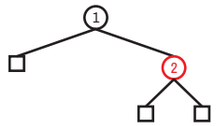
int[] testData = {1, 2, 3, 4, 5, 6};

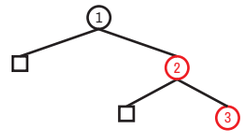
by Vladimir V. KUCHINOV

while root can't be a RED node, so...

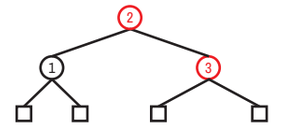
inserting 2



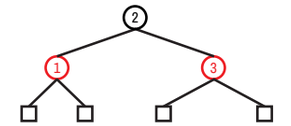
no violations, inserting 3



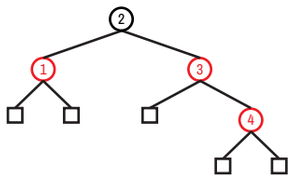
RED node (2) should have two BLACK node, so going for Case 3:
left rotate around grandparent



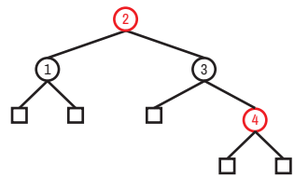
RED node (2) still violating rules, as well as it should have two BLACK nodes, so...



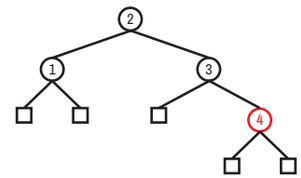
valid, inserting 4



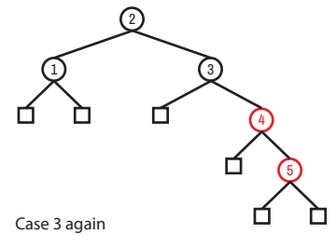
RED node 3 has RED child, it's a violation, so we are going to use Case 1:
changing colors of grandparent, parent and an uncle (RED 1)



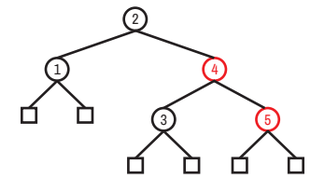
RED node 2 could not be a root node, changing it to BLACK



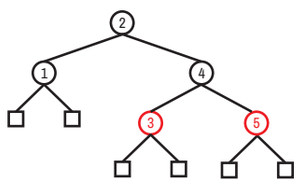
valid, inserting 5



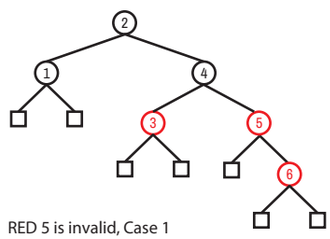
Case 3 again



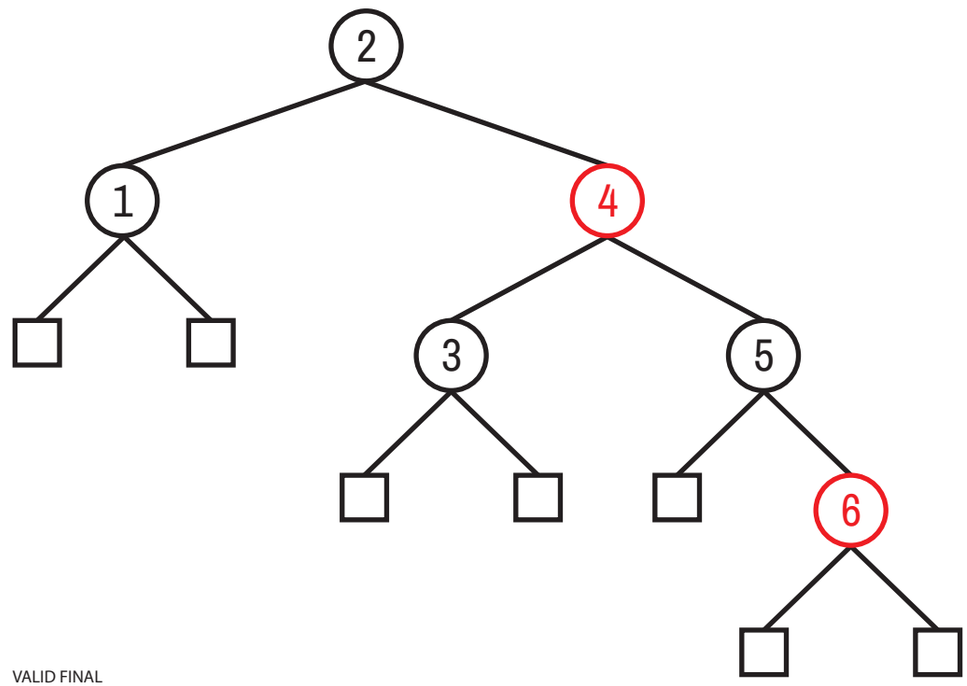
RED node 4 couldn't have RED, Case 1



valid, time for the last insert of 6

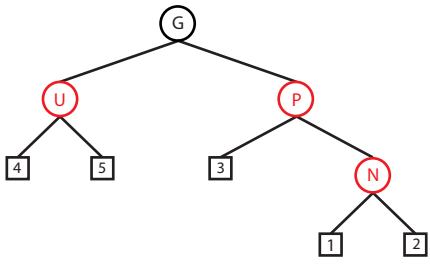


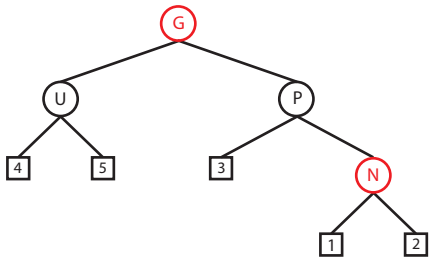
RED 5 is invalid, Case 1



VALID FINAL

CASE 1

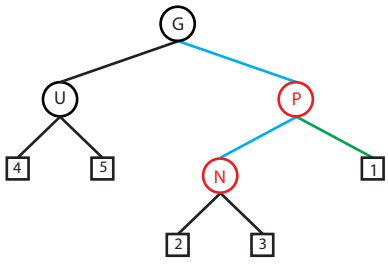


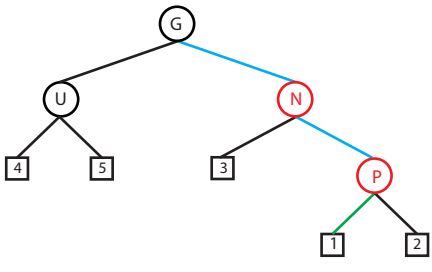


RED 3 has a violation, couldn't have RED child, only two BLACKs.
CASE 1 only applies when uncle node (RED 1) is RED.

Changing colors of those three nodes: parent, grandparent and an uncle.*

CASE 2



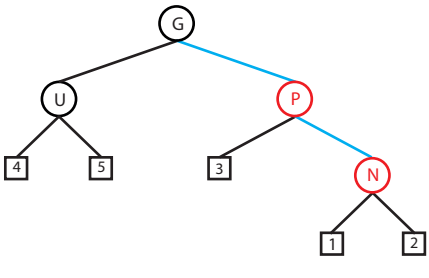


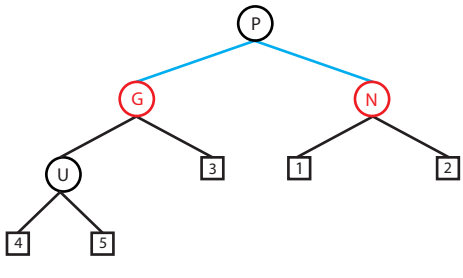
RED 3 has a violation, couldn't have RED child, only two BLACK.

CASE 2 only applies when uncle node (BLACK 1) is BLACK,and it's a right child of the left child / left child of the right child.

Rotate around parent node.

CASE 3





RED 3 has a violation, couldn't have RED child, only two BLACK.

CASE 3 only applies when uncle node (BLACK 1) is BLACK,and it's a right child of the right child / left child of the left child.

Rotate around grandparent node this time.

** the result is still invalid, however it is a matter of other future manipulations.*