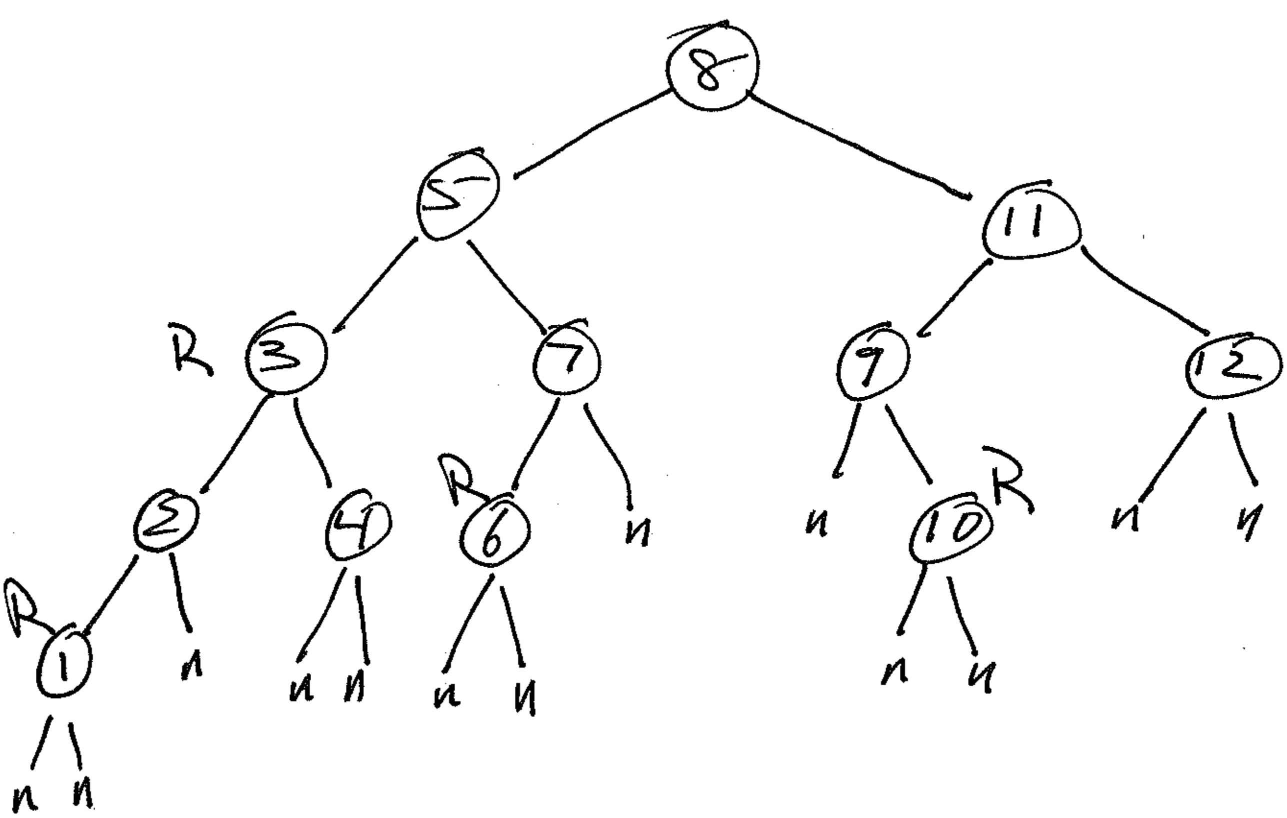
cse 101-01 5-24-22

- mid2: solns posted
- Pa8: (RBT) posted
- Pa7: last ext. to Wed. 10 PM

Ex. RBT



nodes	black height
8	3
5, 11, 3	2
1, 2, 4, 6, 7, 9, 10, 12	1
all nil (leaves)	0

note:

bh(x) = 0 iff height(x) = 0 iff x is a leaf (nil)

Theorem

A RBT with n internal (non-nil) nodes and height h satisfies

$$h \le 2\lg(n+1)$$

Rmks

- Any Binary Tree satisfies

$$h \geq \lfloor \lg n \rfloor$$

- so a RBT satisfies both

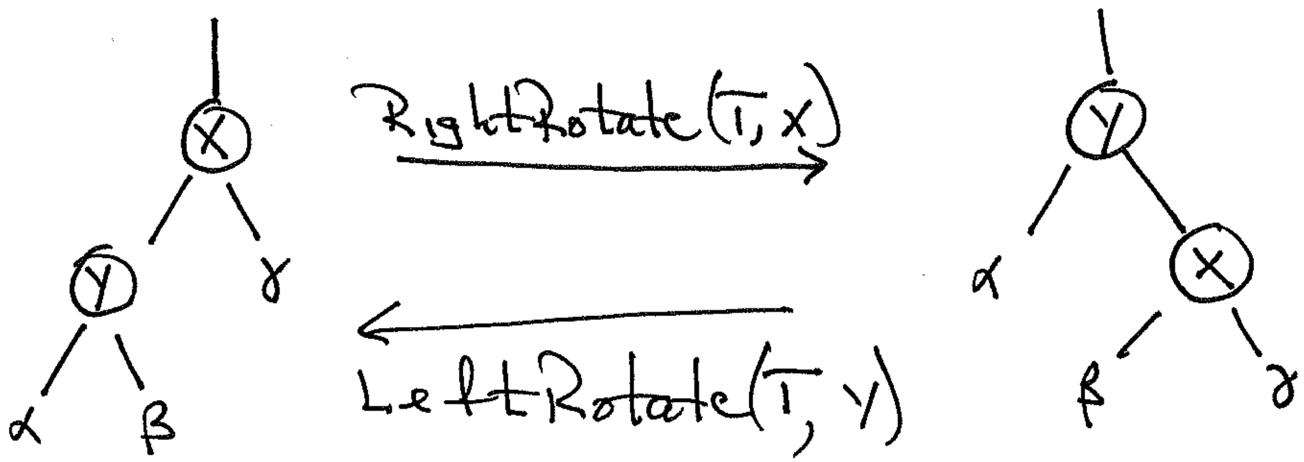
$$\Omega(\log n) \leq \text{height}(T) \leq O(\log n)$$

- so $\text{height}(T) = \Theta(\log n)$

- Therefore runtime of queries is $\Theta(\log n)$.

(13.2) Rotations

Picture



α, β, γ are arbitrary subtrees

Note: BST Properties are Preserved

$$keys(\alpha) \le key[y] \le keys(\beta) \le key[x] \le keys(\gamma)$$

neigther Preserve RBT Properties.

Summary of RightRotate(T, x) :

β
- left[x] = right[y]
- p[right[y]] = x

Parent
- p[y] = p[x]
- p[x]'s (left or right) child = y

$x \leftrightarrow y$
- right[y] = x
- p[x] = y

see pseudo-code

(13.3) Insertion

RB_Insert(T, z)

RB_InsertFixUp(T, z)

if p[z] is Red and

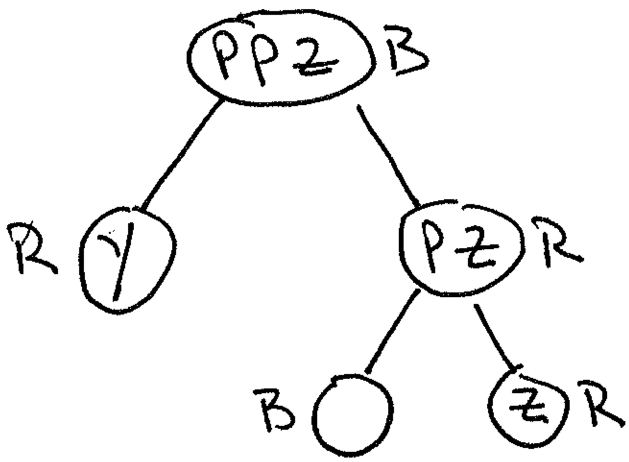
- pz == left[ppz] → cases 1, 2, 3
- pz == right[ppz] → cases 4, 5, 6

we do cases 4, 5, 6.

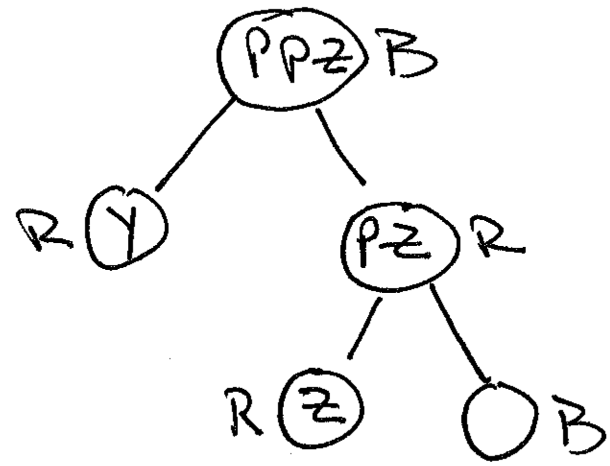
let y = left[ppz] (z's uncle).

Case 4: y is Red

(4.1)

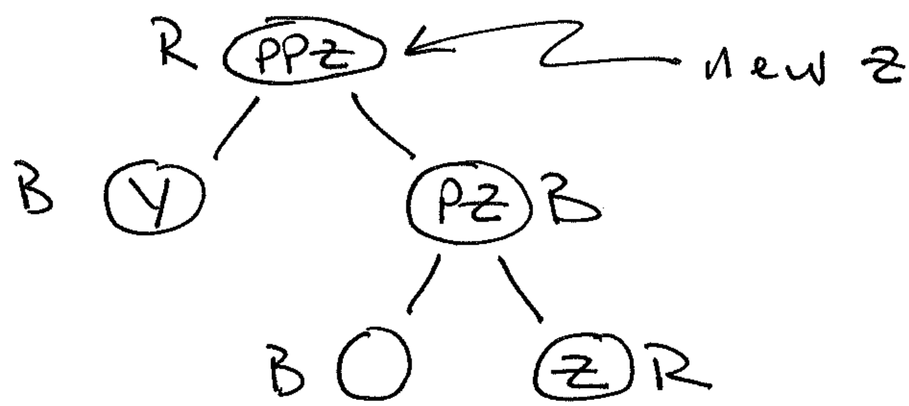


(4.2)



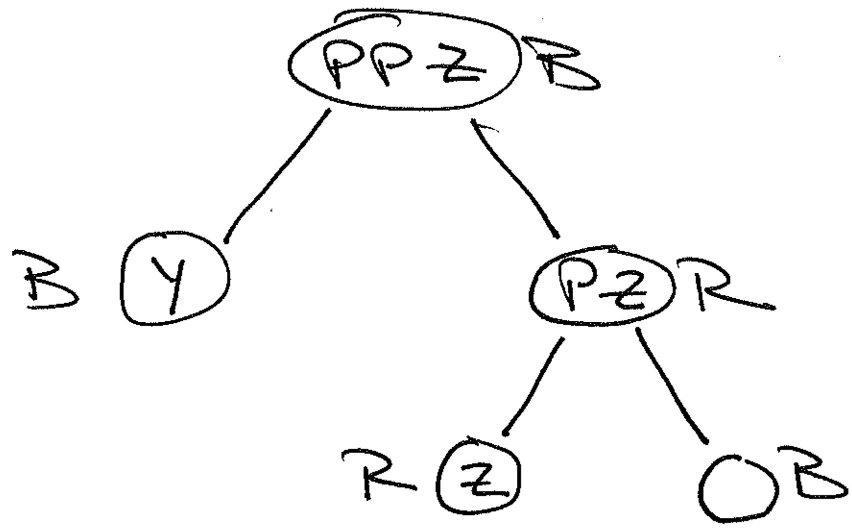
In this case: swap colors y, pz ⟺ ppz, then let z climb up to ppz

(4.1) ⟶



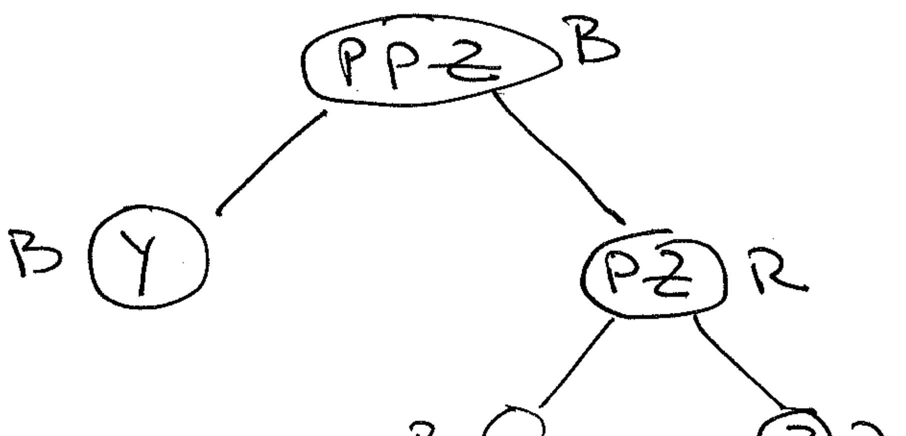
may iterate again.

<u>case 5</u>: y is Black, left[pz]==z

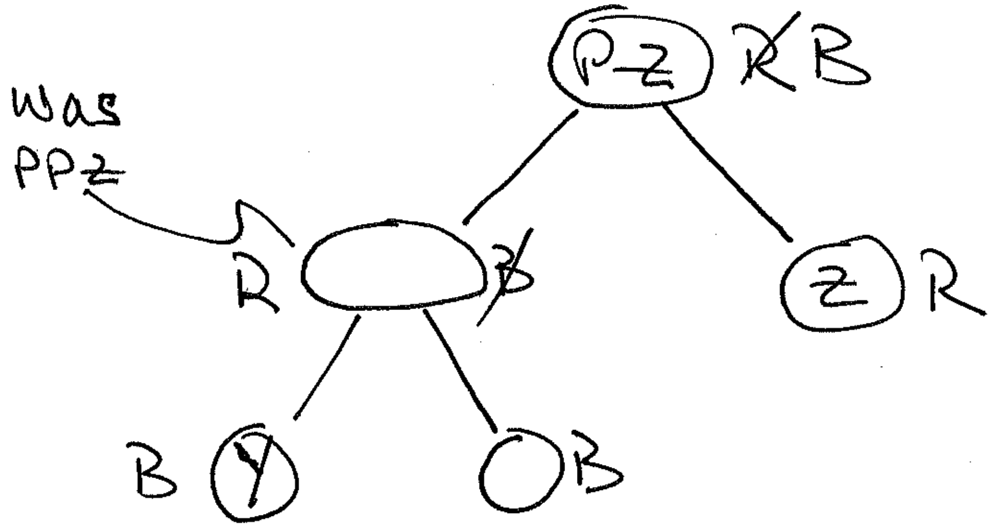


In this case: convert to case 6

<u>case 6</u>:



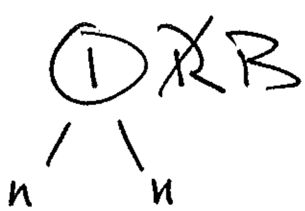
in this case: color ppz Red, then
color pz Black
left rotate about ppz, to get



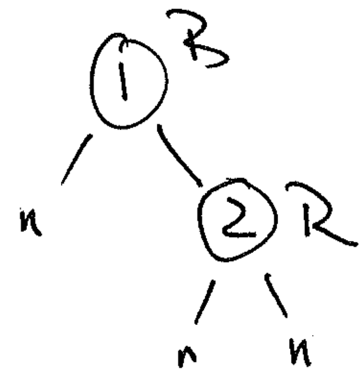
while loop must terminate now.

EX. insert ~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~ into an initially empty RBT.

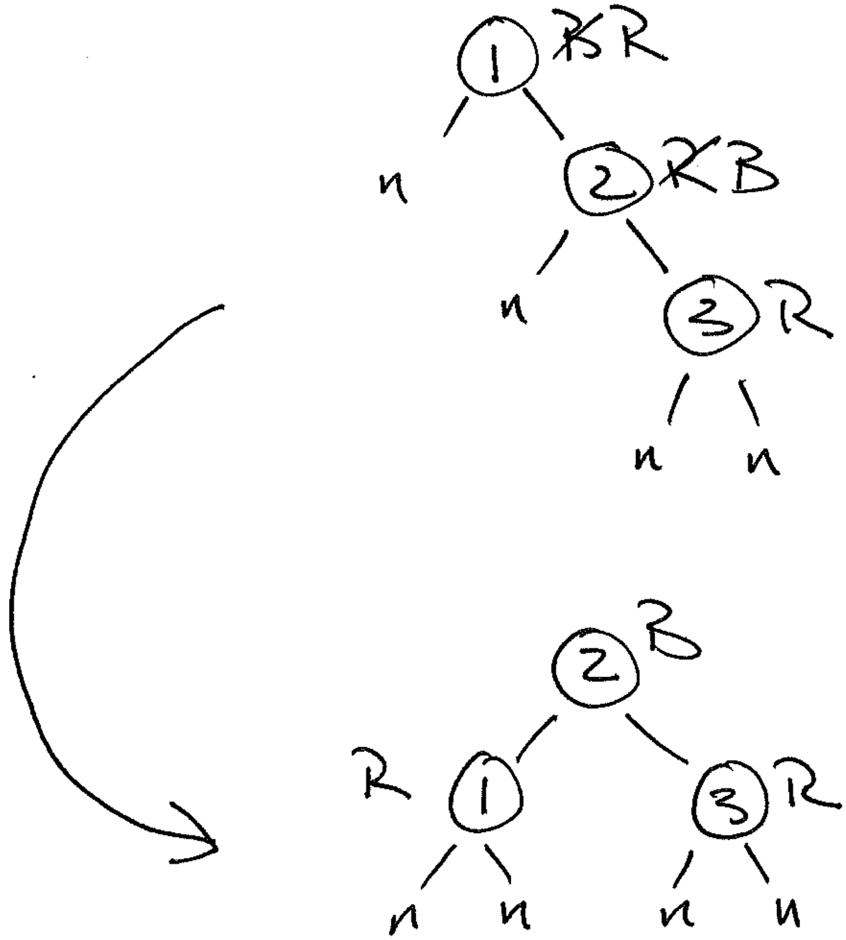
insert(1)



insert(2)



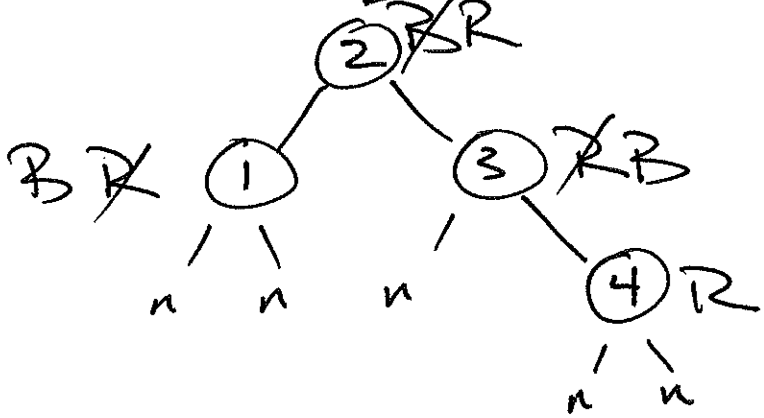
insert(3)



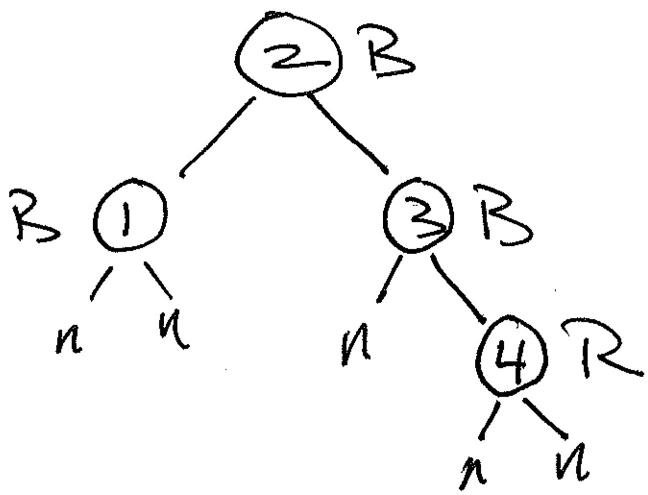
Case 6

insert(4)

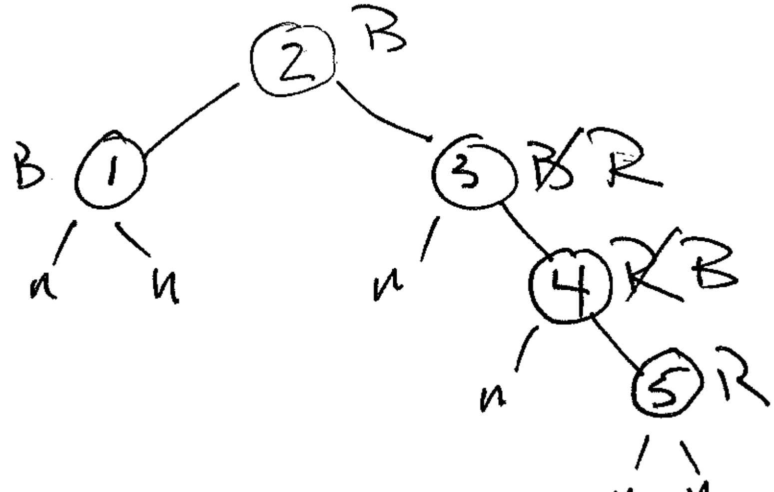
new z



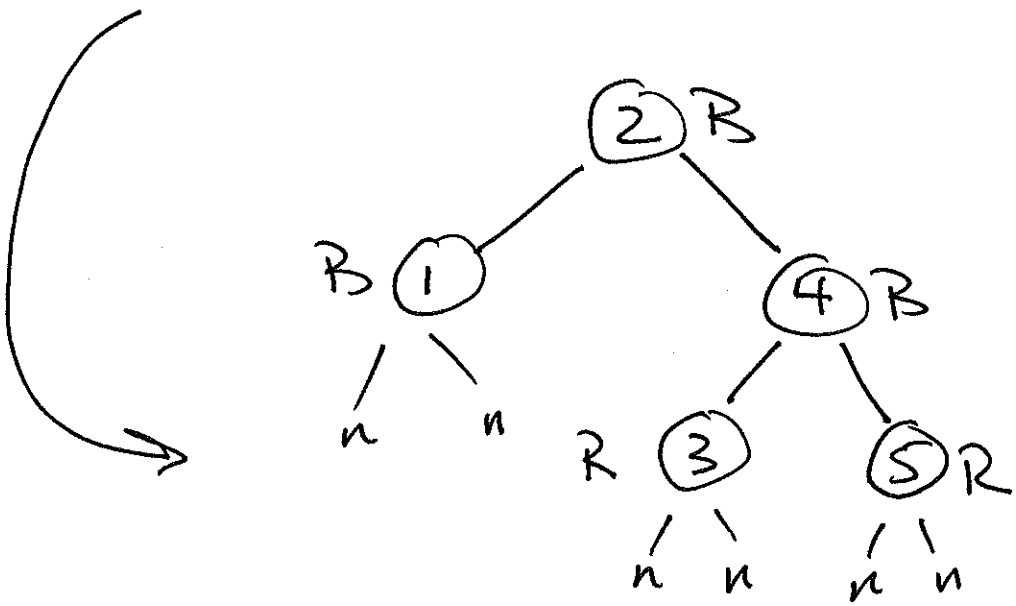
case 4



<u>insert(5)</u>



case 6


2 B
B 1
4 B
n n
R 3
5 R
n n n n