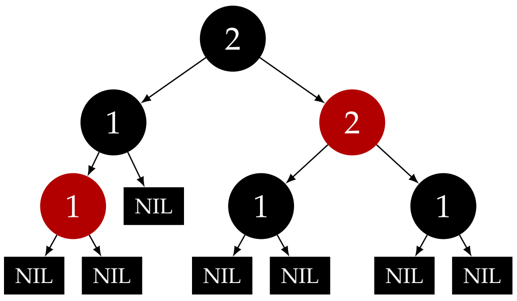
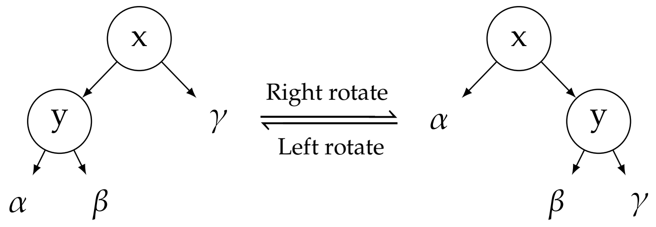
1 RB Tree

1.1 Black Height



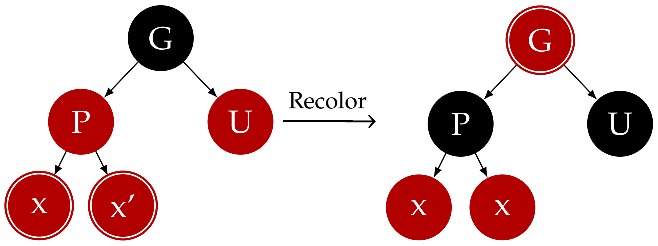
1.2 Rotation



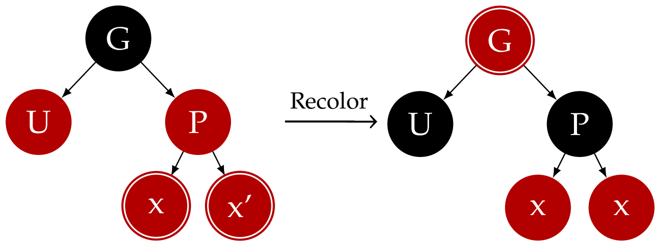
1.3 Insert Fixup

Loop invariant: Always red violation.

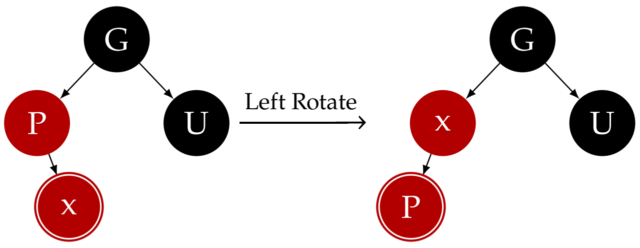
Case 1: **U**ncle is red (My **P**arent is left child)



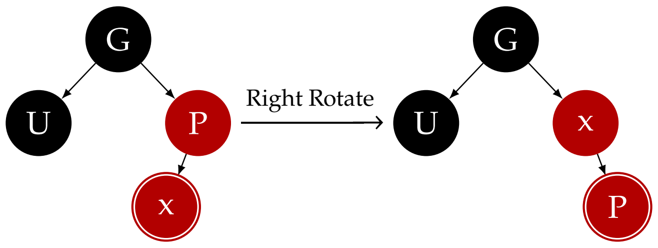
Case 4: **U**ncle is red (My **P**arent is right child)



Case 2: **U**ncle is black, I am his near nephew

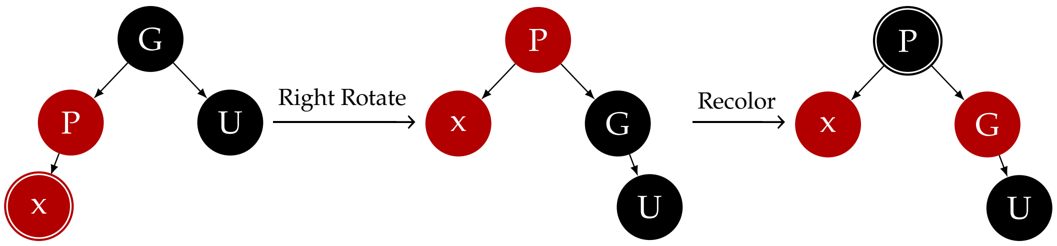


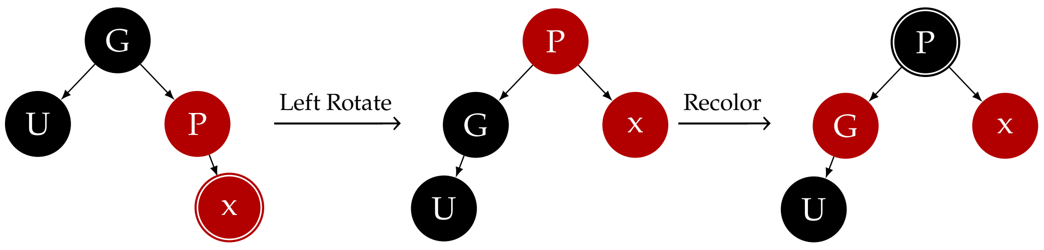
Case 5: **U**ncle is black, I am his near nephew



Case 3: **U**ncle is black, I am his distant nephew (my **P**arent is left child)

Case 6: **U**ncle is black, I am right child (my **P**arent is right child)





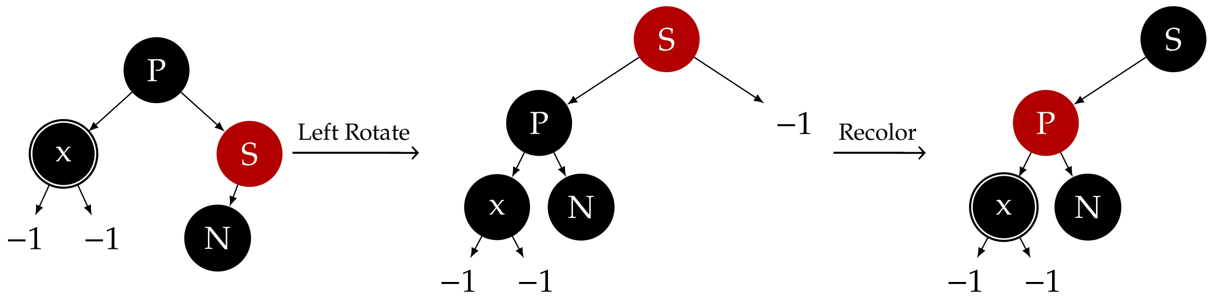
1.4 Delete

If target node has only one child, then move it up and call fixup. Otherwise, let current be z, next node be y, y has only a child x. We move y's key to z, and remove y (child moves up). Call fixup on y.

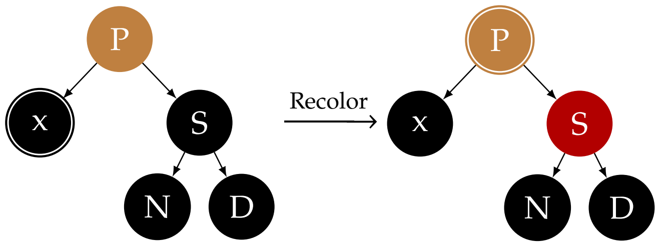
1.5 Delete Fixup

Loop invariant: Subtree of current node always missing one black height.
Case 0: I am red or root — color myself to black, and terminate (fixup only need for black)

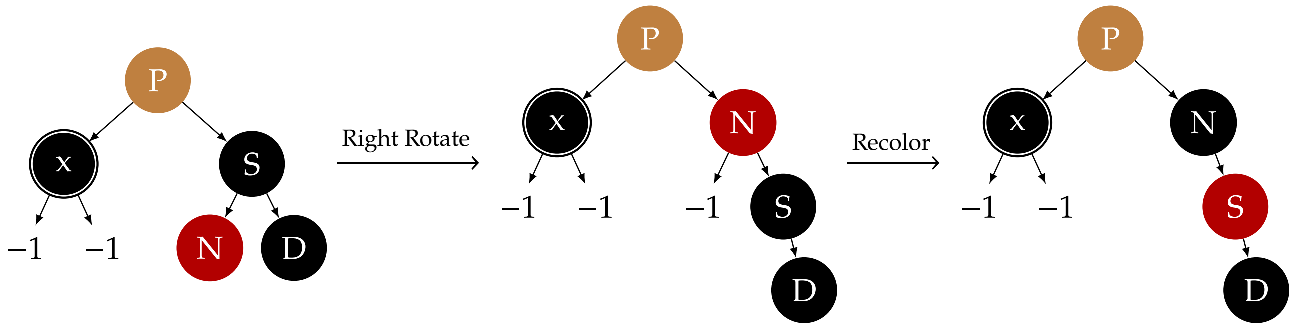
Case 1: My **S**ibling is red



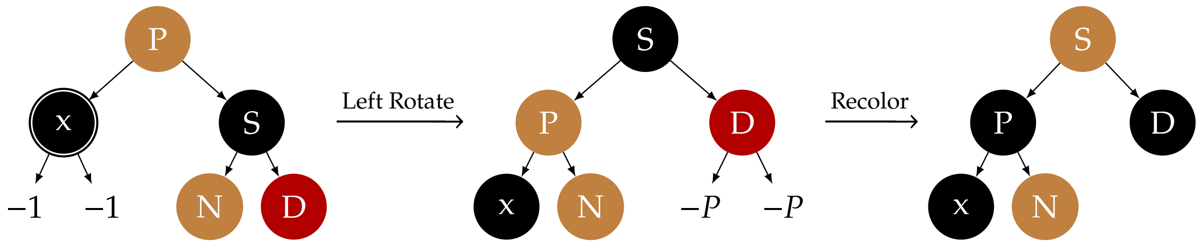
Case 2: My **S**ibling is black, and both its children are black



Case 3: My **S**ibling is black, and the **D**istant child is black



Case 4: My **S**ibling is black, and the **D**istant child is red


P
x
S
N
D
−1
−1
Left Rotate
S
P
D
x
N
−P
−P
Recolor
S
P
D
x
N