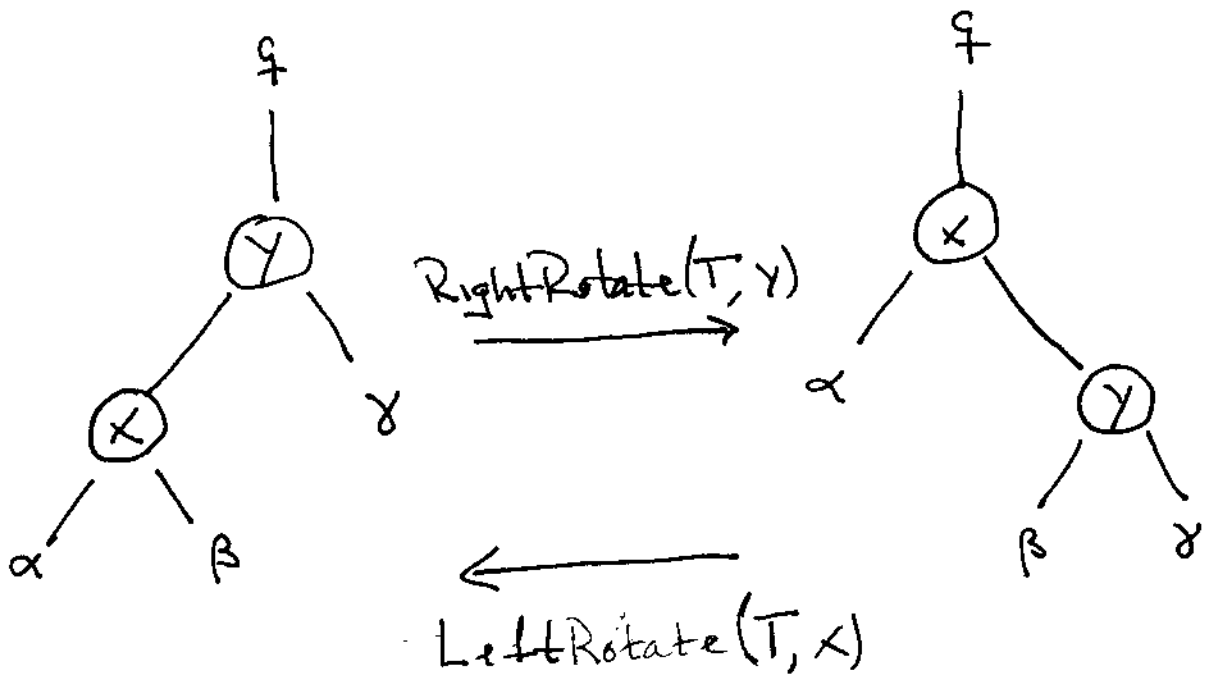
CSE 101 5-14-20

Pas: ext another day

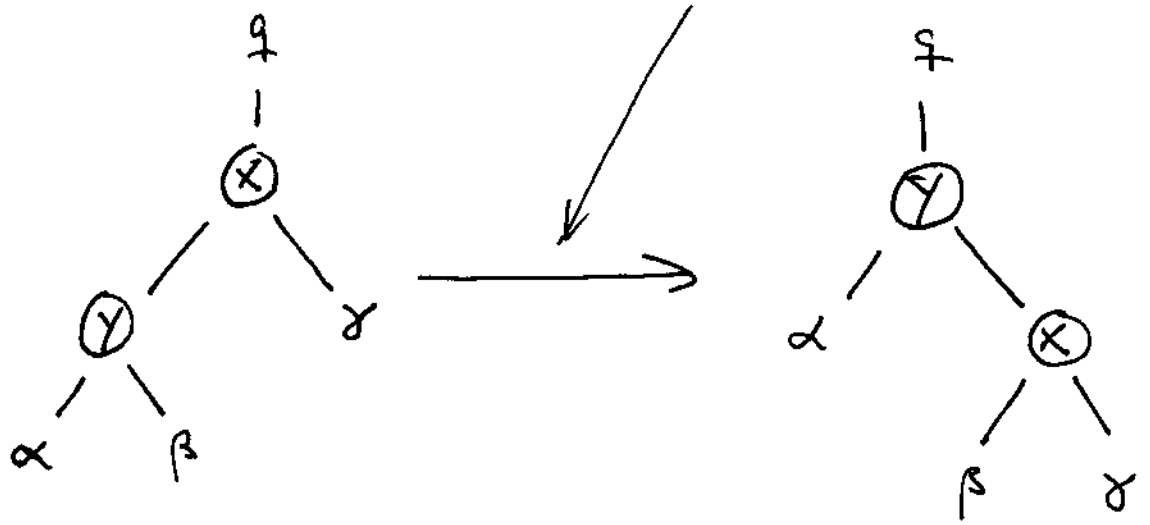
## 13.2 Rotations

LeftRotate(T, x) } inverses
RightRotate(T, x)



Book describes LeftRotate(T, x)

will discuss RightRotate(T, x)




note: Rotation does Preserve BST Properties

$$Keys[\alpha] \le Key[y] \le Keys[\beta] \le Key[x] \le Keys[\gamma]$$

note: Preconditions for

RightRotate(T, x) : left[x] ≠ nil

LeftRotate(T, x) : right[x] ≠ nil

Summarize RightRotate(T, x):

- fix B

  left[x] = right[y]

  Parent[right[y]] = x

- fix Parent q:

  Parent[y] = Parent[x]

  Parent[x] (left or right child) = y

- fix relationship between x & y

  right[y] = x

  Parent[x] = y

Runtime: $\Theta(1)$

# 13.3 Insertion

Recall RBT

1. all nodes R or B
2. nil (leaves) are B
3. root is B
4. R have only B children
5. bh(x) is well-defined for all $x \in T$

BST insert:

Preserves 1, 2, 3

if new node is colored RED,

RBT #s also preserved.

(if insertion is at root, turn it BLACK $\Rightarrow$ RBT #s still satisfied)

Problem: RBT #4 may be violated

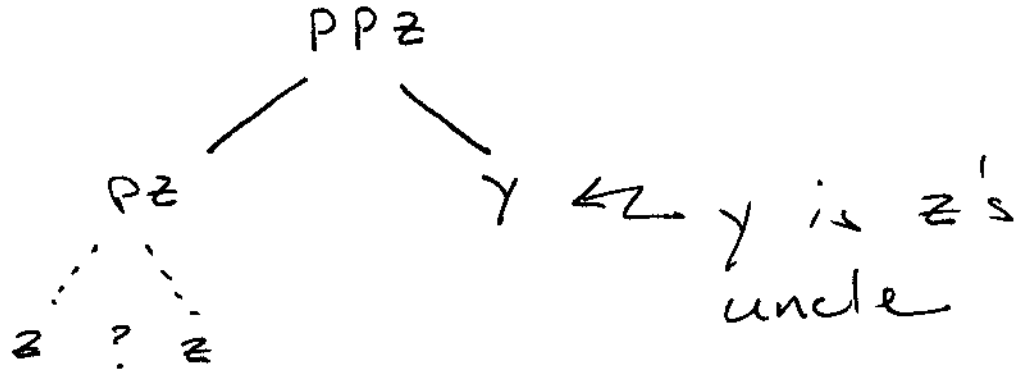
RB_Insert(T, z)

RB_InsertFixUp(T, z)

How does RB_InsertFixUp(T, z) work?

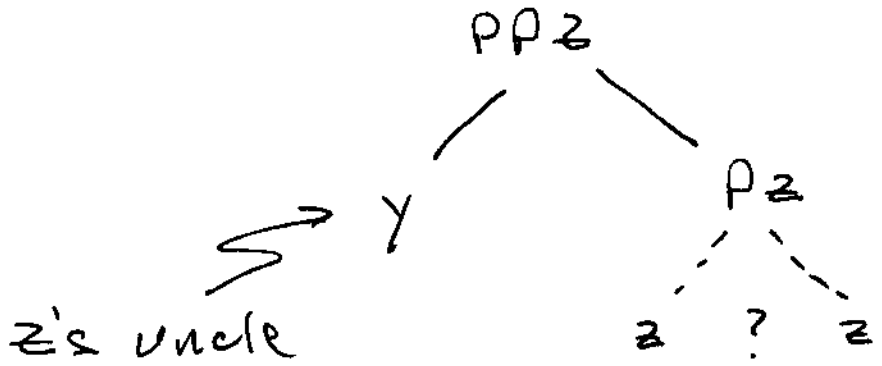
if color[z] is Red, have violation of RBT #4. Two cases

- p[z] is left child of its parent



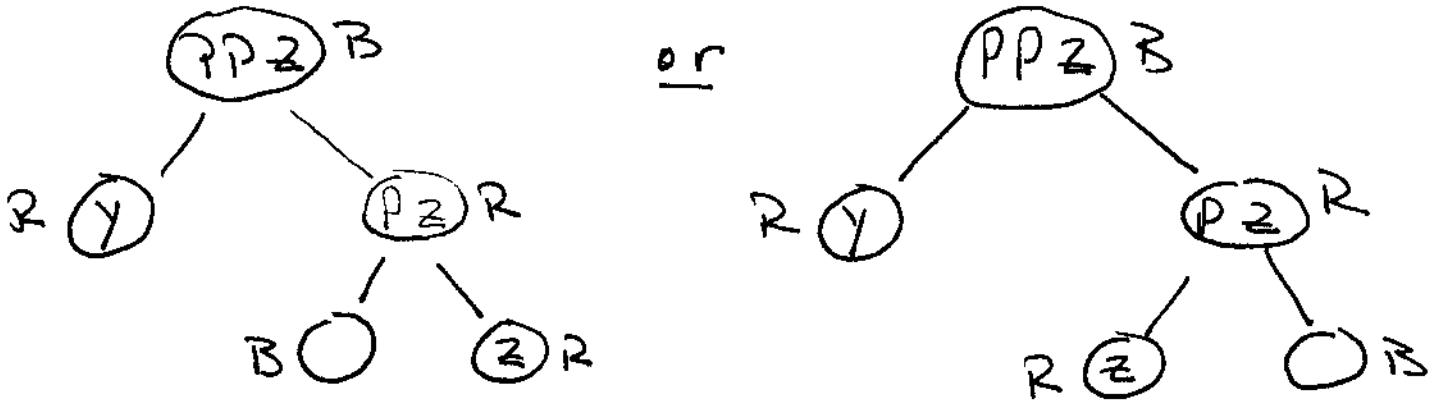

subcases 1, 2, 3

- p[z] is right child of its parent

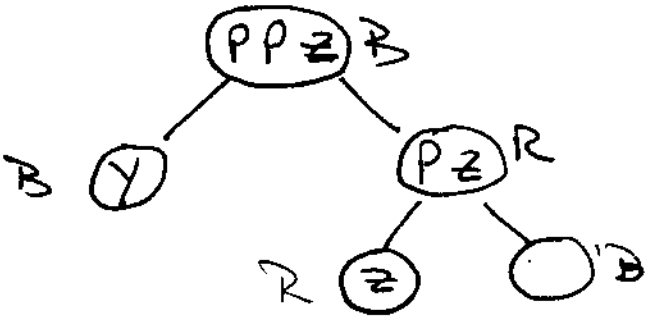



subcases 3, 4, 5

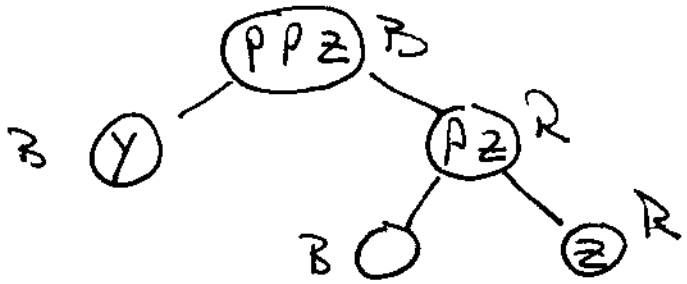
Case 4 : y is RED



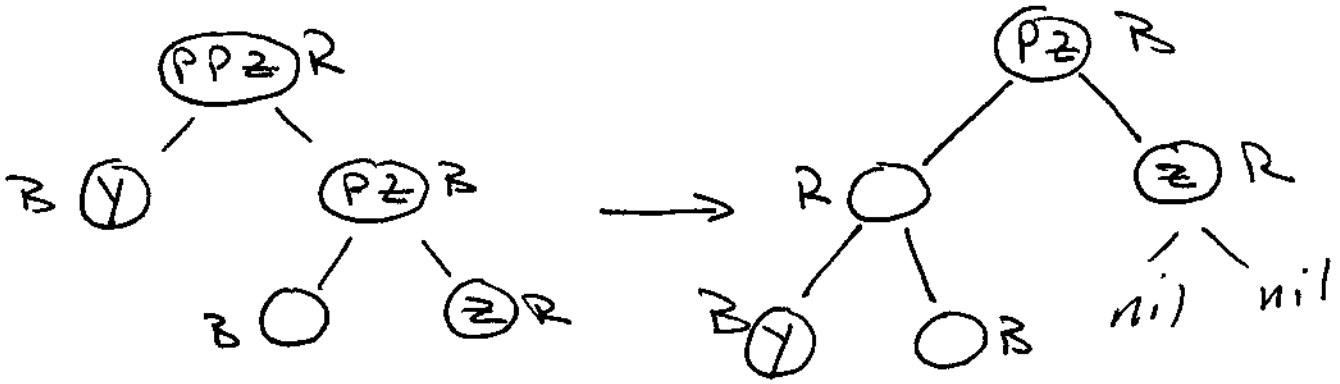

case 5 : y is Black, z is Pz's left child




Case 6 : y is Black, z is Pz's right child



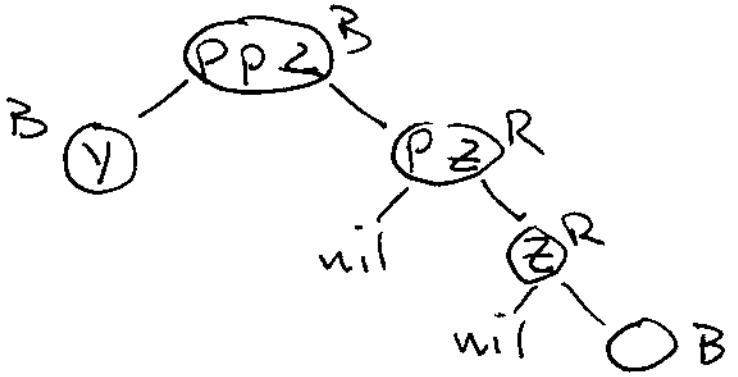
- In case 6 : color pz BLACK, color ppz RED, then LeftRotate about ppz




note RBT #4 is satisfied and RBT #5 is also satisfied.

- In case 5 : connect to case 6

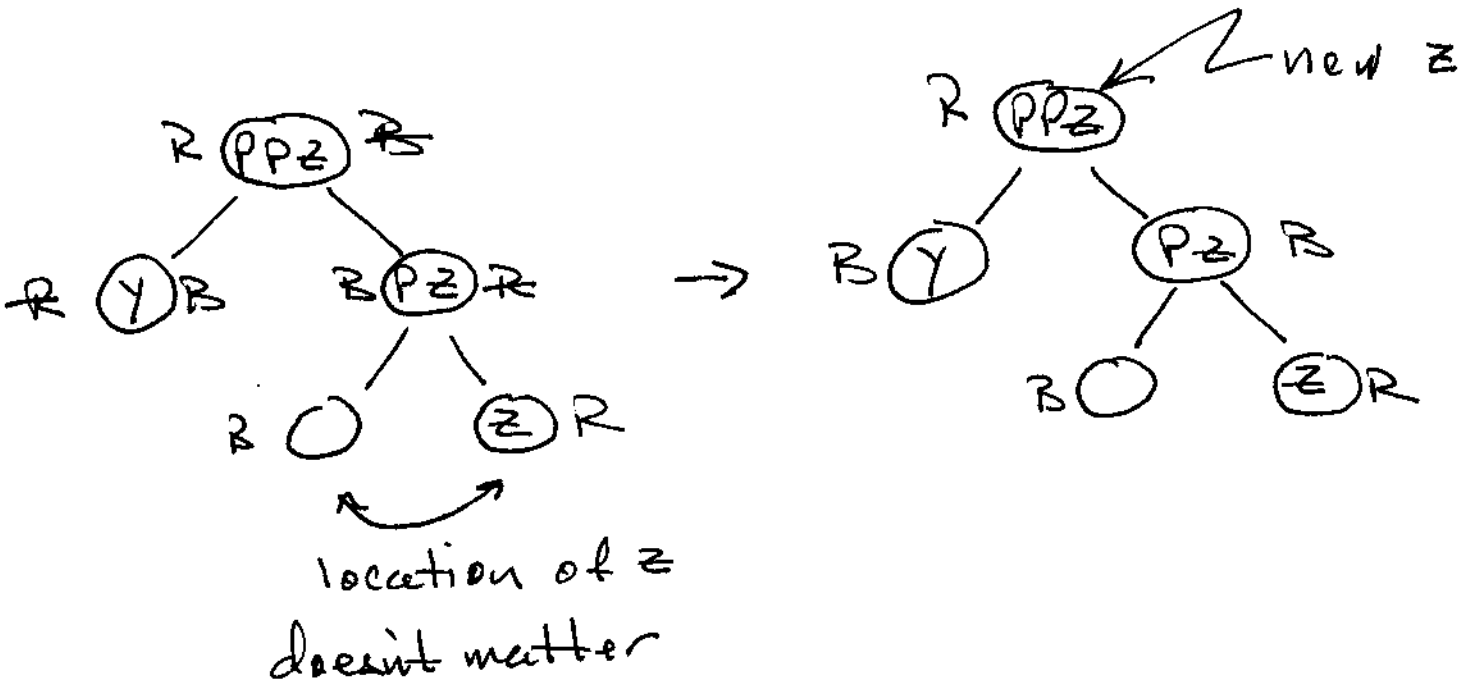
let z 'climb up' to pz, then Right Rotate about z. we get



- In case 4 :

fix color relations among ; y, pz, ppz,

then let z 'climb up' to ppz




Possible further iteration of while loop.