Red-Black Tree | Set 3 (Delete)

We have discussed following topics on Red-Black tree in previous posts. We strongly recommend to refer following post as prerequisite of this post.

Red-Black Tree Introduction
Red Black Tree Insert

Insertion Vs Deletion:
Like Insertion, recoloring and rotations are used to maintain the Red-Black properties.

In insert operation, we check color of uncle to decide the appropriate case. In delete operation, ***we check color of sibling*** to decide the appropriate case.

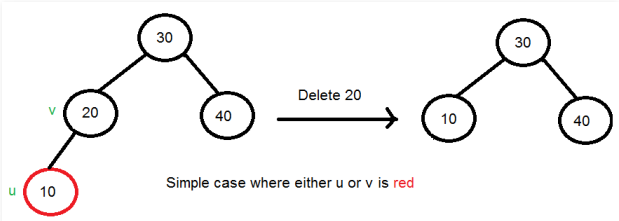
The main property that violates after insertion is two consecutive reds. In delete, the main violated property is, change of black height in subtrees as deletion of a black node may cause reduced black height in one root to leaf path.

Deletion is fairly complex process. To understand deletion, notion of double black is used. When a black node is deleted and replaced by a black child, the child is marked as ***double black***. The main task now becomes to convert this double black to single black.

Deletion Steps
Following are detailed steps for deletion.

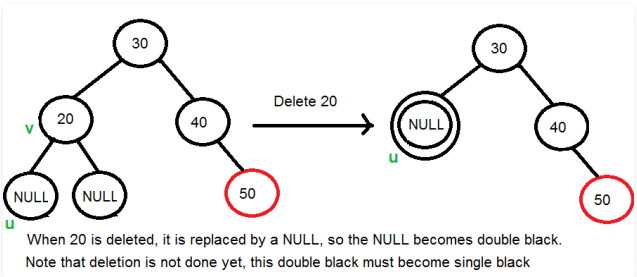
1) Perform standard BST delete. When we perform standard delete operation in BST, we always end up deleting a node which is either leaf or has only one child (For an internal node, we copy the successor and then recursively call delete for successor, successor is always a leaf node or a node with one child). So we only need to handle cases where a node is leaf or has one child. Let v be the node to be deleted and u be the child that replaces v (Note that u is NULL when v is a leaf and color of NULL is considered as Black).

2) Simple Case: If either u or v is red, we mark the replaced child as black (No change in black height). Note that both u and v cannot be red as v is parent of u and two consecutive reds are not allowed in red-black tree.



3) If Both u and v are Black.

3.1) Color u as double black. Now our task reduces to convert this double black to single black. Note that If v is leaf, then u is NULL and color of NULL is considered as black. So the deletion of a black leaf also causes a double black.

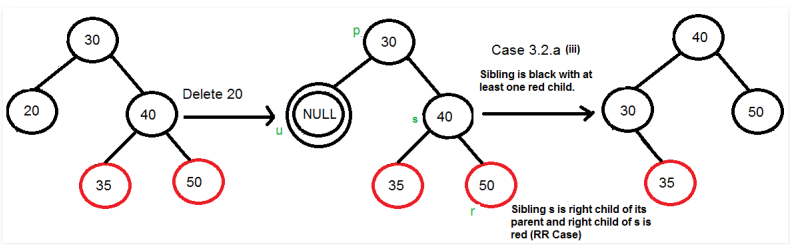


3.2) Do following while the current node u is double black and it is not root. Let sibling of node be **s**.
….**(a): If sibling s is black and at least one of sibling's children is red**, perform rotation(s). Let the red child of s be **r**. This case can be divided in four subcases depending upon positions of s and r.

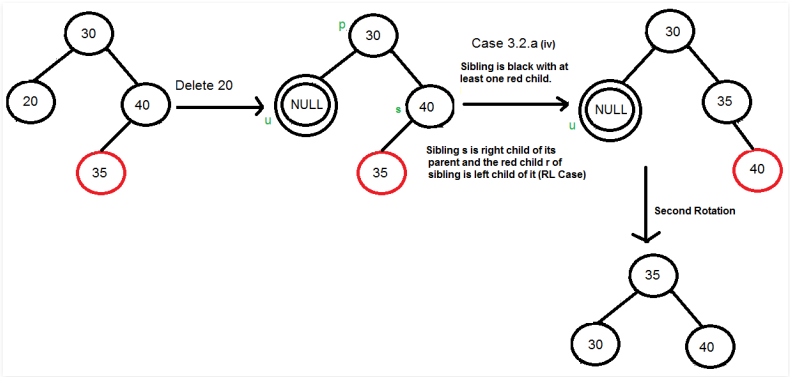
……..**(i)** Left Left Case (s is left child of its parent and r is left child of s or both children of s are red). This is mirror of right right case shown in below diagram.

……..**(ii)** Left Right Case (s is left child of its parent and r is right child). This is mirror of right left case shown in below diagram.

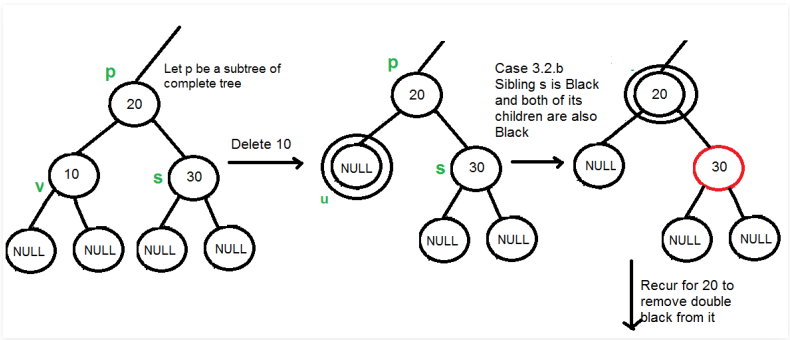
……..**(iii)** Right Right Case (s is right child of its parent and r is right child of s or both children of s are red)



……..**(iv)** Right Left Case (s is right child of its parent and r is left child of s)

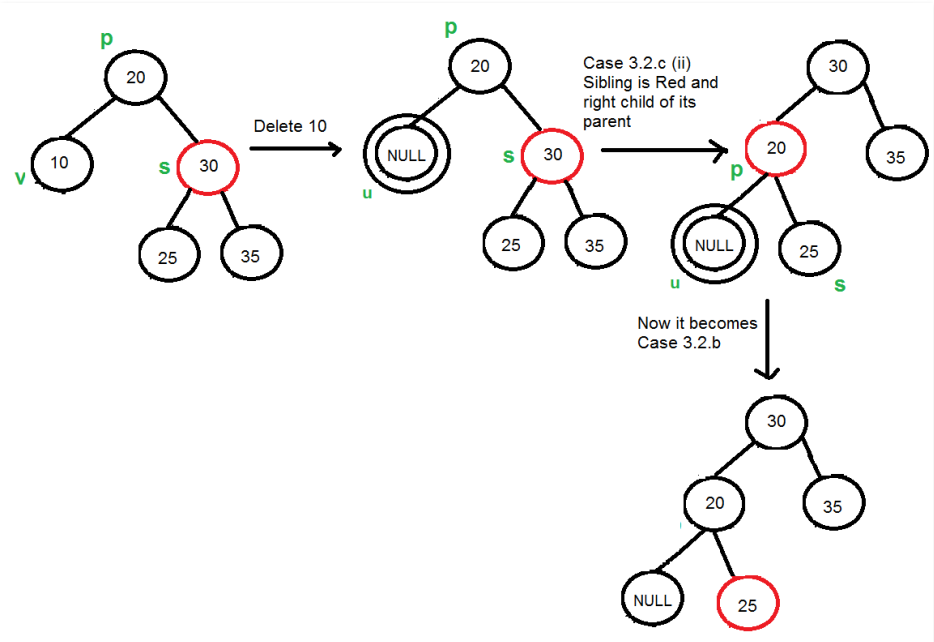


…..**(b): If sibling is black and its both children are black**, perform recoloring, and recur for the parent if parent is black.



In this case, if parent was red, then we didn't need to recur for prent, we can simply make it black (red + double black = single black)

…..**(c): If sibling is red**, perform a rotation to move old sibling up, recolor the old sibling and parent. The new sibling is always black (See the below diagram). This mainly converts the tree to black sibling case (by rotation) and leads to case (a) or (b). This case can be divided in two subcases.
……..**(i)** Left Case (s is left child of its parent). This is mirror of right right case shown in below diagram. We right rotate the parent p.
……..**(iii)** Right Case (s is right child of its parent). We left rotate the parent p.



3.3) If u is root, make it single black and return (Black height of complete tree reduces by 1).

References:

https://www.cs.purdue.edu/homes/ayg/CS251/slides/chap13c.pdf

Introduction to Algorithms 3rd Edition by Clifford Stein, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above