CSE 101 3-1-22

PA7: ext 2 more day (last)

continue RBT's ...

Note:

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

Theorem

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

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

Remarks

1. any Binary Tree satisfies $h \geq \lfloor \lg(n) \rfloor$, so A RBT T satisfies both:

$$\lfloor \lg n \rfloor \leq \text{height}(T) \leq 2\lg(n+1)$$

i.e.

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

$\therefore$ $\text{height}(T) = \Theta(\log n)$.

so BST algorithms all run in time $\Theta(\log n)$

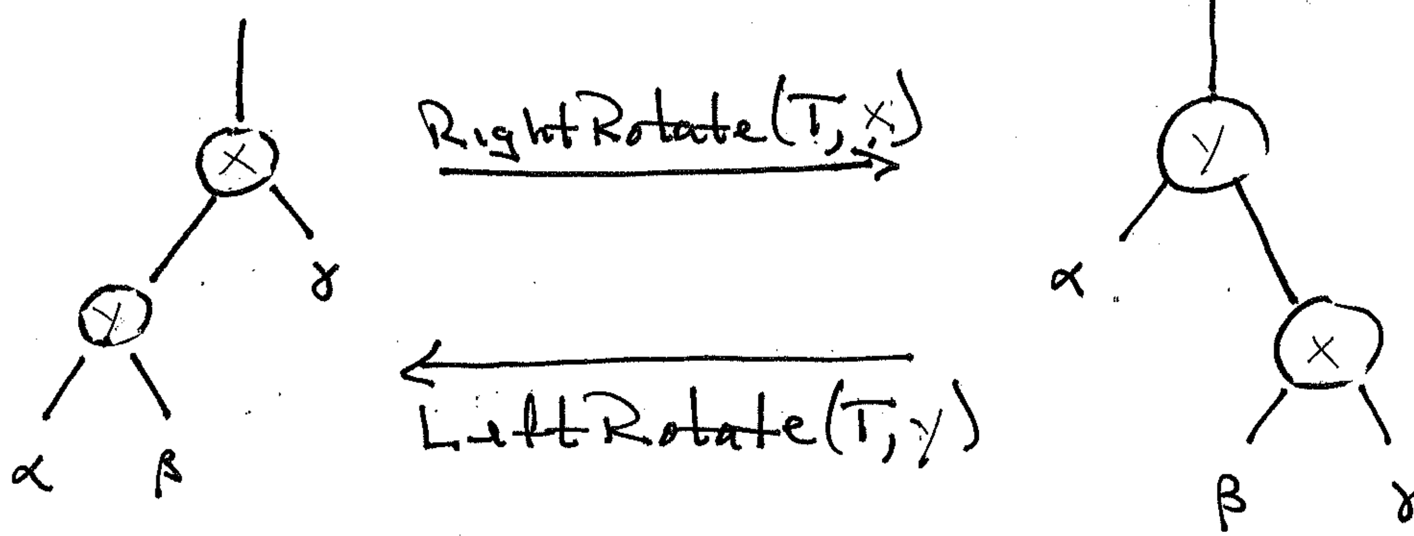
2. BST Insert() and Delete() don't preserve RBT properties, But, they can be altered to do so, and still run in time

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

(13.2) Rotations

To fix RBT properties, must change some colors, and tree structure.

Pictures



α, β, γ are arbitrary subtrees.

Both Rotations Preserve

$$\text{Key}(\alpha) \le \text{Key}[x] \le \text{Keys}(\beta) \le \text{Key}[y] \le \text{Keys}(\gamma),$$

Preconditions:

RightRotate(T, y) : left[y] ≠ nil

LeftRotate(T, x) : right[x] ≠ nil.

Summary of RightRotate(T, x):

$$\beta : \begin{cases} left[x] = right[y] \\ p[right[y]] = x \end{cases}$$

$$\text{Parent}: \begin{cases} p[y] = p[x] \\ p[x]\text{'s (left or right) child} = y \end{cases}$$

$$x, y : \begin{cases} right[y] = x \\ p[x] = y \end{cases}$$

(13.3) Insertion

Recall: TreeInsert(z) inserts a new node z in a BST.

which RBT prop. can then be violated.

✓ 1.) all node R or B.

✓ 2.) root is B

✓ 3.) leaves (nils) are B.

⇒ 4.) R nodes have only B children.

✓ 5.) black-heights are well defined.

we can make (5) true by coloring z RED.

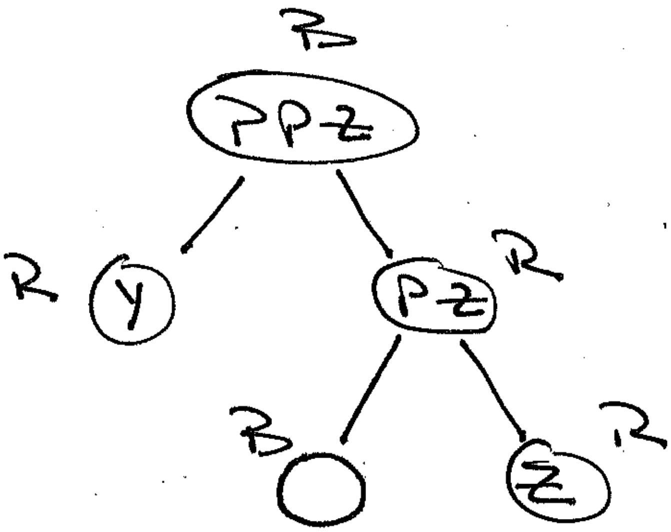
Goal: fix RBT prop. #4.

cases {4, 5, 6} arise when

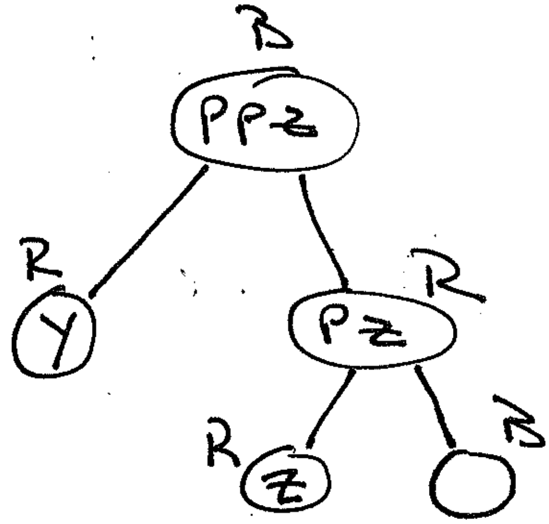
Parent[z] is right child of its Parent. assign y to be the left child of that grand-parent, i.e. y is z's uncle

$$y = \text{left}[p[p[z]]].$$

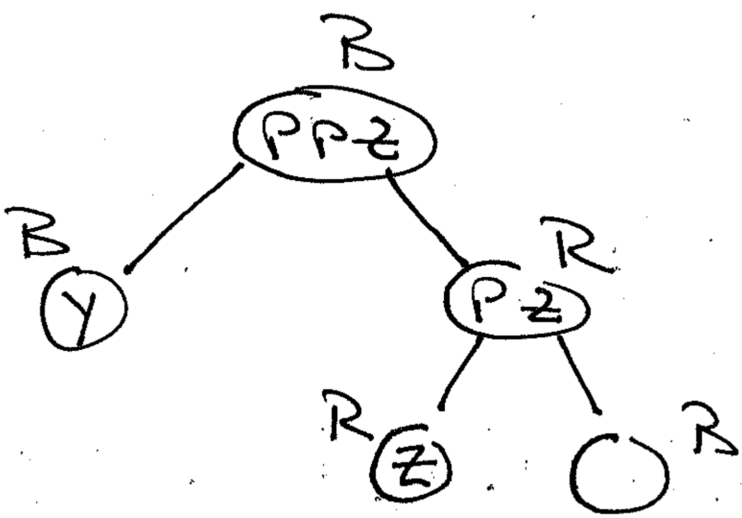
case 4: y is RED



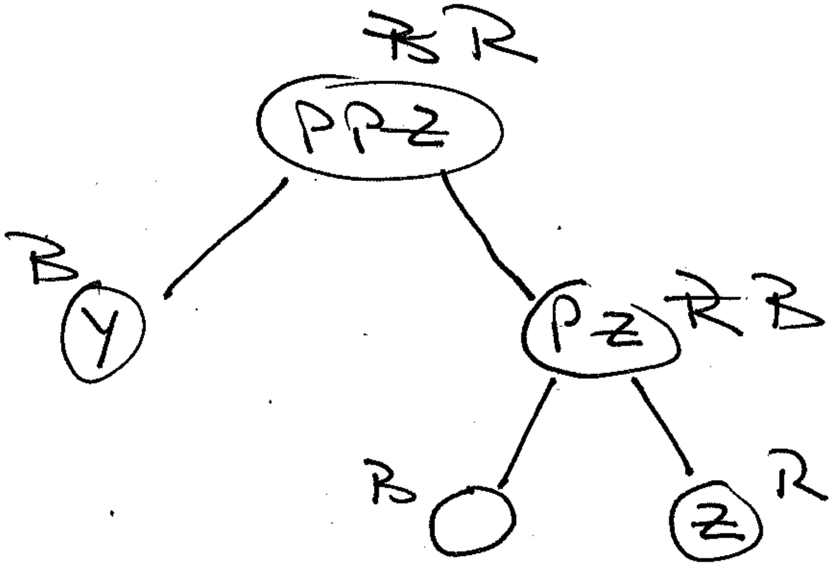
or



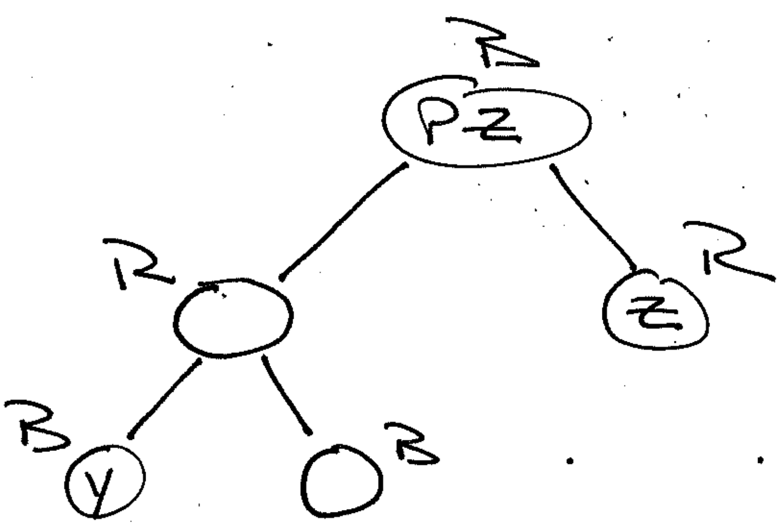
case 5: y is BLACK, z = left[p[z]]



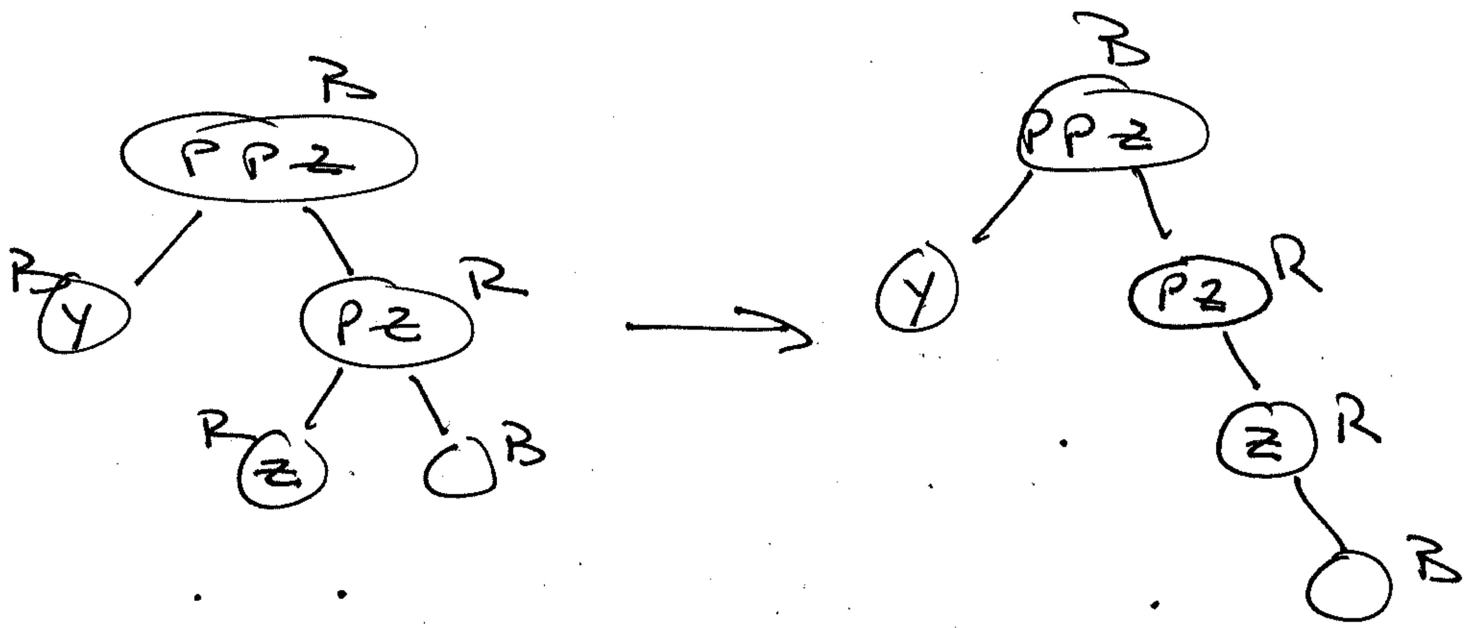
<u>case 6</u> : y is Black z = right[P[z]]



In case 6 : color P[z] Black, color P[P[z]] RED, then left rotate about P[P[z]].



In case 5, we convert to case 6 By : let z climb up to p[z], then right rotate about z



In case 4: fix colors among y, p[z] and p[p[z]], then let z climb up to p[p[z]] and possibly iterate