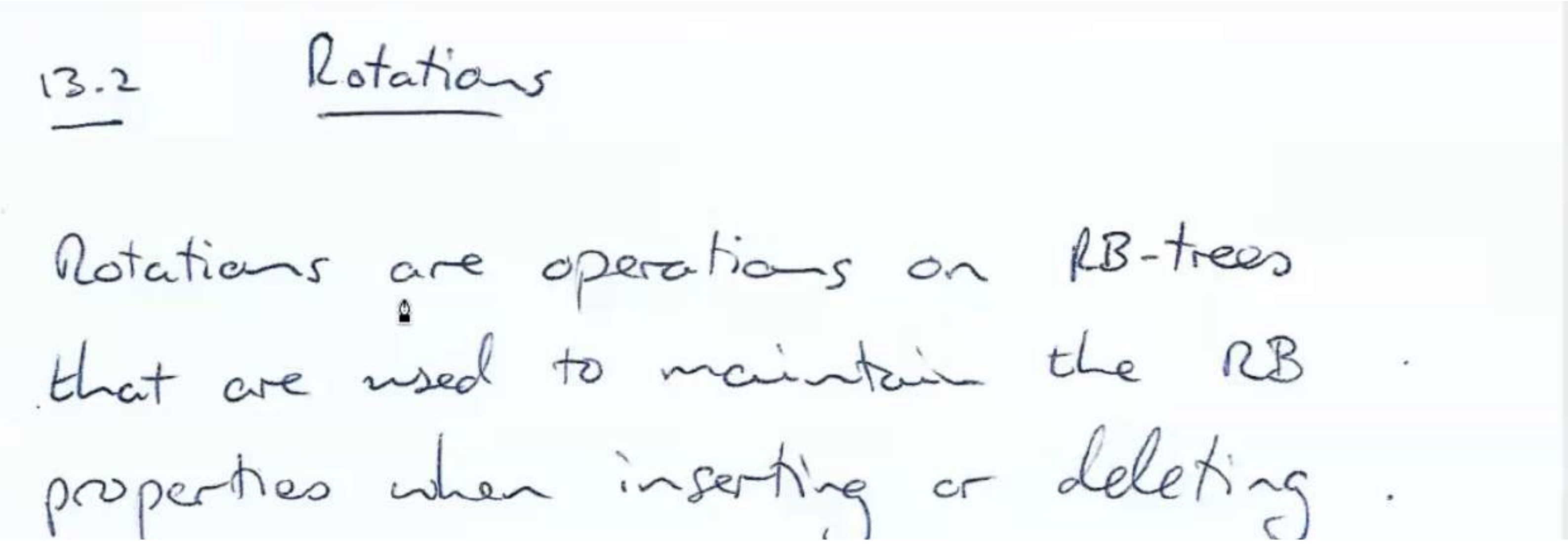
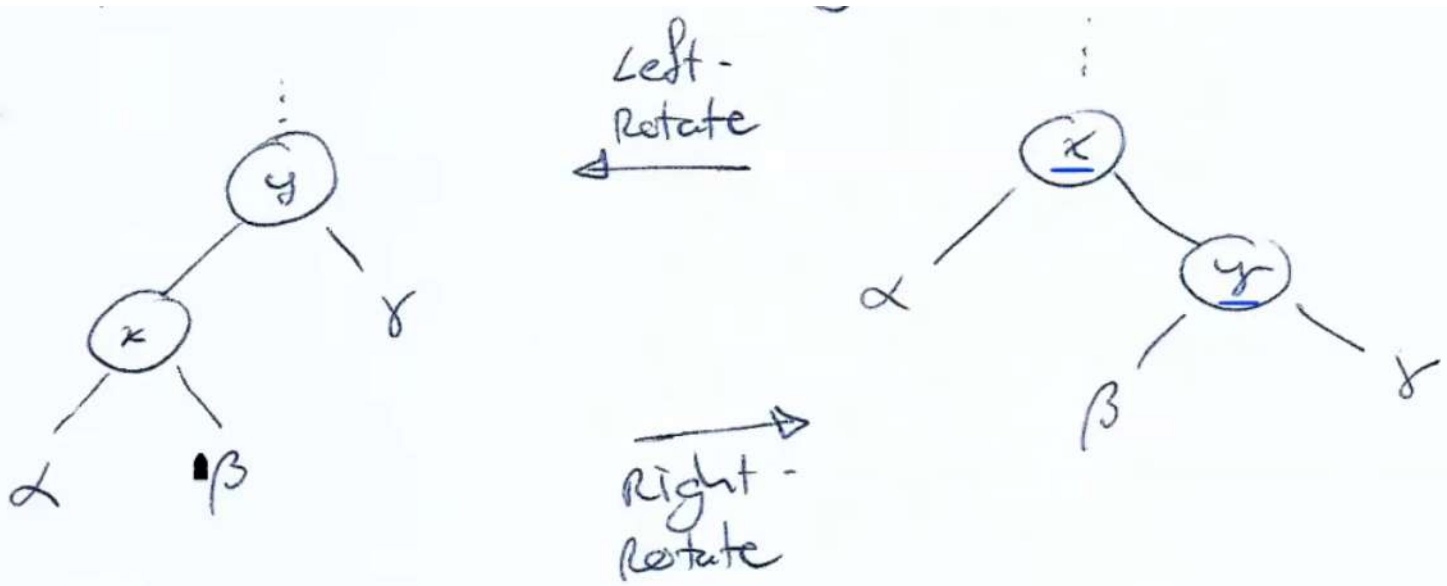
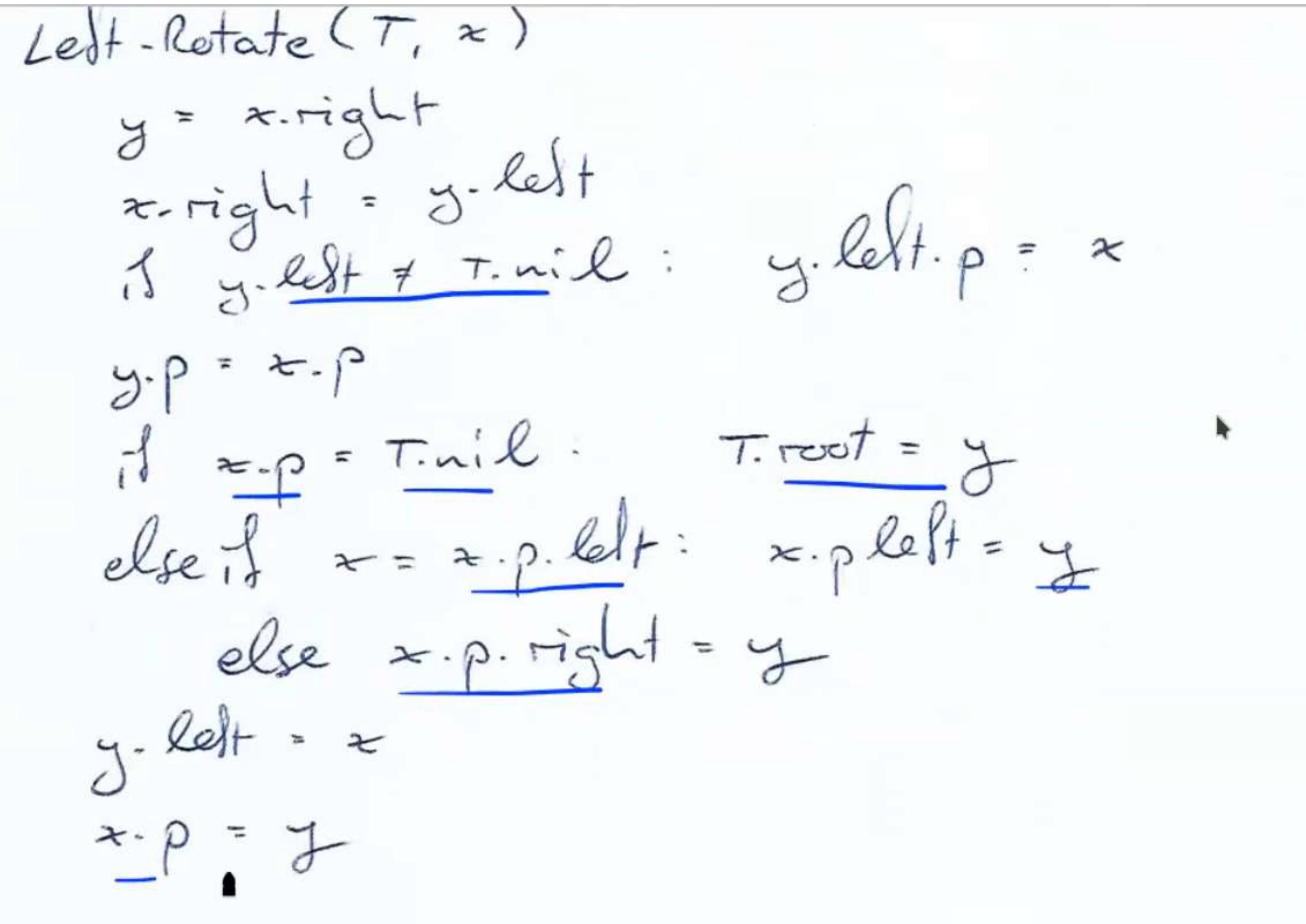


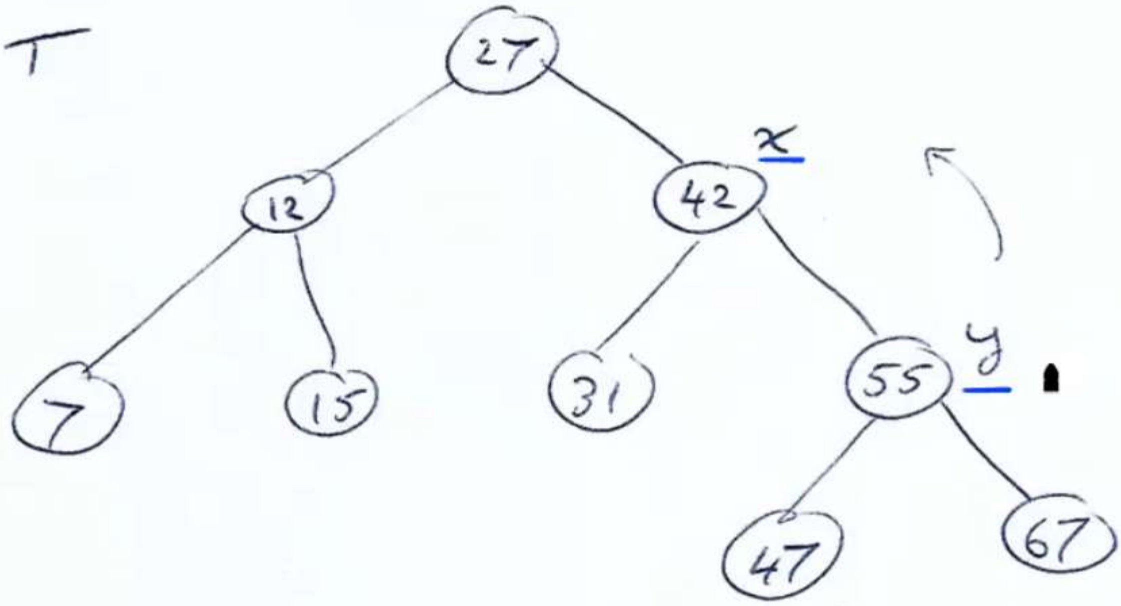
13.2 Rotations

Rotations are operations on RB-trees that are used to maintain the RB properties when inserting or deleting.

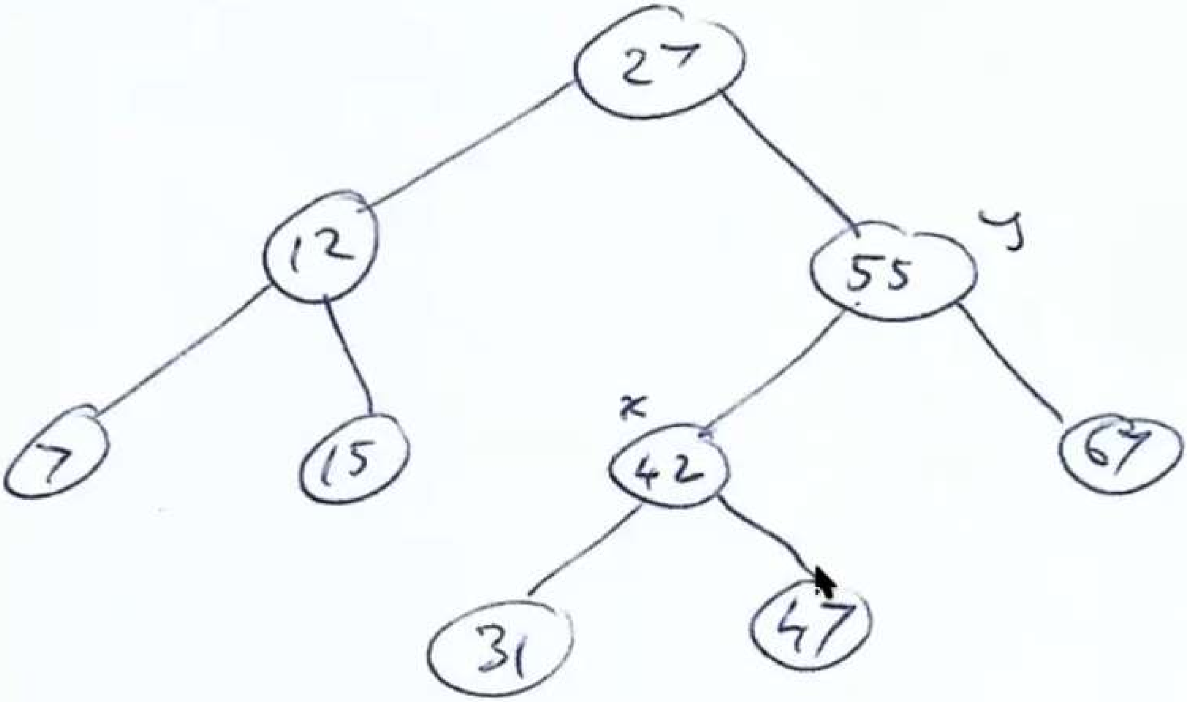

y
x
α
β
γ
Left-Rotate
Right-Rotate
x
y
α
β
γ

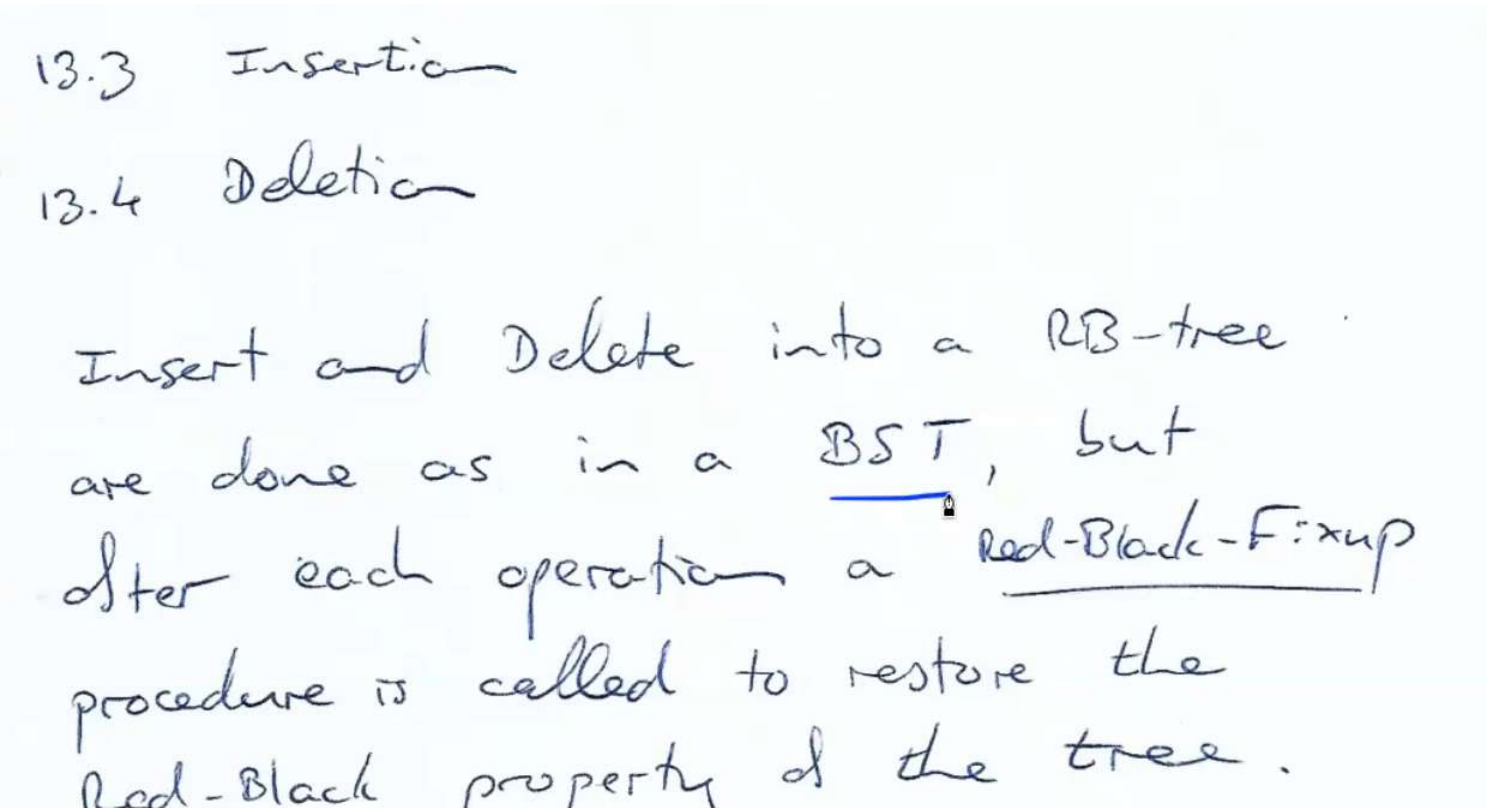


```
Left-Rotate(T, x)
    y = x.right
    x.right = y.left
    if y.left ≠ T.nil :   y.left.p = x
    y.p = x.p
    if x.p = T.nil .   T.root = y
    else if x == x.p.left :   x.p left = y
        else x.p.right = y
    y.left = x
    x.p = y
```


T
27
12
42
x
7
15
31
55
y
47
67

Left-Rotate(T, x)

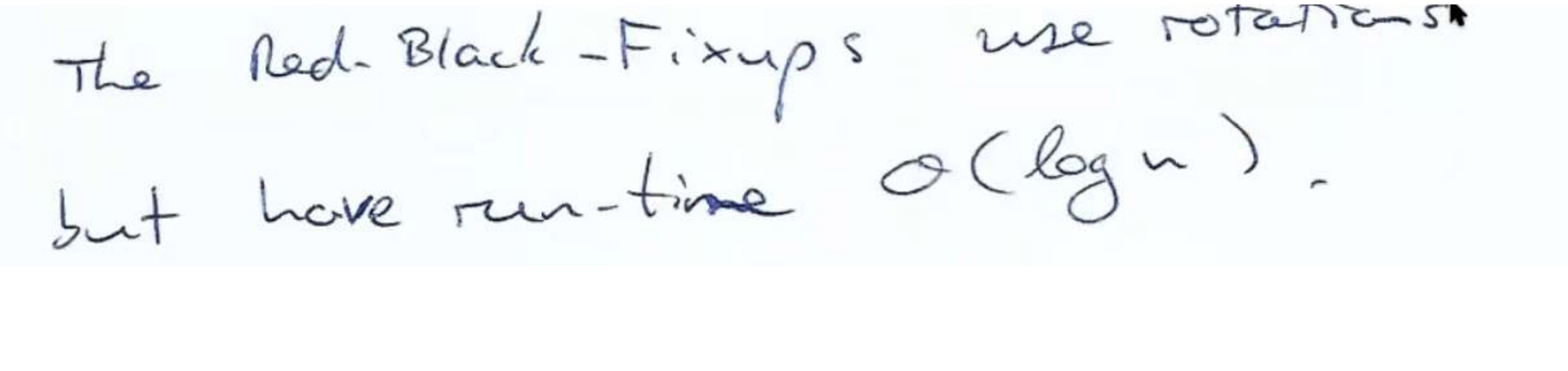




13.3 Insertion

13.4 Deletion

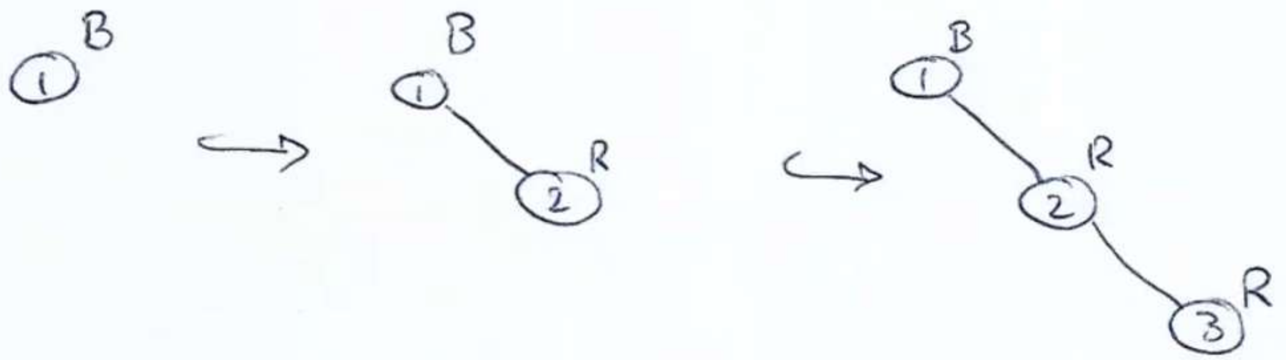
Insert and Delete into a RB-tree are done as in a BST, but after each operation a Red-Black-Fixup procedure is called to restore the Red-Black property of the tree.



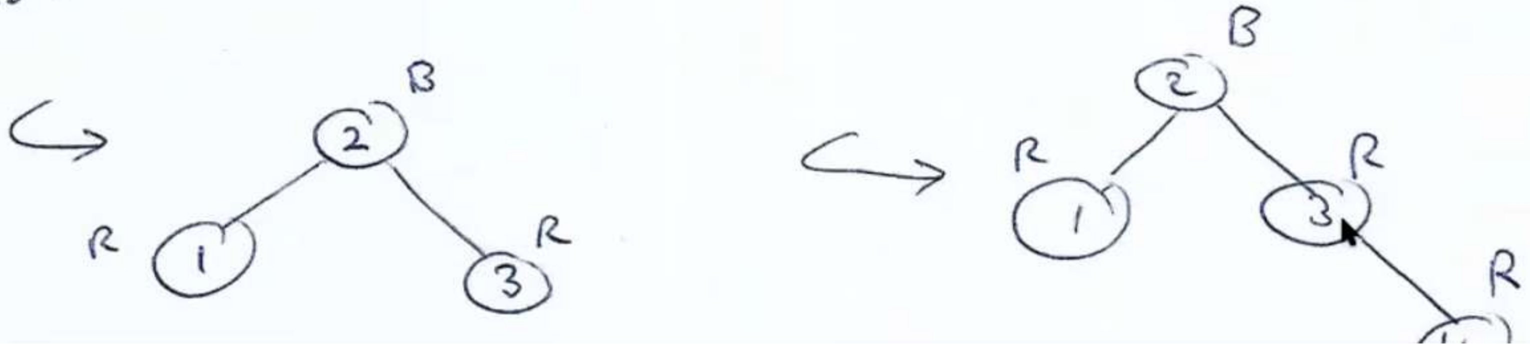
The Red-Black-Fixups use rotations

but have run-time $O(\log n)$.

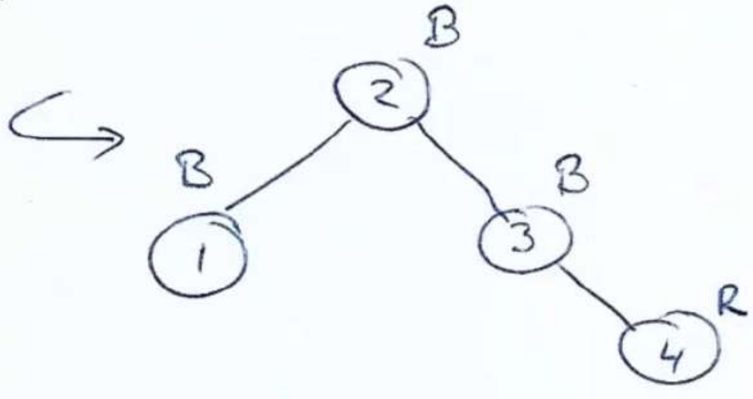
Build RB-tree with keys 1, 2, 3, 4, 5

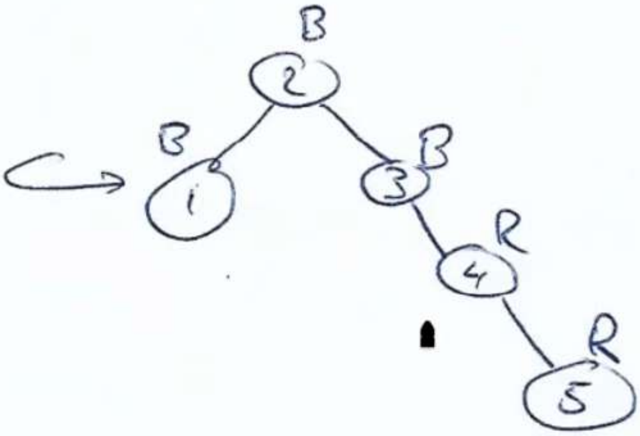


RB-Insert-fixup

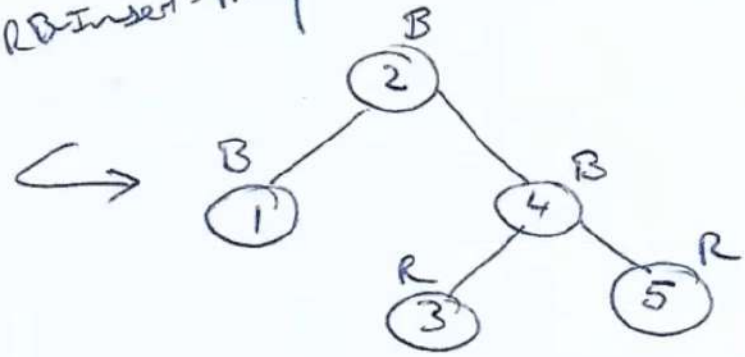


RB-Insert-fixup





RB-Insert-fixup


B
2
B
1
B
4
R
3
R
5