Tutorial 7 : Red-Black Tree

Properties :

1. Every node is either red or black
2. The root is black
3. Every leaf (NIL) is black
4. If a node is red, then both its children are black
5. For each node, all paths from the node to descendant leaves contains the same number of black nodes.

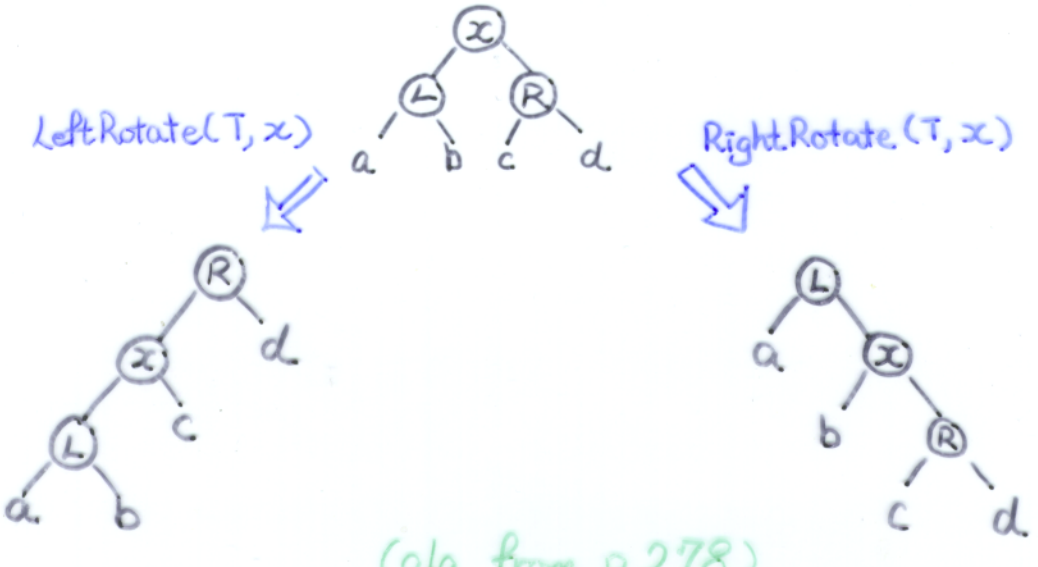
Black-height

The number of black nodes on any path from, but not including a node x, down to a leaf, denoted by $bh(x)$

The tree height of RB tree is at most $2\lg(n+1)$ where n = number of internal nodes.

⇒ Search, Minimum, Maximum, Successor and Predecessor takes $O(h) = O(\lg n)$ time.

Rotation

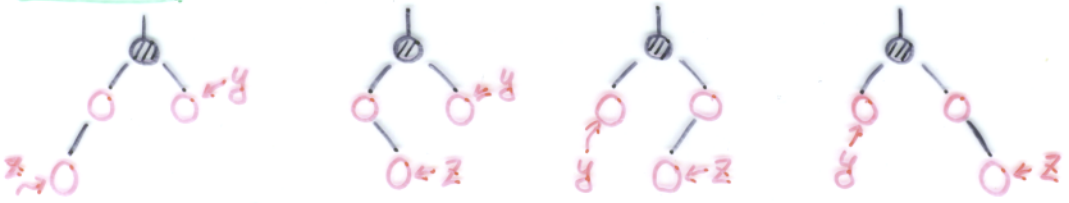


(alg from p.278)

while color(p(z)) = RED :

Insertion : fix up

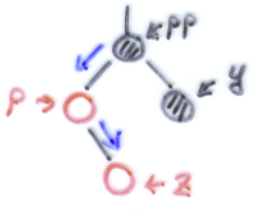
Case 1 : y is RED



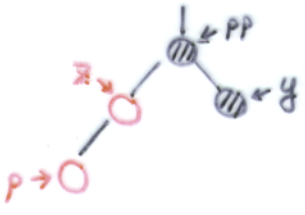
↳ re-color

↝ case 1, 2, 3
or done
or reach the root.

Case 2 :



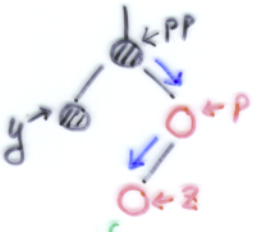
↳ LeftRotate(T, p)



↳

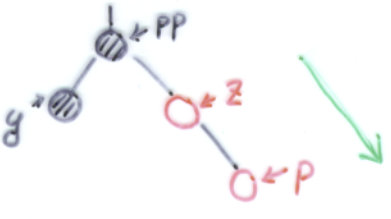
↳ case 3

Case 2



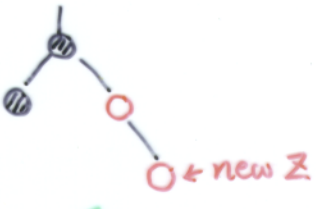
z = left(p(z))
p(z) = right(p(p(z)))

↳ RightRotate(T, p)



ALIGNED!

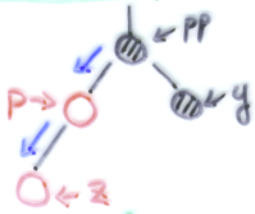
↳



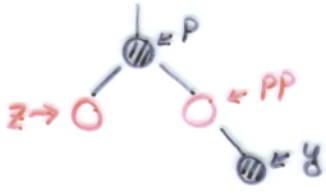
↳ case 3

Insertion (cont)

case 3:

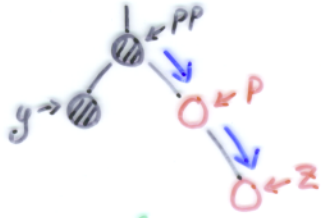


Right Rotate (T, pp)
+ re-color

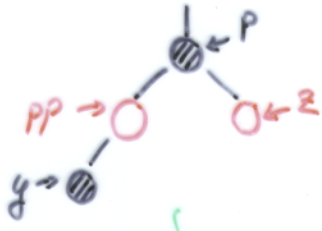


done!

case 3:



Left Rotate (T, pp)
+ re-color



done!

(alg from p. 280)

Remark: If parent of z is black in color, no fix need to be done.
If z is root of the Tree, color it black and then done.

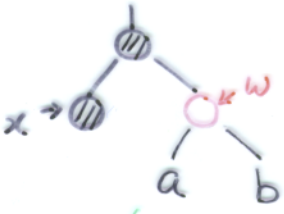
RB-DELETE(T, z) applied

x is the only child of y, or is the sentinel nil[T]

x is the orphan.

Deletion Fix-up

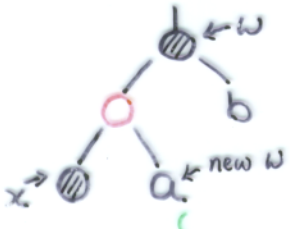
case 1:



↓

LeftRotate(T, p[x])

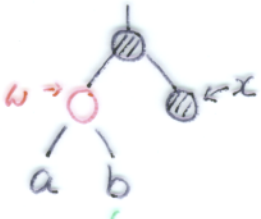
+ recolor



↓

case 2, 3, 4

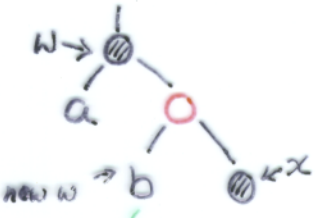
case 1:



↓

RightRotate(T, p[x])

+ re-color



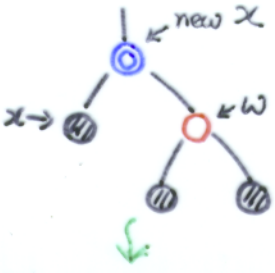
↓

case 2, 3, 4

case 2:

↓

Re-color



↓

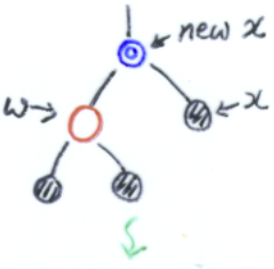
case 1, 2, 3, 4

or ◎ is red.

case 2:

↓

Re-color



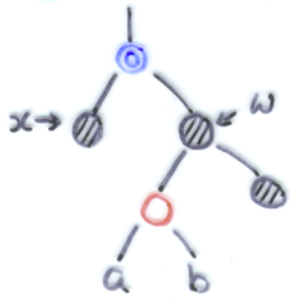
↓

case 1, 2, 3, 4

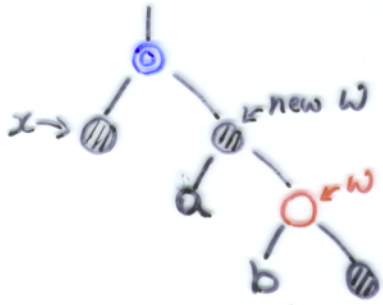
or ◎ is red.

Deletion (cont)

case 3:

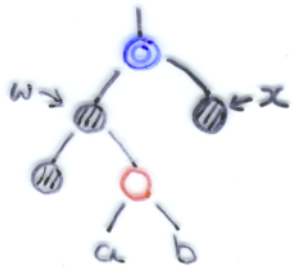


↓ Right Rotate (T, w)
+ re-color

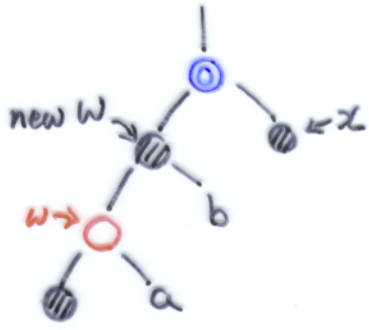


↓ case 4

case 3:

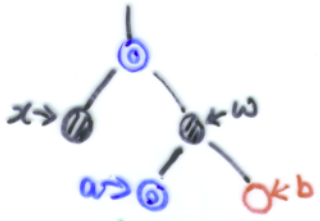


↓ Left Rotate (T, w)
+ re-color

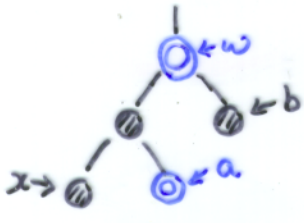


↓ case 4

case 4:



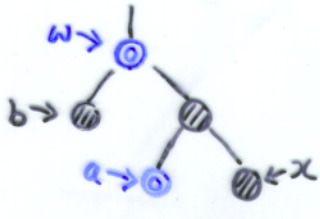
↓ Left Rotate (T, p[x])
+ recolor



↓ done!

case 4:

↓ Right Rotate (T, p[x])
+ recolor



↓ done!

(alg from p.289)

Deletion (cont)

Remarks: · If the node removed is red in color, no fix needed to be done.

· If the node x is red in color, color it black and then done.

Reasons:

· If the removed node is red, then
  · no bh will be changed
  · no nodes adjacent to red are red.
  · y is not the root.

· If the node x is red, then.
  · y is not red $\Rightarrow$ bh will be decreased by 1.

Recap:

• The while loop in lines 1-22 of p289 moves the extra black up the tree until:

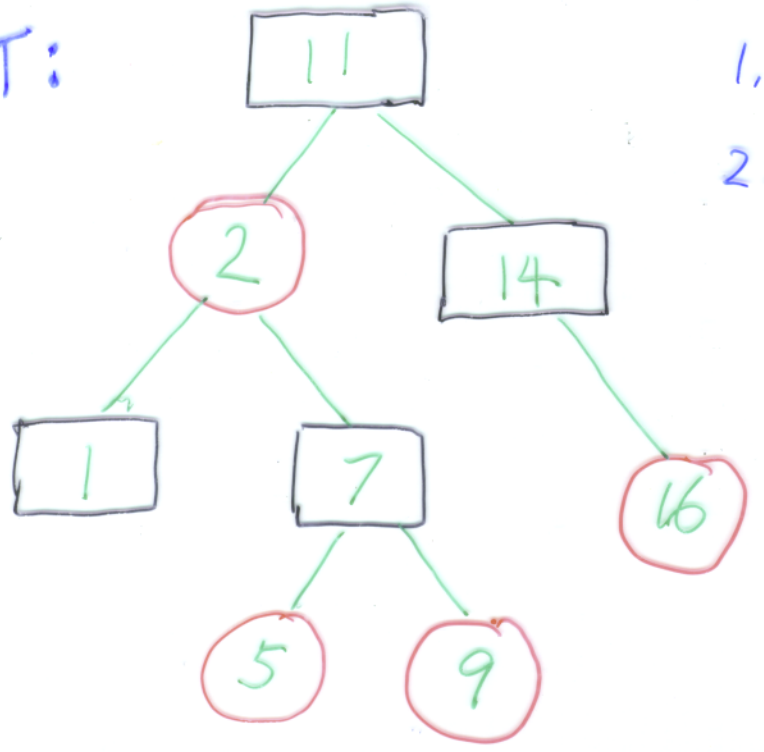
1. x points to a red-and-black node.
2. x points to the root
3. suitable rotations and recolorings can be performed.

• Transformation preserves bh of subtrees $\alpha, \beta, \ldots, \zeta$

EX: 13.4-3

1. Build a RB tree T by successively inserting $\langle 41, 38, 31, 12, 9, 8 \rangle$ into an initially empty tree.
2. Show the RB trees by deleting $\langle 8, 12, 19, 31, 38, 41 \rangle$ from T.

T:



1. RB-INSERT(T, 4)
2. RB-INSERT(T, 15)

$T_0 = \phi$:

for each $x \in \langle 11, 2, 14, 1, 7, 5, 9, 16 \rangle$

    RB-INSERT(T_0, x)

$T_0 = T$?

How about $\langle 1, 2, 5, 7, 9, 11, 14, 16 \rangle$?