Red black tree Step by Step Rules

General Rules

1. Always keep as a Binary Search tree by following its properties.
2. Every node is colored either red or black.
3. The root is colored black.
4. If a node is red, its children are black.
5. Every path from a node to a null link must contain the same number of black nodes. This number is called the black height of x and is denoted bh(x).

   ***Note*:** The black height of NULL is 0.

Insertion Rules

Rule 1 - If tree is Empty then insert the **newNode** as Root node with color **Black** and exit from the operation.

Rule 2 - If tree is not Empty then insert the newNode as leaf node with color **Red**.

Rule 3 - If the parent of newNode is **Black** then exit from the operation.

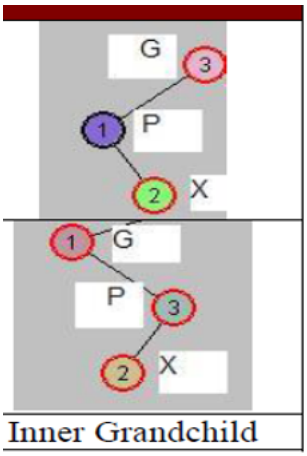
Rule 4 - If the parent of newNode is **Red** then check the color of parent node's sibling(Uncle) of newNode.

- **a.** If sibling is colored Black or NULL then make suitable Rotation discussed in the AVL tree(L-Rotation, R-Rotation, LR-Rotation or RL-Rotation) and Recolor it.
- **b.** If sibling is colored Red then perform Recolor & also check if Parent's Parent(Grand Parent-GP) of newNode is not root node then recolor it. Recheck the RB property until reach the root node.

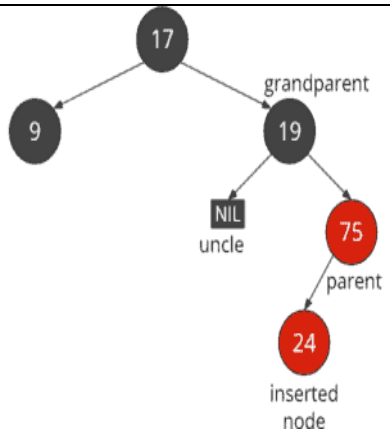
Recoloring Rule 4(a)

"Inner grandchild" means that the path from the grandparent node to the inserted node forms a triangle.

Scenario



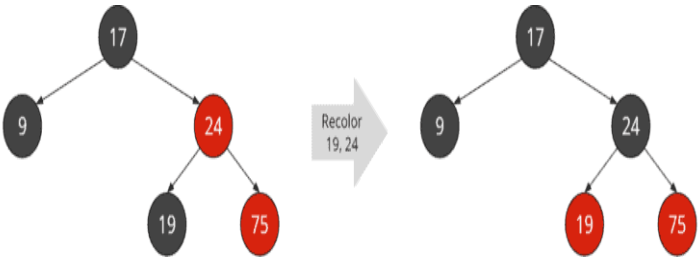
X is the newly inserted node



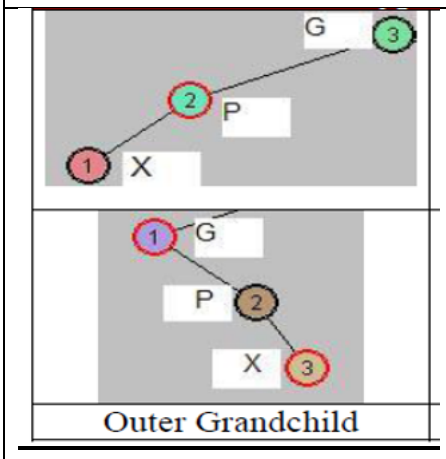
Color change and flip rules of Inner grandchild:

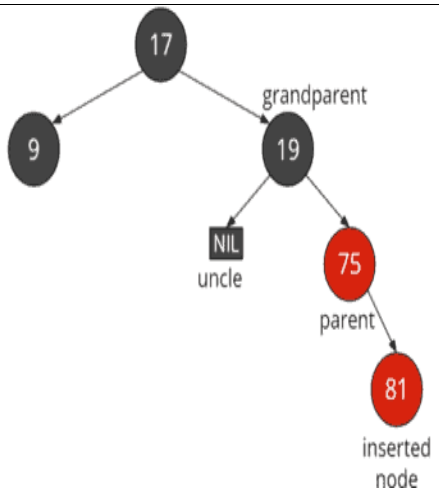
1. Perform suitable double rotation.
2. Change color of GP
3. Change color of new inserted node X

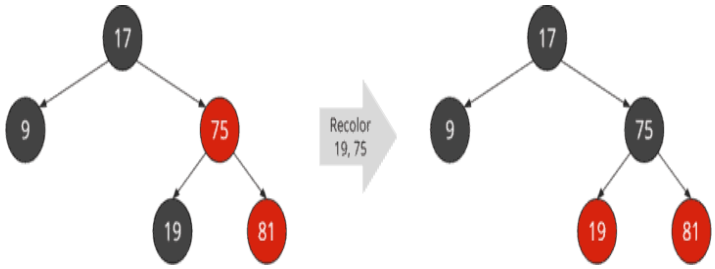
After double rotation and color change



"Outer grandchild" means that the path from grandparent to inserted node forms a line





	Color change and flip rules of Outer grandchild: • Change color of GP • Change color of P • Rotate P, G in the direction that lifts X 

You must satisfy the General Rules of Red Black Tree

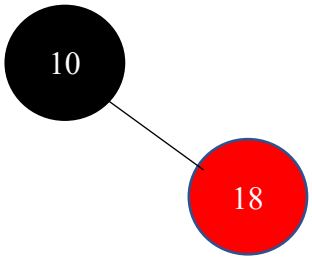
1. Root is black in color.
2. No two adjacent Red Nodes.
3. Every path, ensure same count of black nodes.

Insert the below set of nodes in the Red black Tree

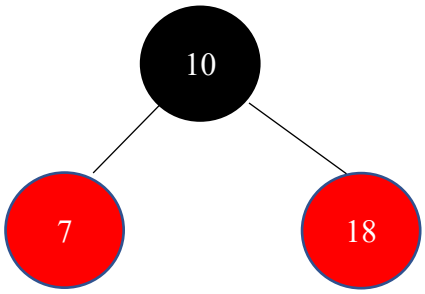
10 18 7 15 16 30 25 40 60 2 1 70

Step 1: Tree is Empty and insert 10 as root and colored as Black.

Step 2: Insert 18, Tree is not Empty. Should be inserted at the right side of 10, as a leaf node and colored as Red. Rules are obeyed. No violation. Successful insertion.



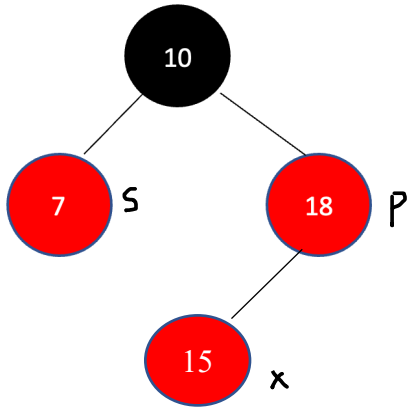
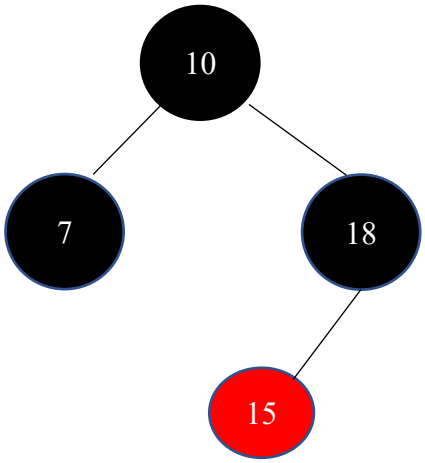
Step 3: Insert 7, Tree is not Empty. Should be inserted at the left side of 10, as a leaf node and colored as Red. Rules are obeyed. No violation. Successful insertion.



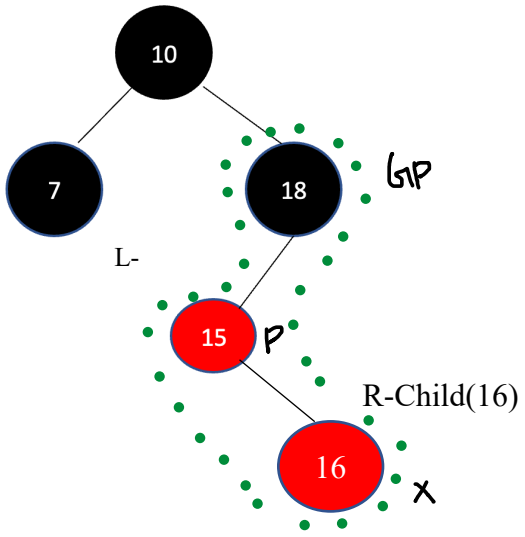
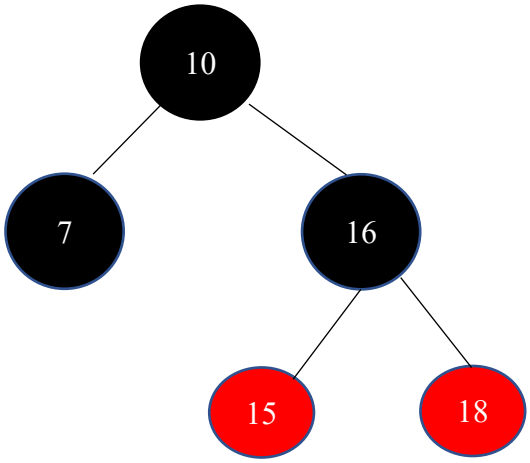
Step 4: Insert 15, Should be inserted at the left side of 18, as a leaf node and colored as Red. Adjacent of two red nodes. It's a **Rule violation**.

> **Rule 4 -** If the parent of newNode is Red then check the color of parent node's sibling of newNode.
>
> **a.** If it is colored Black or NULL then make suitable Rotation(AVL Rotations) and Recolor it.
> **b.** If it is colored Red then perform Recolor & also check if Parent's Parent(Grand Parent) of newNode is not root node then recolor it

Parent P = 18, Sibling S of 18 is 7. Apply the Rule 4. Now the Sibling is Red. Need to Recolor both Parent and it's Siblings. Red recolored as Black for both P and S. Here 18 and 7 are recolored in Black. Check G, grant Parent. Here it's a root. No recolor the root. Done insertion.

Before Change [Red-Red-Violation]	After Change [Proper RBT]
	

Step 5: Insert 16, Should be inserted at the right side of 15, as a leaf node and colored as Red. Adjacent of two red nodes rule violation. Check with the Rule 4, Parent 15 Sibling is null. Need to make a LR Rotation and Recolor it. Recolor 16 and 18.

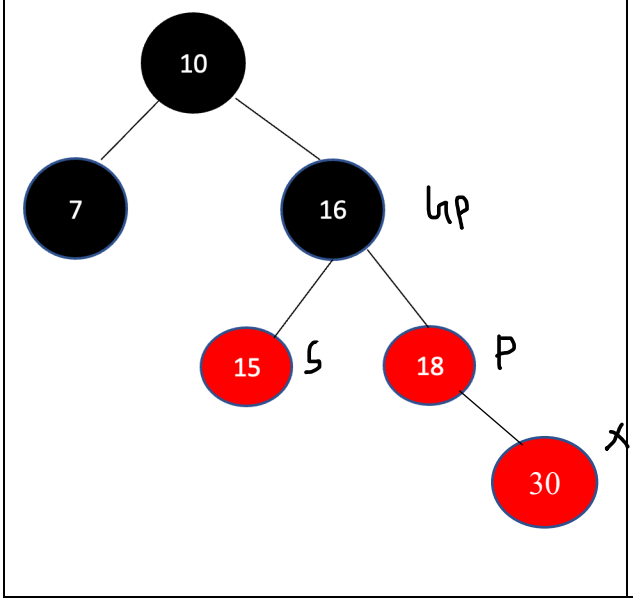
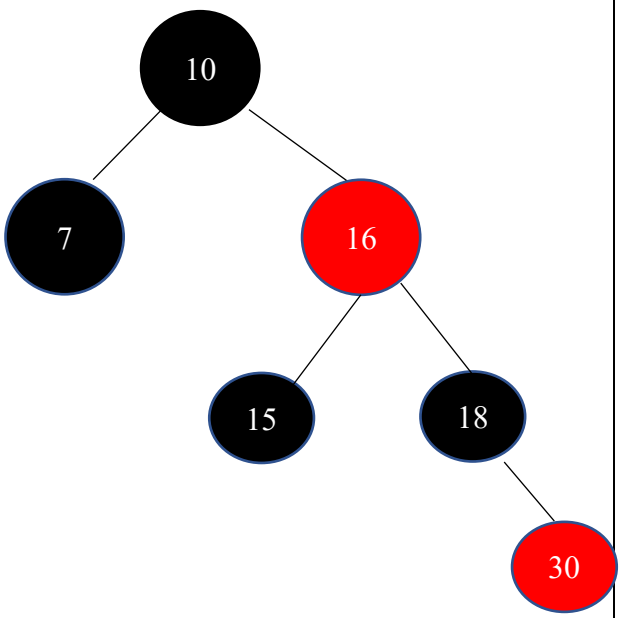
Before Change [Red-Red-Violation]	After Change [Proper RBT]
	

Here Recolored the Grand Parent and new node X using Inner Grand child.

Step 6: Insert 30, Should be inserted at the left side of 18, as a leaf node and colored as Red. Adjacent of two red nodes rule violation.

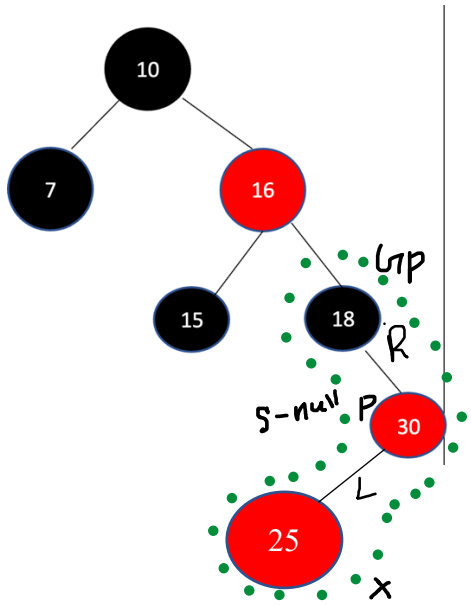
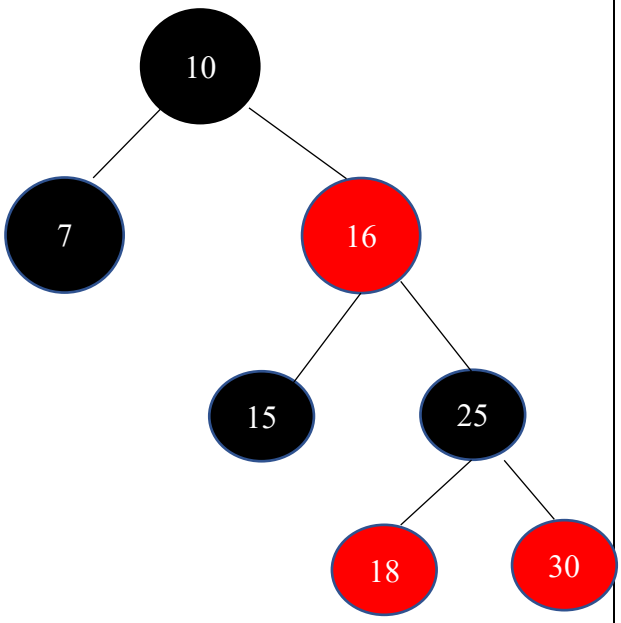
Check with the Rule 4, Parent's sibling is Red. So, perform recolor with Parent and Parent's Sibling, no rotation. After this color change 15 and 18 colored as black.

Then check the newNode's Grand Parent (GP) which is 16, and not a root. So recolor 16 with Red.

Before Change [Red-Red-Violation]	After Change [Proper RBT]
	

Step 7: Insert 25, Should be inserted at the left side of 30, as a leaf node and colored as Red. Adjacent of two red nodes rule violation.

Check with the Rule 4, Parent's sibling is null. So perform RL double rotation & Recolor 25 & 18.

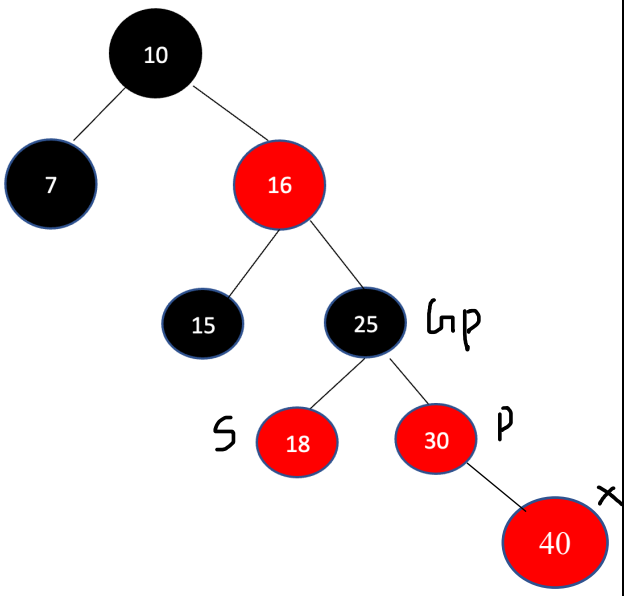
Before Change [Red-Red-Violation]	After Change [Proper RBT]
	

Grand Parent and New node X got color change using Inner Child Property.

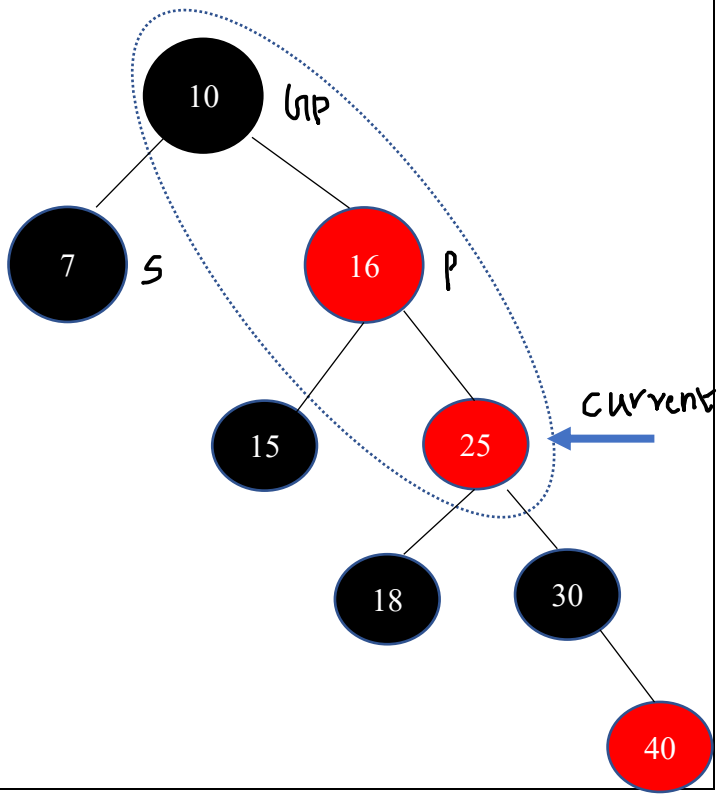
Step 8: Insert 40, Should be inserted at the left side of 30, as a leaf node and colored as Red. Adjacent of two red nodes rule violation.

Parent Sibling is Red, No rotation, just recolor P(30) and S(18) as Black. GP is not a root, recolor 25.

Before Change [Red-Red-Violation]

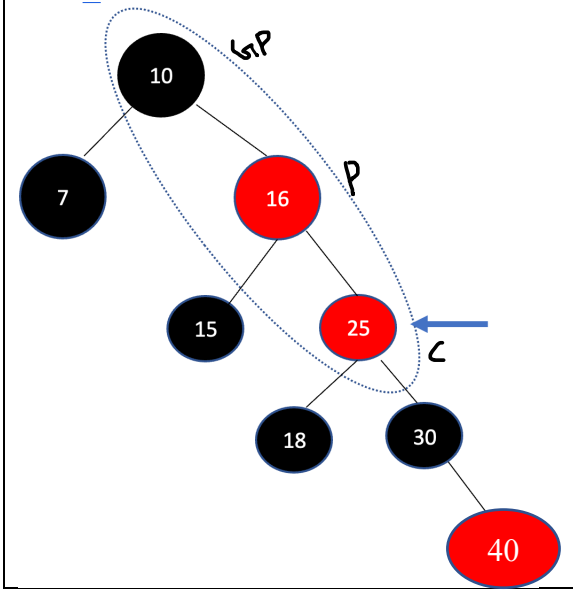


After Recolor GP, P and S, While recheck got a Red-Red violation. Now 25 becomes the current new Node. Check P(16), it's sibling S(7) is a black. Need L-Rotation.

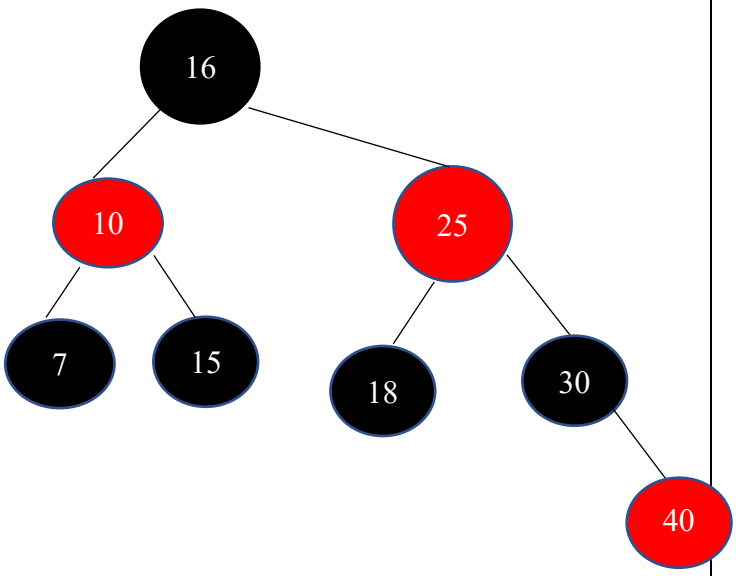


Here it's a cross over Right to Left rotation. 16 becomes the Root and 10 is the left child of 16, and 25 becomes the right child of 16. Here 16 has the left child 15, will become right child of 10.

After rotation recoloring required. 16 becomes black color and 10 becomes Red color.



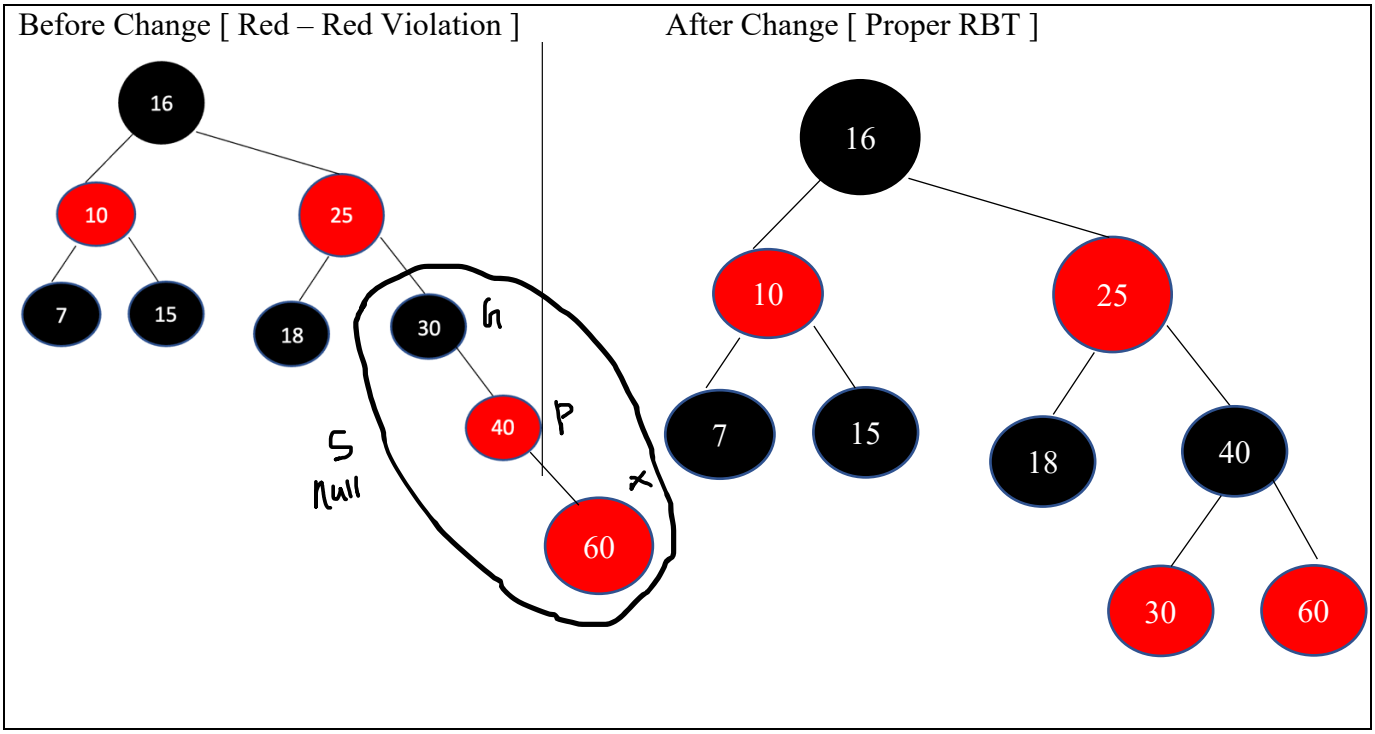
After Change [Proper RBT]



Due to 16 becomes root, Parent and Grand Parent got color change using Outer grand child.

Step 9: Insert 60, Should be inserted at the right side of 40, as a leaf node and colored as Red. Adjacent of two red nodes rule violation.

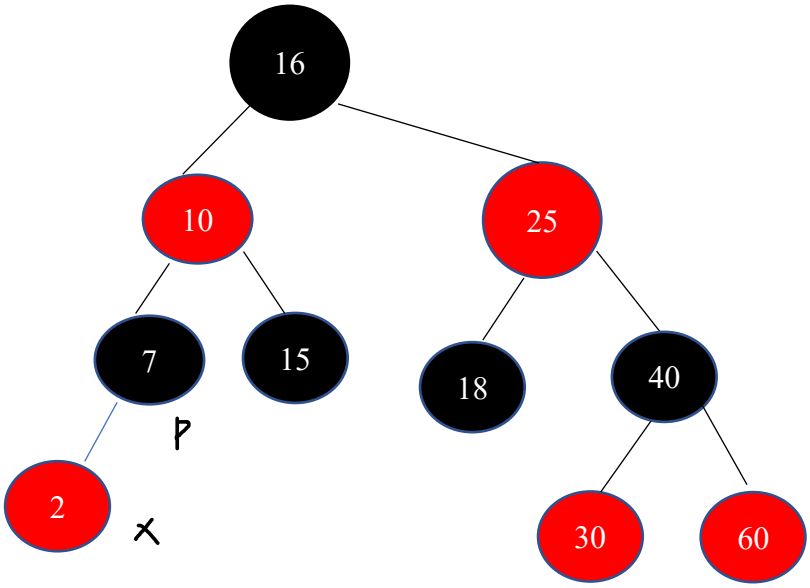
Apply Rule 4, check Parent(40) Sibling is null. Need L-Rotation and Recolor. 40 becomes root, 30 is the left child of 40 and 60 becomes right child of 30. Then 30 becomes red and 40 becomes black.



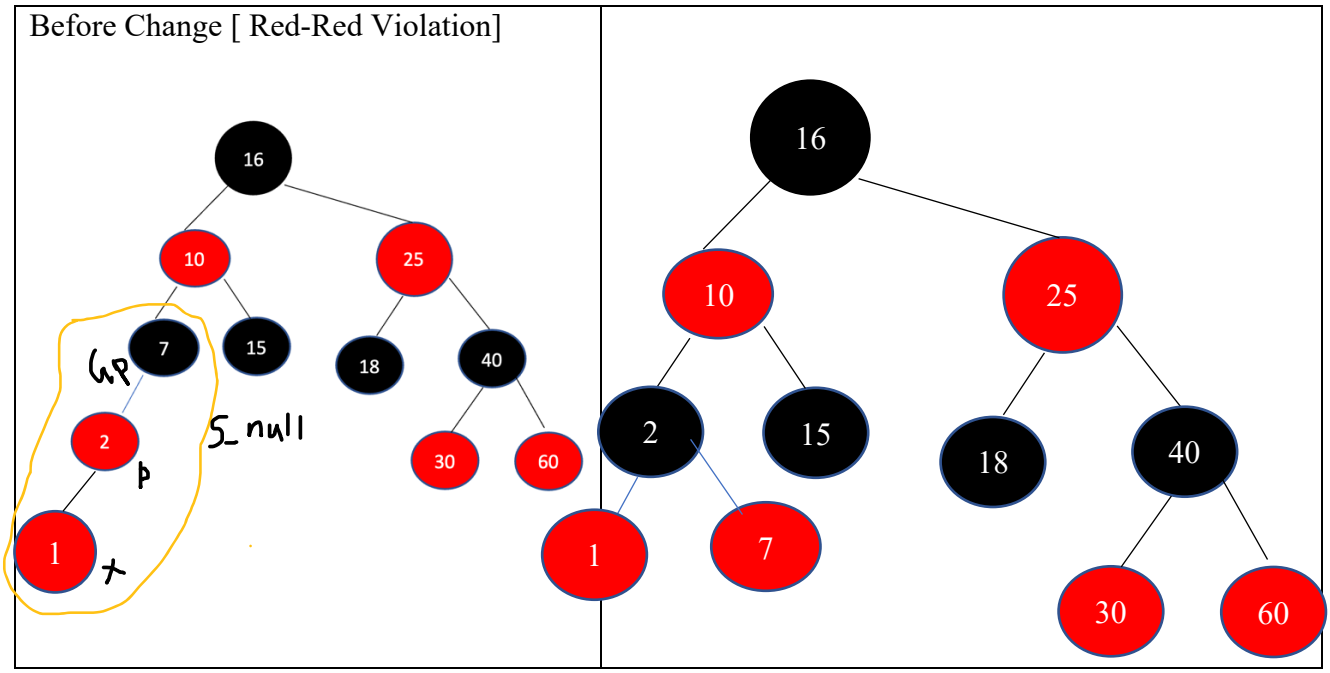
Here GP and P got color change. It's easy to use the color of Sibling after rotation.

Step 10: Insert 2, Should be inserted at the left side of 7, as a leaf node and colored as Red. Parent 7 is black. Exit operation.

After Inserting 2 [Proper RBT]

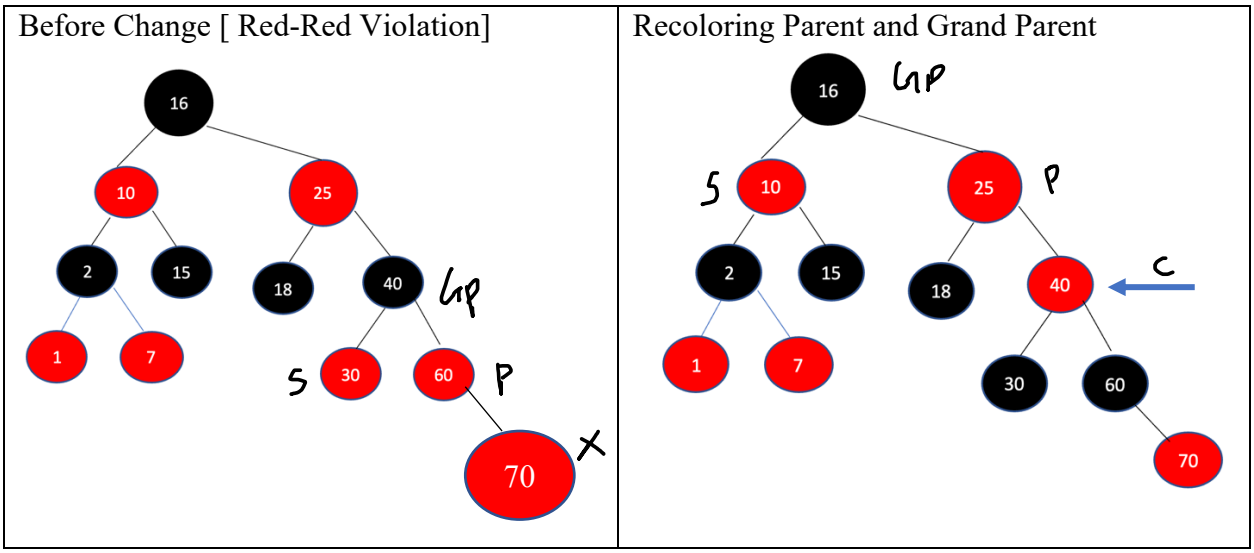


Step 11: Insert 1, Should be inserted at the left side of 2, as a leaf node and colored as Red. Parent(2) Sibling is null. So, need R-rotation and Recoloring. 2 Becomes the root, 7 becomes the right child of 2, 1 becomes the left side of 2. Then recoloring of 7 to Red and 2 to Black



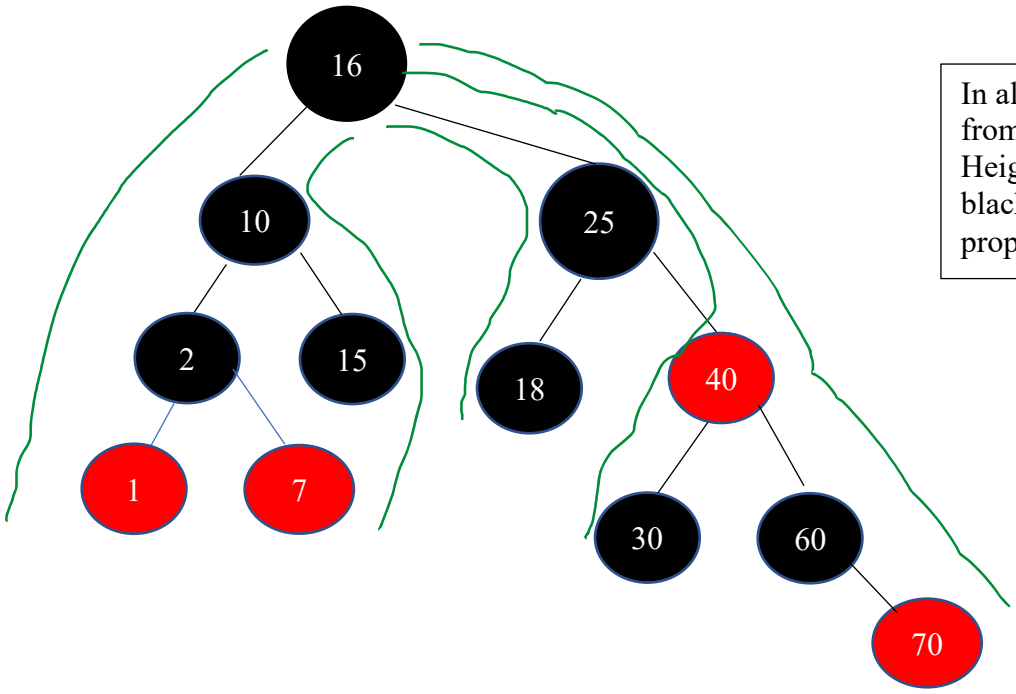
Here GP and P got color change using outer grandchild.

Step 12: Insert 70, Should be inserted at the right side of 60, as a leaf node and colored as Red. Will get Red-Red violation.
Apply Rule 4, check Parent(60) Sibling(30) is Red color. So Recolor 30 and 60 into black color, recolor Grand Parent G(40) to Red will create Red-Red-Violation between 40 and 25.



Here current new node is 40, check Parent(25) Sibling(10) is color Red. So We need to recolor as black for 10 and 25. The Grand Parent(16) is the root. So exit the operation. Now tree becomes proper RBT.

Final RBT for the given set of input numbers.

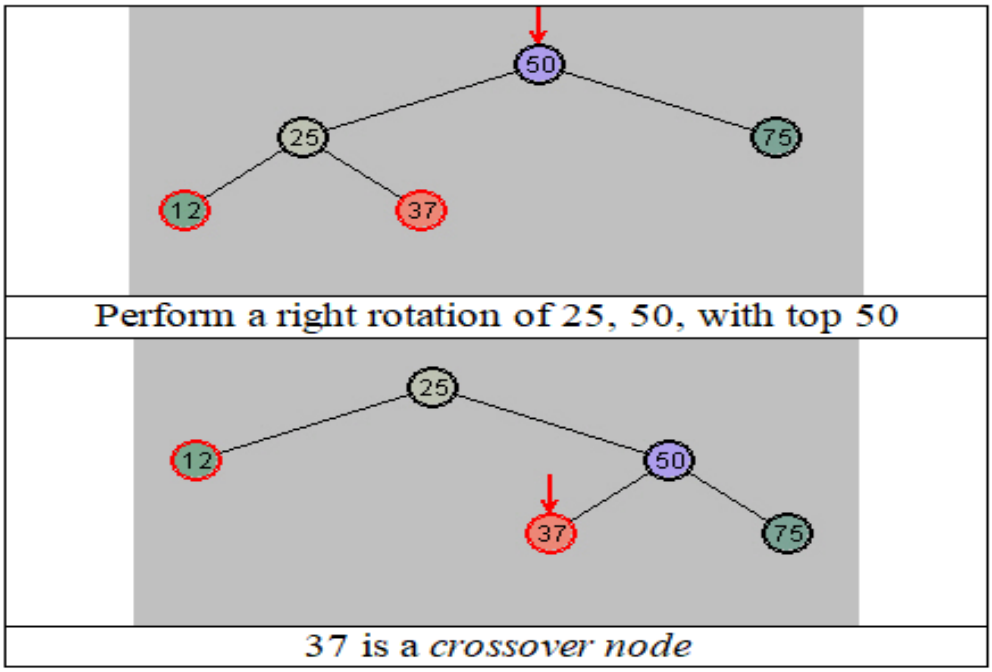


In all the green color paths from root to leaf the black node Height is same. Or counts of black nodes are same. It's proper Red-Black tree.

Complicated Gross Over Rotations

Example 1: Refer the Step 8.

Example 2: In complicated rotations (for instance, in a right rotation where the child being lifted already has a right child), there may be a *crossover node.*



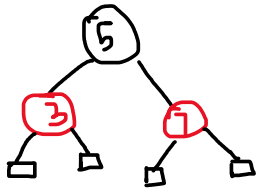
Quick summary about Red-Black tree Properties

Insertion, Deletion and Searching on Red-Black tree is O(log n) due to balanced tree height.

Node Colors: Every node is either red or black.

Root Property: The root is always black.

Leaf Property: Every leaf NIL node is black.



Red Node Property: If a red node exists, both its children are black (no two red nodes can be adjacent).

Black Height Property: Every path from a node to its leaf NIL nodes has the same number of black nodes.

Visualization Link

https://www.cs.usfca.edu/~galles/visualization/RedBlack.html