Red-Black Tree | Set 2 (Insert)

In the previous post, we discussed introduction to Red-Black Trees. In this post, insertion is discussed.

In AVL tree insertion, we used rotation as a tool to do balancing after insertion caused imbalance. In Red-Black tree, we use two tools to do balancing.

1) Recoloring

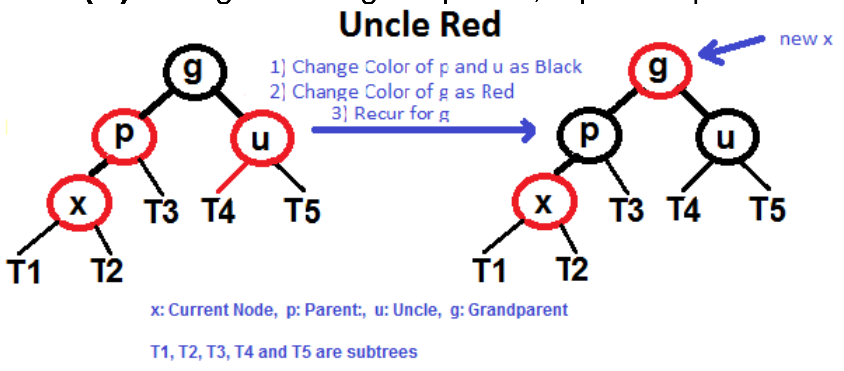
2) Rotation

1) Every node has a color either red or black.
2) Root of tree is always black.
3) There are no two adjacent red nodes (A red node cannot have a red parent or red child).
4) Every path from root to a NULL node has same number of black nodes.

We try recoloring first, if recoloring doesn't work, then we go for rotation. Following is detailed algorithm. The algorithms has mainly two cases depending upon the color of uncle. If uncle is red, we do recoloring. If uncle is black, we do rotations and/or recoloring.

Let x be the newly inserted node.
1) Perform standard BST insertion and make the color of newly inserted nodes as RED.

2) Do following if color of x's parent is not BLACK or x is not root.
….**a) If x's uncle is RED** (Grand parent must have been black from property 4)
……..**(i)** Change color of parent and uncle as BLACK.
……..**(ii)** color of grand parent as RED.
……..**(iii)** Change x = x's grandparent, repeat steps 2 and 3 for new x.


Uncle Red
1) Change Color of p and u as Black
2) Change Color of g as Red
3) Recur for g
new x
g
p
u
x
T1
T2
T3
T4
T5
x: Current Node, p: Parent:, u: Uncle, g: Grandparent
T1, T2, T3, T4 and T5 are subtrees

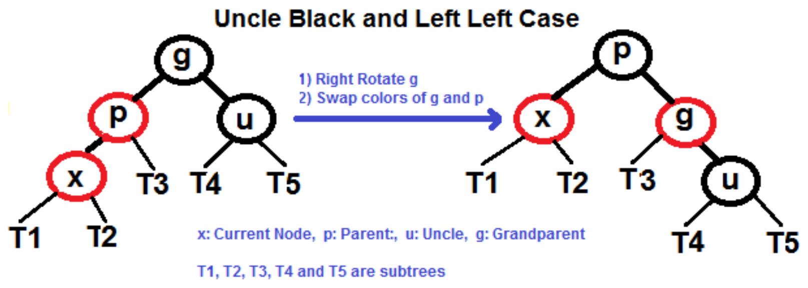
….**b) If x's uncle is BLACK**, then there can be four configurations for x, x's parent (**p**) and x's grandparent (**g**) (This is similar to AVL Tree)
……..**i)** Left Left Case (p is left child of g and x is left child of p)
……..**ii)** Left Right Case (p is left child of g and x is right child of p)
……..**iii)** Right Right Case (Mirror of case a)
……..**iv)** Right Left Case (Mirror of case c)

3) If x is root, change color of x as BLACK (Black height of complete tree increases by 1).

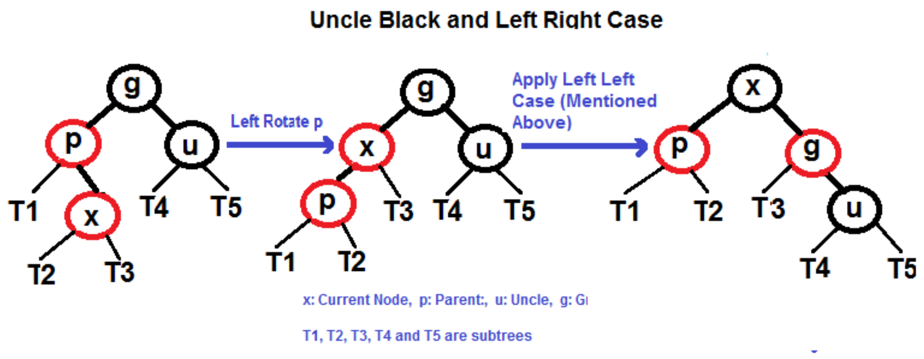
Following are operations to be performed in four subcases when uncle is BLACK.

All four cases when Uncle is BLACK

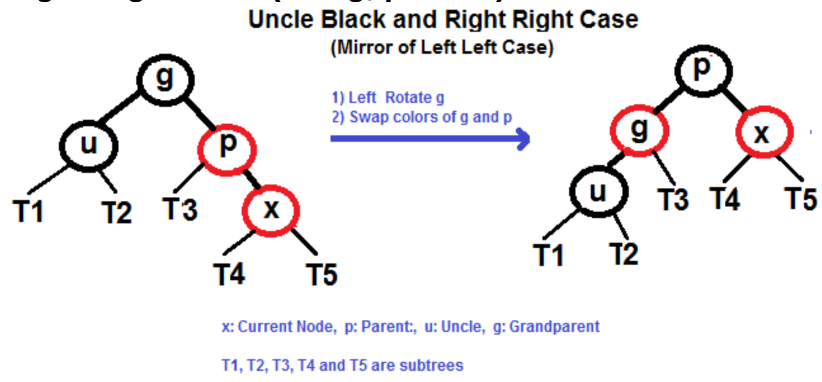
Left Left Case (See g, p and x)


Uncle Black and Left Left Case
g
p
u
x
T1
T2
T3
T4
T5
1) Right Rotate g
2) Swap colors of g and p
p
x
g
u
T1
T2
T3
T4
T5
x: Current Node, p: Parent:, u: Uncle, g: Grandparent
T1, T2, T3, T4 and T5 are subtrees

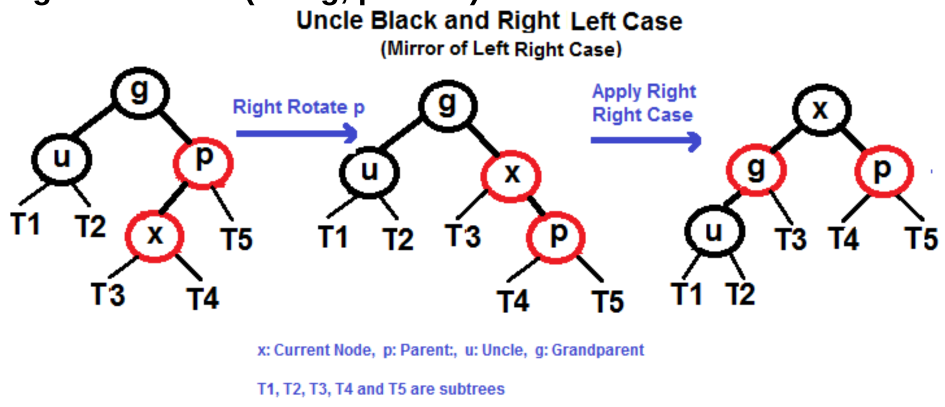
Left Right Case (See g, p and x)


Uncle Black and Left Right Case
g
p
u
T1
x
T4
T5
T2
T3
Left Rotate p
g
x
u
p
T3
T4
T5
T1
T2
Apply Left Left Case (Mentioned Above)
x
p
g
T1
T2
T3
u
T4
T5
x: Current Node, p: Parent:, u: Uncle, g: Gı
T1, T2, T3, T4 and T5 are subtrees

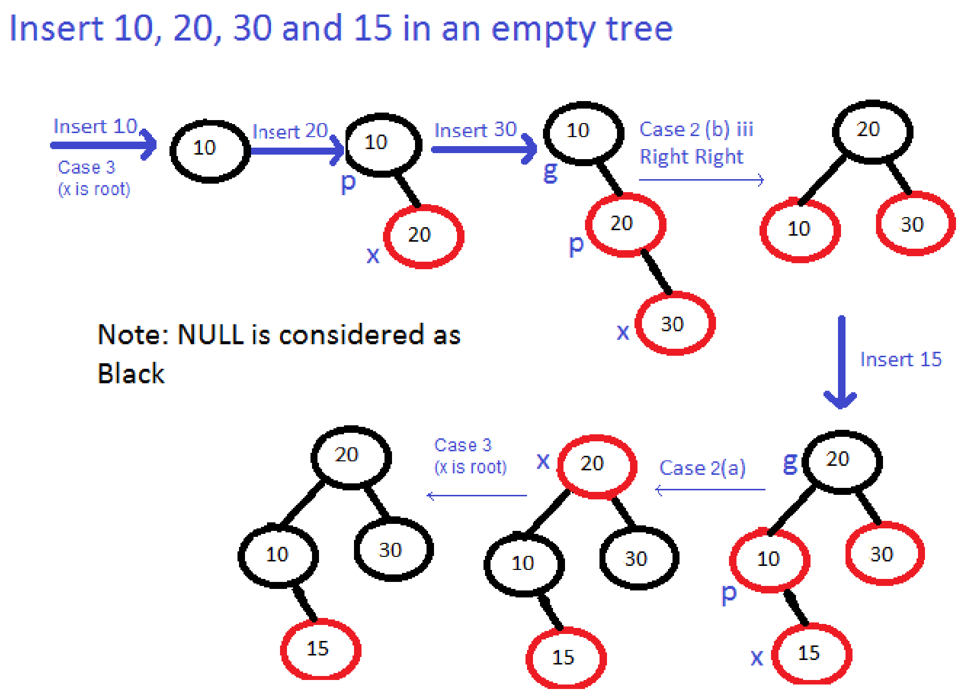
Right Right Case (See g, p and x)


Uncle Black and Right Right Case
(Mirror of Left Left Case)
g
u
p
T1
T2
T3
x
T4
T5
1) Left Rotate g
2) Swap colors of g and p
p
g
x
u
T3
T4
T5
T1
T2
x: Current Node, p: Parent:, u: Uncle, g: Grandparent
T1, T2, T3, T4 and T5 are subtrees

Right Left Case (See g, p and x)


Uncle Black and Right Left Case
(Mirror of Left Right Case)
Right Rotate p
Apply Right Right Case
g
u
p
x
T1
T2
T3
T4
T5
x: Current Node, p: Parent:, u: Uncle, g: Grandparent
T1, T2, T3, T4 and T5 are subtrees

Examples of Insertion



Please refer C Program for Red Black Tree Insertion for complete implementation of above algorithm.