 90-80 first, then do a left rotation. This results in 50 and 90 becoming the left and right children of 80. 80 will be the right child of 20.

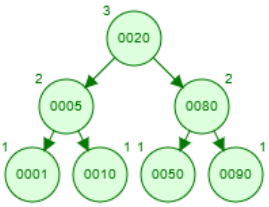


No rotations are needed after inserting 5, because the resulting tree remains a valid AVL tree.



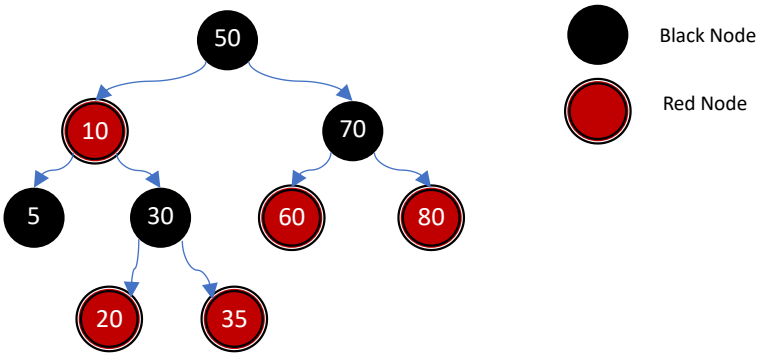


Here, node 10 has a balance factor of -2, and we must do a right rotation to fix it.



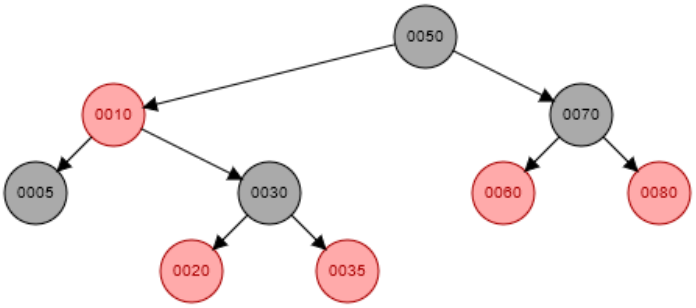
The final tree has a height of 2.

1.3 Given the following Red-Black tree:

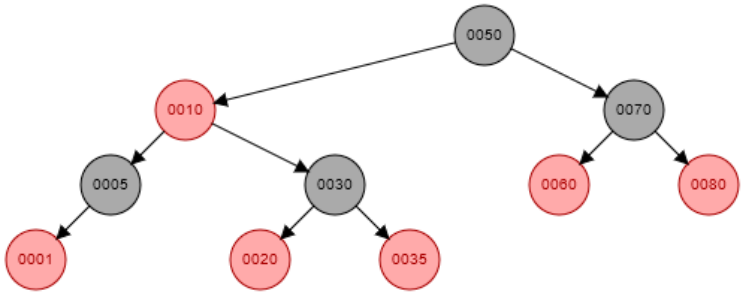


- Show the steps required in inserting the following numbers and the steps required (if any) to maintain the Red-Black tree rules:
    - Insert 1
    - Insert 85
    - Insert 6

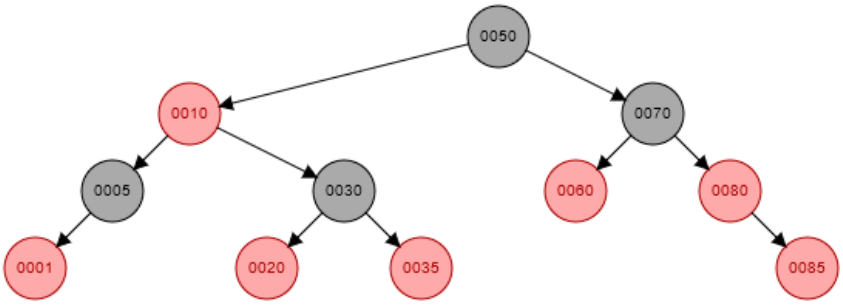
  Show the tree after each step.



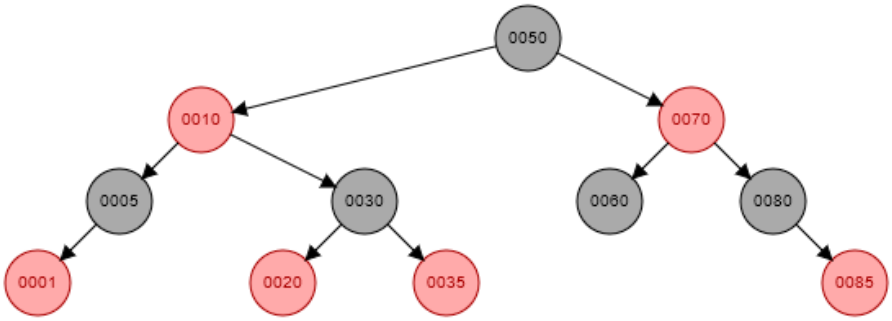
1 will be inserted as the left child of 5, as a red node. This doesn't violate any RB Tree properties. So, no rotation or recoloring is needed.



85 will be inserted as the right child of 80 (as a red node), resulting in the following tree:

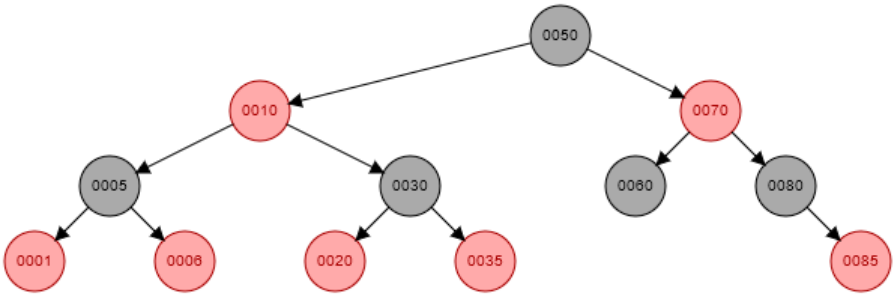


This violates the RB Tree property because 80 (which is a red node) cannot have a red child. This is a case where the uncle of the new node is also red, that is, case 1 from the lecture slides. We will attempt to fix it by doing a recoloring – Change the color of uncle and parent to black, and to cancel out the increase in black height, change the color of the grandparent to red.



This is the resulting tree, after the recoloring. This is now a valid RB Tree.

6 will be inserted as the right child of 5, as a red node. This doesn't violate any RB Tree properties. So, no rotation or recoloring is needed here also.



This is the final tree, after inserting the three numbers.