Red-Black Trees

1) It is a binary search tree
2) Every node is colored either red (R) or Black (B).
3) root is colored B
4) no red node can have red children
5) if a node does not have left or right child, add an external node. External nodes are not colored.
6) for all the external nodes, the path from external node to root contains the same number of black nodes.

Insert Operation

T - Red Black tree

Insert (T, z)

→ insert as you insert in BST

Problem :- Tree no longer be Red Black.

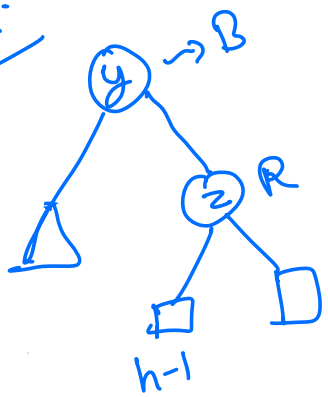
* Color of node z is R.

If T was an empty tree

→ recolor z as B

Suppose T was not empty.

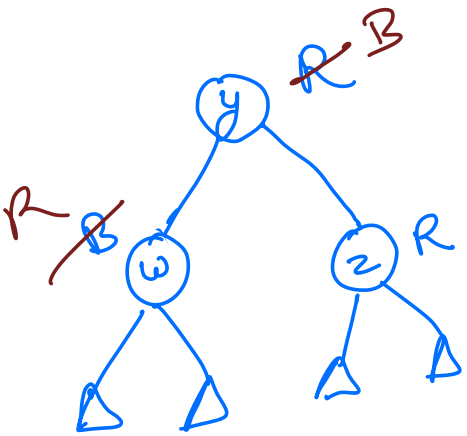
Case1:



p(z) → parent of z

If p(z) is Black, it is still a Red Black tree.

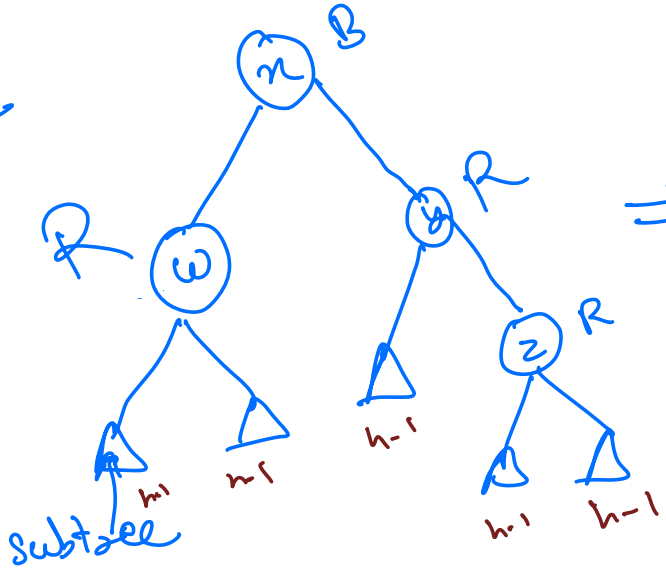
p(z) is Red.
Double Red problem.
Color of w is Black

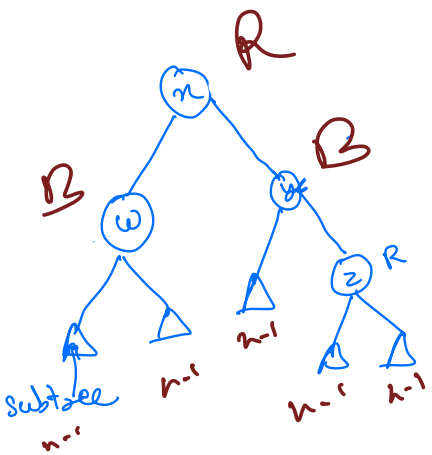


Change the color of y to B & w to R — Not correct!

Case2:

(i)





If x is a root, then it is not a Red Black tree.

Make x black.

Suppose x is not root.

if $p(x)$ is B, then it is a Red Black tree.

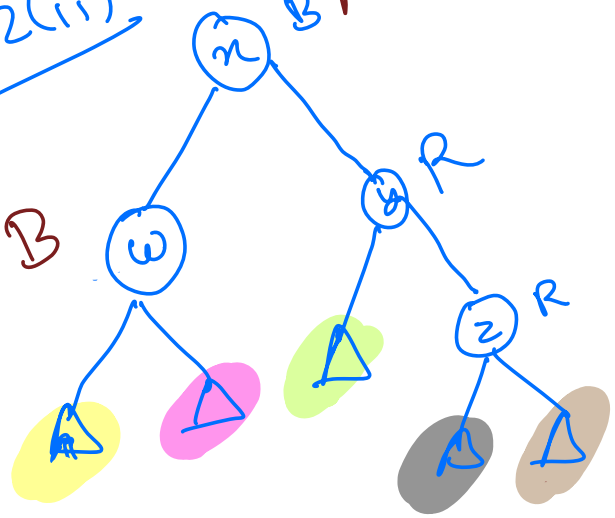
Otherwise, not.

If $p(x)$ is R, then Double Red has moved up.

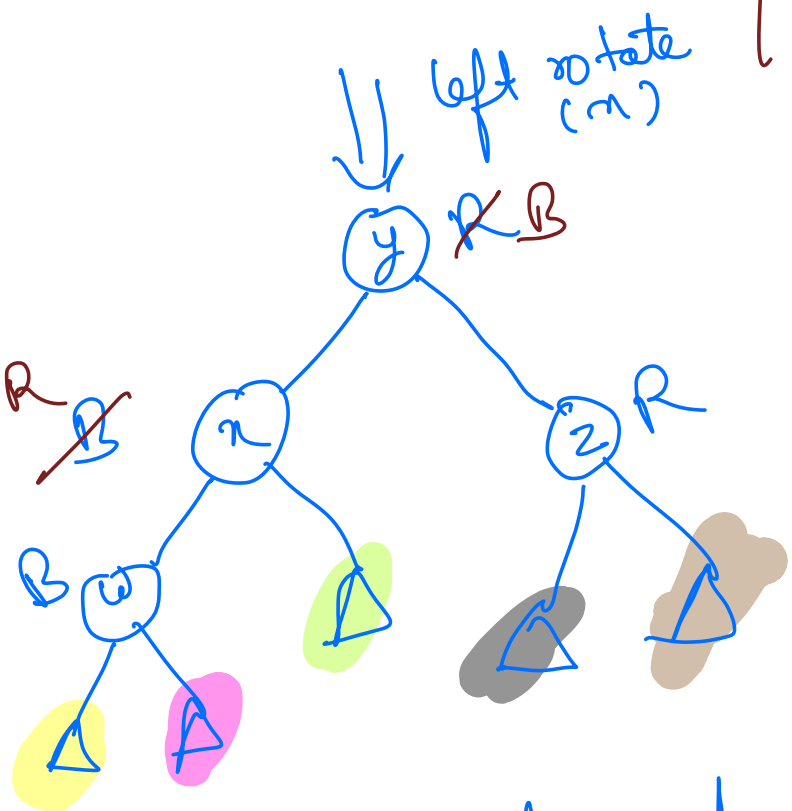
But we can move up only $O(\log n)$ times.

Color of w is Black.

Case 2(ii)



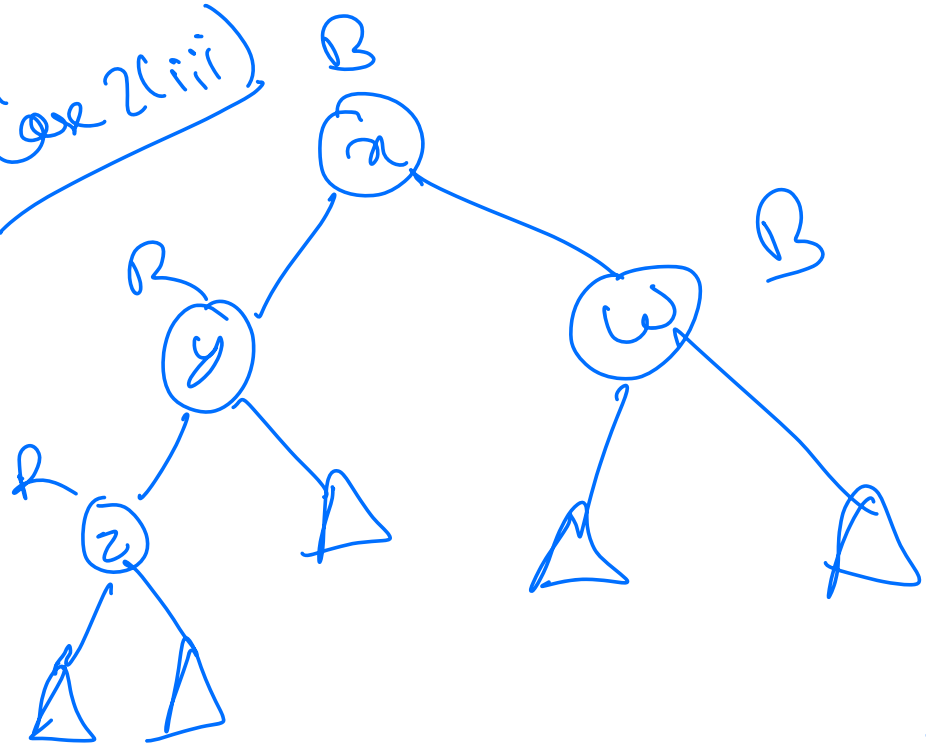
z - Red
p(z) - Red.
p(p(z)) - Black
sibling (p(z)) - Black
left_rotate (x)



It is still a BST

Change color of y to B & x to R.

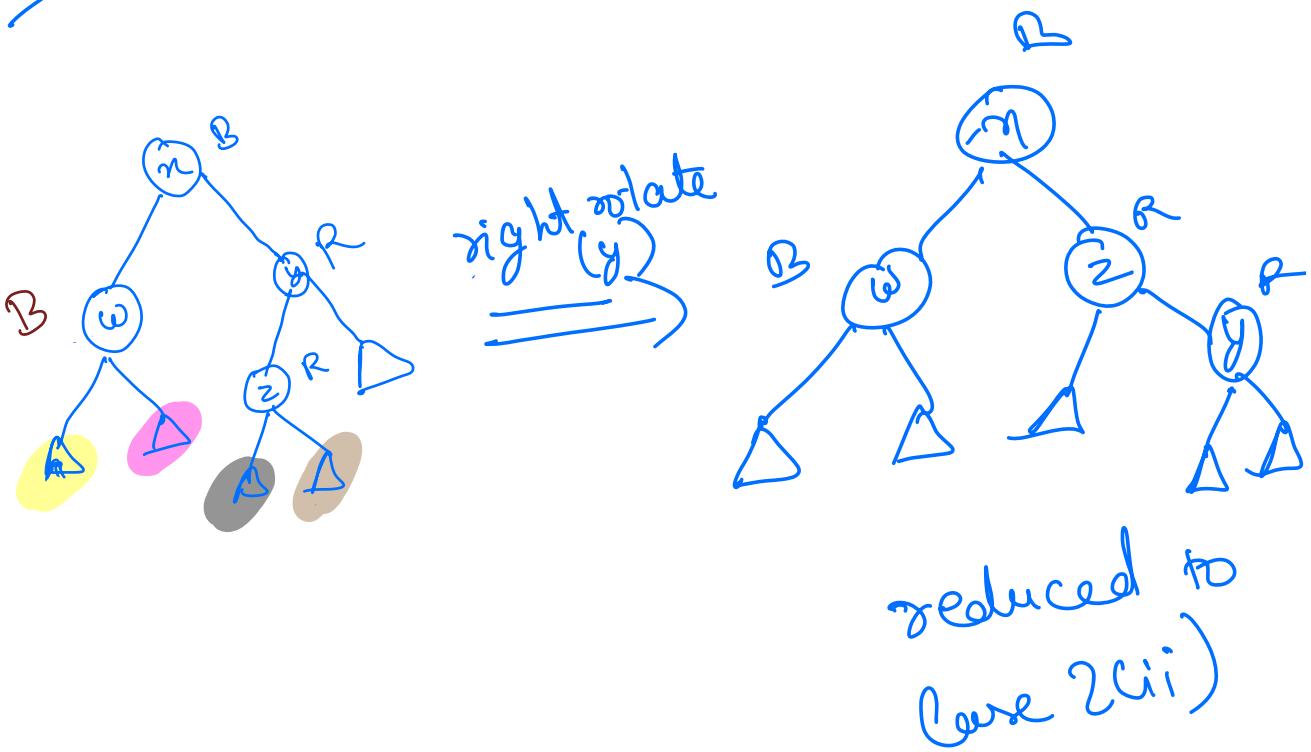
Case 2(iii)



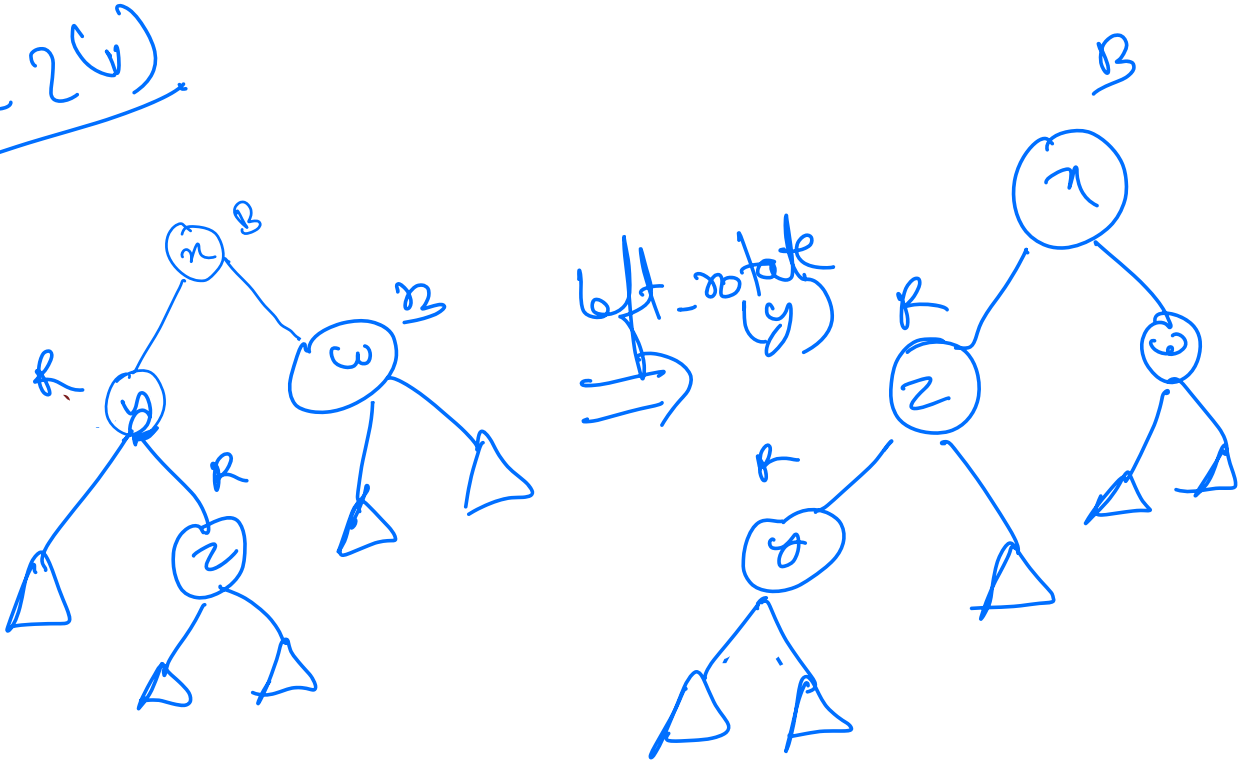
right_rotate(x)

change color of y to B & x to R.

Case 2(iv)


B
x
R
w
y
R
z
R
right rotate (y)
B
y
R
x
R
w
z
reduced to Case 2(ii)

Case 2(v)


B
x
R
w
y
R
z
left-rotate (y)
B
y
R
x
w
R
z

reduced to 2 (iii) Case.

Insertion is $O(\log n)$ in Red Black Tree