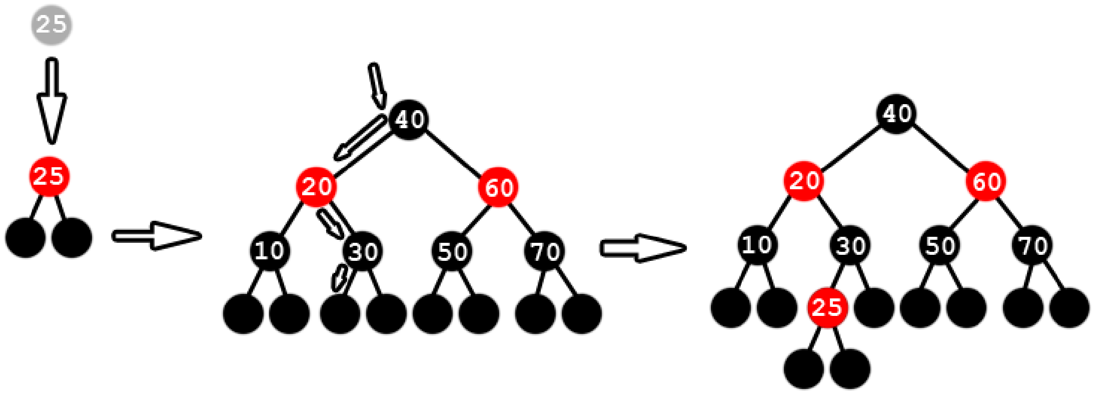
Ariel Stolerman

Red-Black Trees – Insertion, Deletion

Insertion:

Insertion:

- Find the correct leaf to insert new node instead of it
- Color node in red, and attach 2 black leafs to it
- Make sure RB-tree properties hold

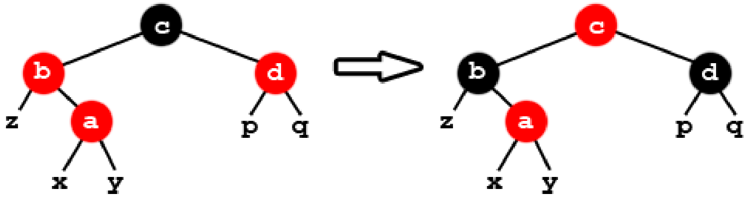


Correction:

When a violation is created for inserting **a** (cases 4-6 are symmetric):

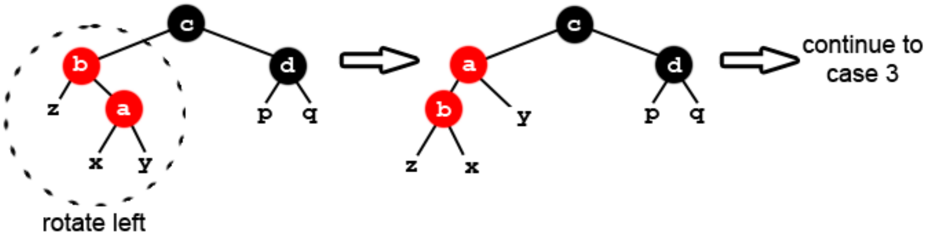
Case 1: a's uncle (d) is red:

- Color a's father (b) and uncle (d) black
- Color a's father (c) red



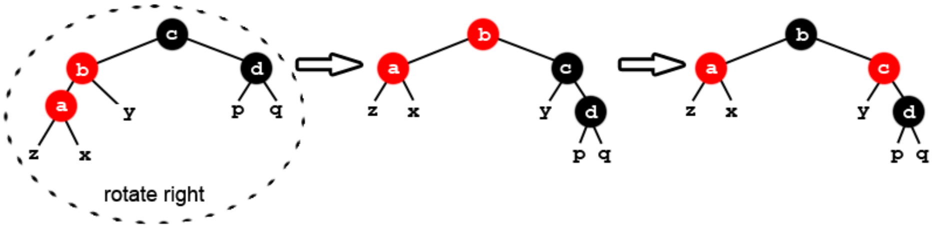
Case 2: a's uncle is black, a is a right child:

- Rotate left around a's father (b)
- Continue to case 3



Case 3: a's uncle is black, a is a left child:

- Rotate right around a's grandfather (c)
- Switch colors between a's father (b) and a's (new) sibling (c)

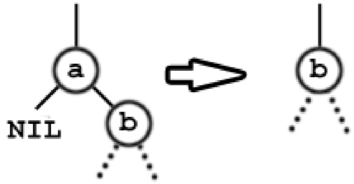


Deletion:

Deletion:

Deleting node **a** (disregard colors, fix later):

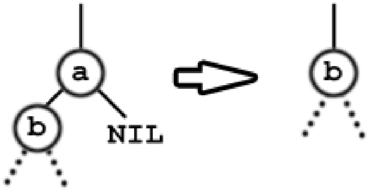
Case 1: a has no left child:



- Remove a and put its right child (b) instead

Note: if the red rule is now broken b and its new father (originally a's father), we can color b in black keeping the black height, since a was definitely black (as its father is red)

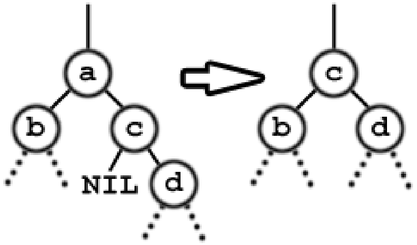
Case 2: a has no right child:



- Remove a and put its left child (b) instead

Note: same as case 1

Case 3: a has two children, a's successor (c) is its right child:

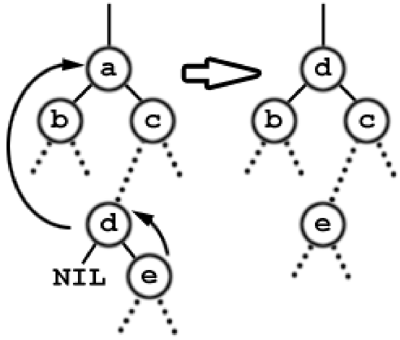


- Remove a and put its successor (c) instead
- Make a's left child (b) the successor's (c) left child

Notes:

- The successor always has no left child
- Moving the successor, we color it in a's color. If the successor was black, the child that replaced it (d) is colored in "extra" black, making it red-black or black-black. This is fixed in the correction.

Case 4: a has two children, a's successor (d) is not its child:



- Put the successor's (d) left child (e) instead of it
- Remove a and put its successor (d) instead of it, making a's children (b, c) its new children

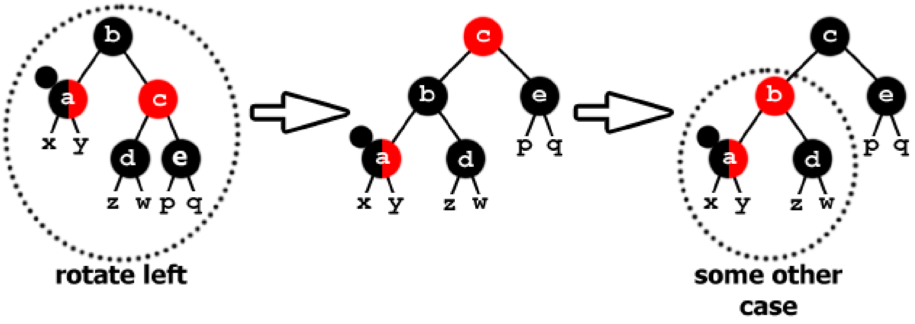
Notes: same as case 3

Correction:

Node **a** has an extra black (◐ denotes a node colored either black or red; cases 5-8 are symmetric):

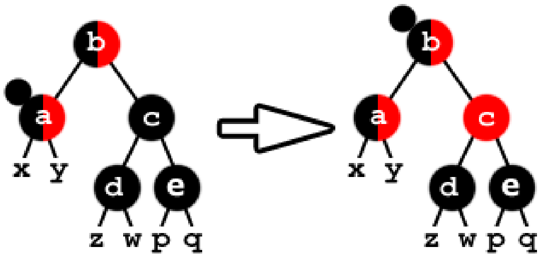
Case 1: a's sibling (c) is red:

- Rotate left around a's father (b)
- Switch colors between a's father (b) and grandfather (c)
- Continue to the next case with the sub-tree rooted at b



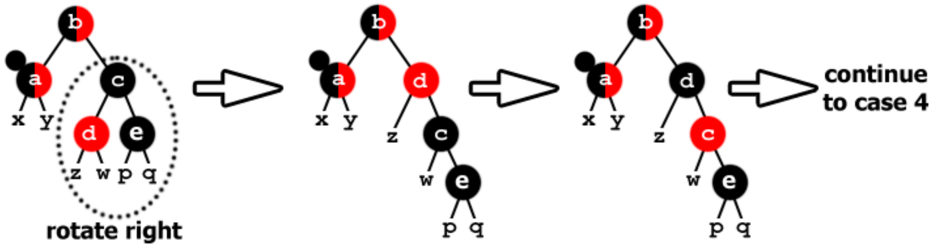
Case 2: a's sibling (c) and its children (d, e) are black:

- Take one black from a and its sibling c and move it up (leaving a with one black and c – red)
- The problem is moved up – continue solving it



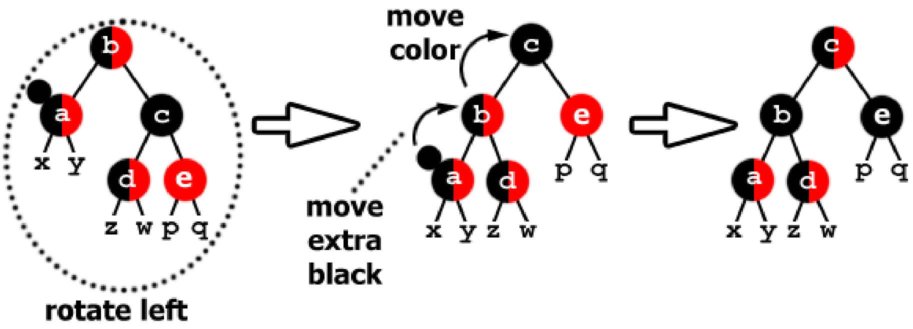
Case 3: a's sibling (c) is black, with left child (d) red and right child (e) black:

- Rotate right around a's sibling (c)
- Switch colors between a's new and old siblings (d, c)
- Continue to case 4



Case 4: a's sibling (c) is black, with right child (e) red:

- Rotate left around a's father (b)
- Color a's new grandfather (c) with a's father's (b) color
- Color a's father (b) with a's extra black, making a singly-colored
- Color a's uncle (e) black (originally a's right "nephew")



At the end all leafs (x-q) have the same black height as at the beginning, no node is double-colored and no violations of the red-black properties occur.