AVL tree: Height balance tree.

Binary Search Tree

RB Tree   AVL Tree

BST → n-nodes:

Search : $\Omega(\lg n) \longrightarrow$ best case

$O(n) \longrightarrow$ the worst case complexity.

Goal: Search : timing complexity $\Theta(\lg n)$.

Best case: $\Omega(\lg n)$

Worst case: $O(\lg n)$

Under BST: $T_{search} \propto h_{BST}$.

$(h_{BST} \longrightarrow Min) \longrightarrow (T_{search} \longrightarrow Min)$.

Min. possible height with n nodes $= \lg n$.

R.B. tree: $2 \cdot \lg(n+1)$ = Max. height of RB tree with n nodes.

$$h_{RB\text{-}Tree} = \Theta(\lg n)$$

$$\therefore T_{search-RB\,Tree} = \Theta(\lg n)$$

AVL tree: Definition: An AVL tree is a binary search tree satisfying the following property.

- $(\forall \text{nodes} \in \text{AVL-Tree})(\max(|\text{Height}(LST(node)) - \text{Height}(RST(node))|) = 1)$

$mod: R \rightarrow R^+ \cup \{0\}$.

$$mod(x) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases} \quad \Big| \quad |x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0. \end{cases}$$

Also denoted by $|x|$

$|x| \geq 0 \quad max(|x|) = 1 \wedge mod(x) \in I.$

$|x| = 0$ or $mod(x) = 1.$

$|x| = 0 \rightarrow x = 0$

$|x| = 1 \rightarrow x = 1$ or $x = -1$

$max(|x|) = 1 \wedge mod(x) \in I.$

$\equiv |x| = 0 \vee |x| = 1$

$\equiv x = 0 \vee x = 1 \vee x = -1.$

$(\forall nodes \in \text{AVL Tree})\ (max(|Height(LST(node)) - Height(RST(node))| = 1)$

$\equiv (\forall nodes \in \text{AVL tree})$

$Height(LST(node)) - Height(RST(node)) = 0$
$\vee$
$Height(LST(node)) - Height(RST(node)) = 1$
$\vee$
$Height(LST(node)) - Height(RST(node)) = -1.$

I-V.

Let $node_0$ be any arbitrary node in AVL Tree.

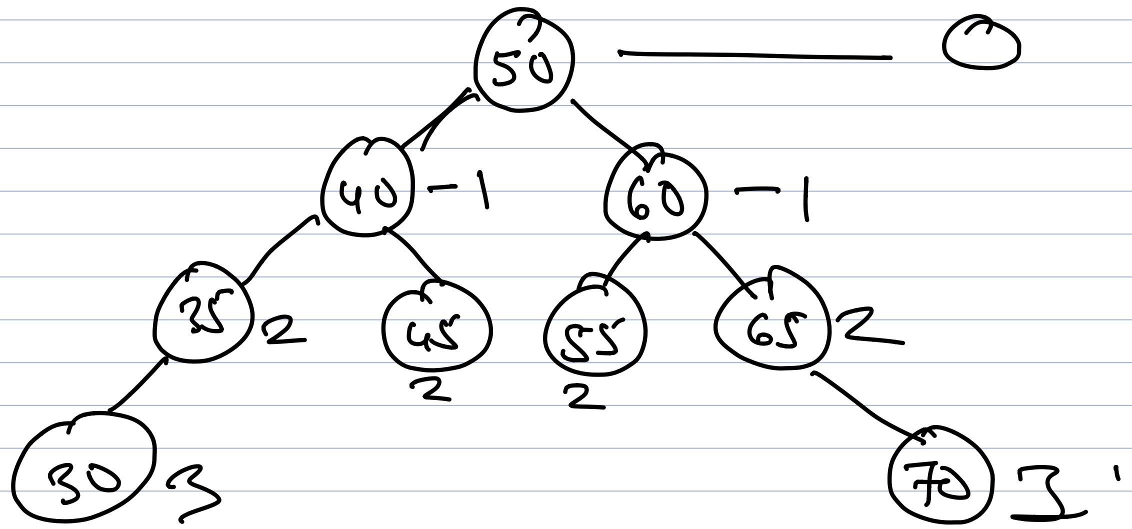
$Height(LST(node_0)) - Height(RST(node_0)) = 0$
$\vee$
$Height(LST(node_0)) - Height(RST(node_0)) = 1$
$\vee$
$Height(LST(node_0)) - Height(RST(node_0)) = -1.$

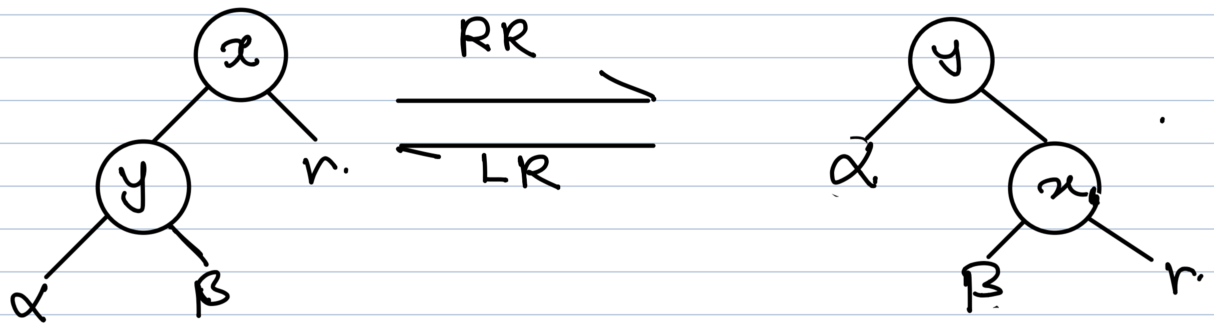
(i) Maintain height of node in struct avl_node.

And modify the insert scheme.

(2) Let T be a BST and let $d \in T$.

Write a routine to find out the height of node containing d as data.

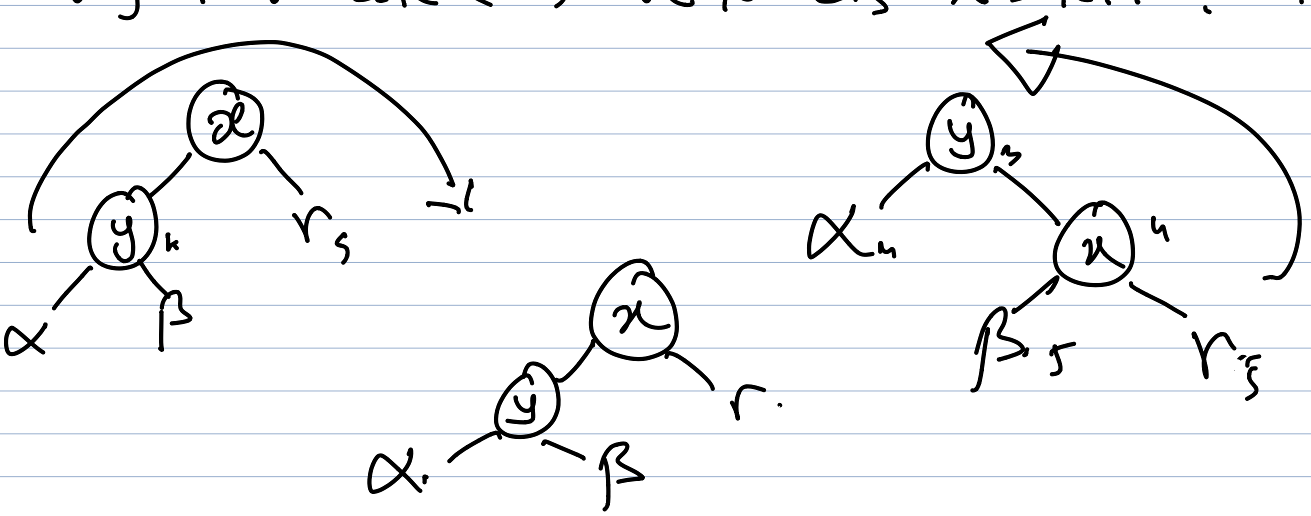


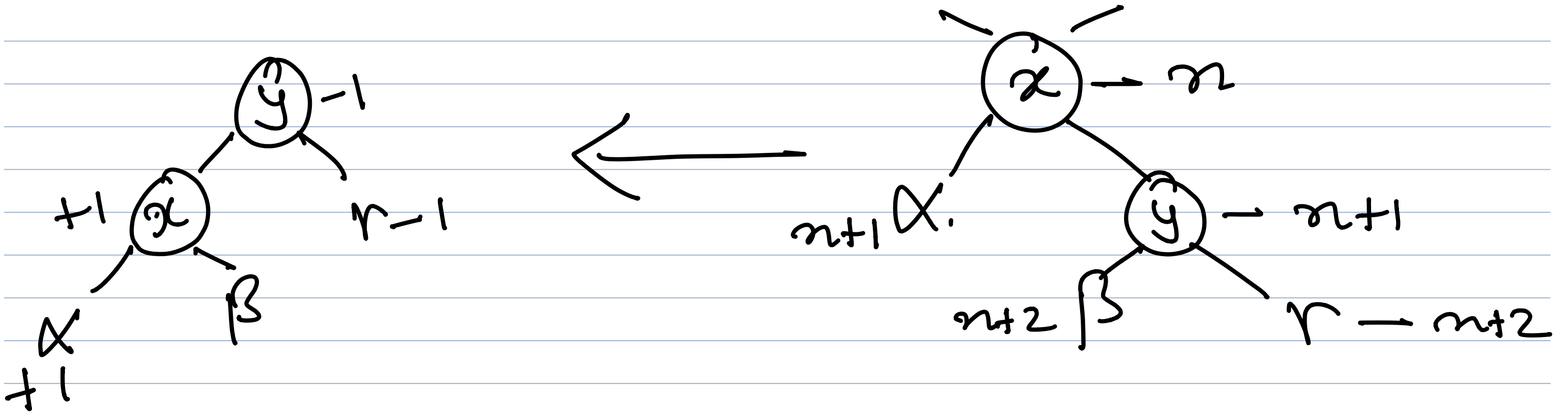


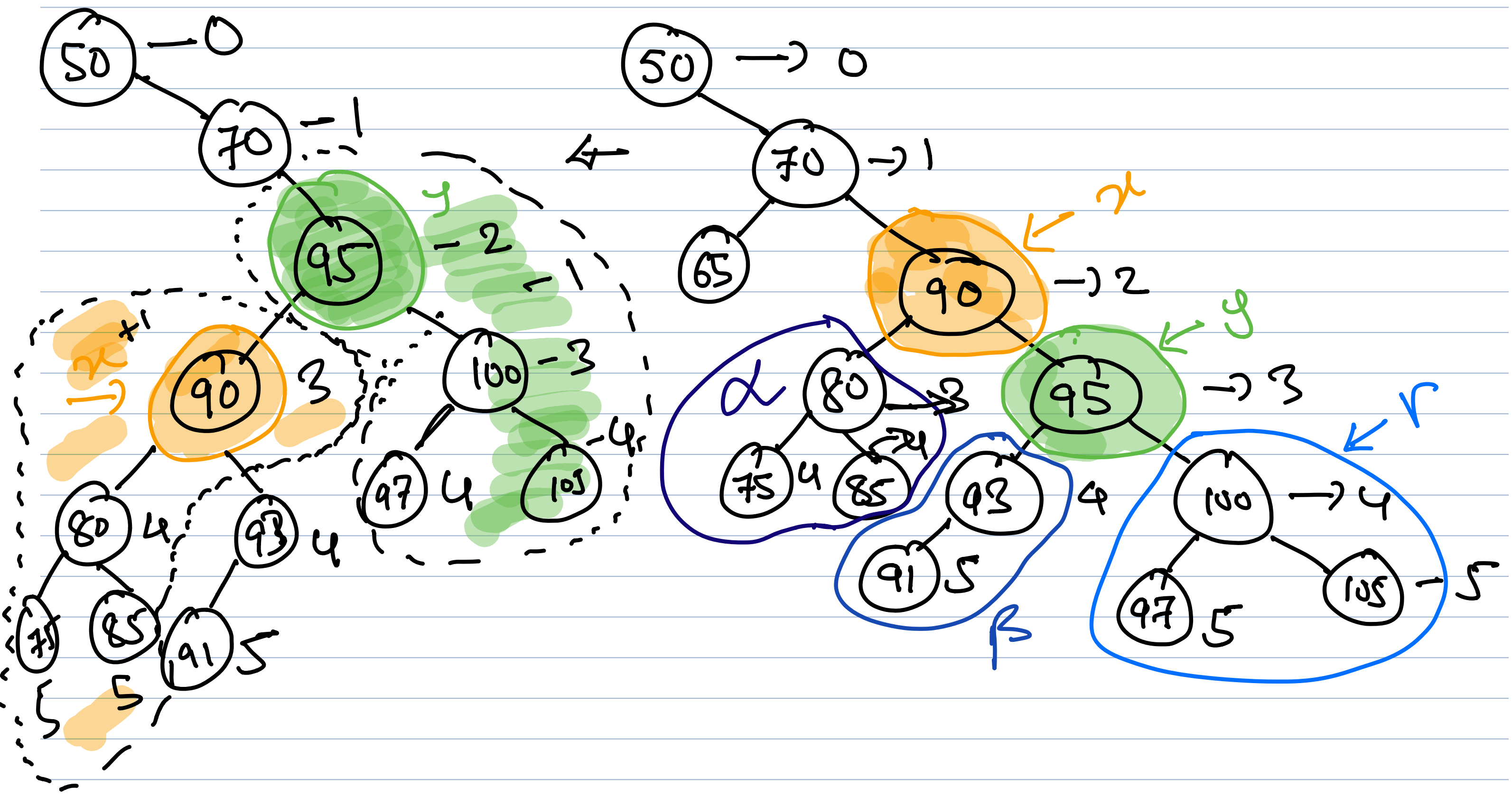
Modify RR + LR routine to adjust the heights of the nodes involved.

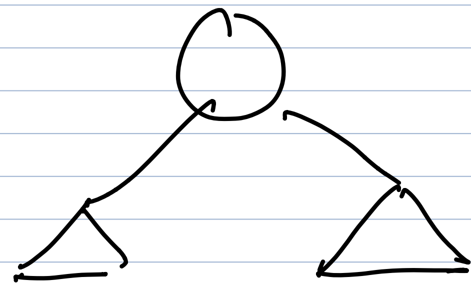
left rotate (x) requires $x \rightarrow$ right != NULL

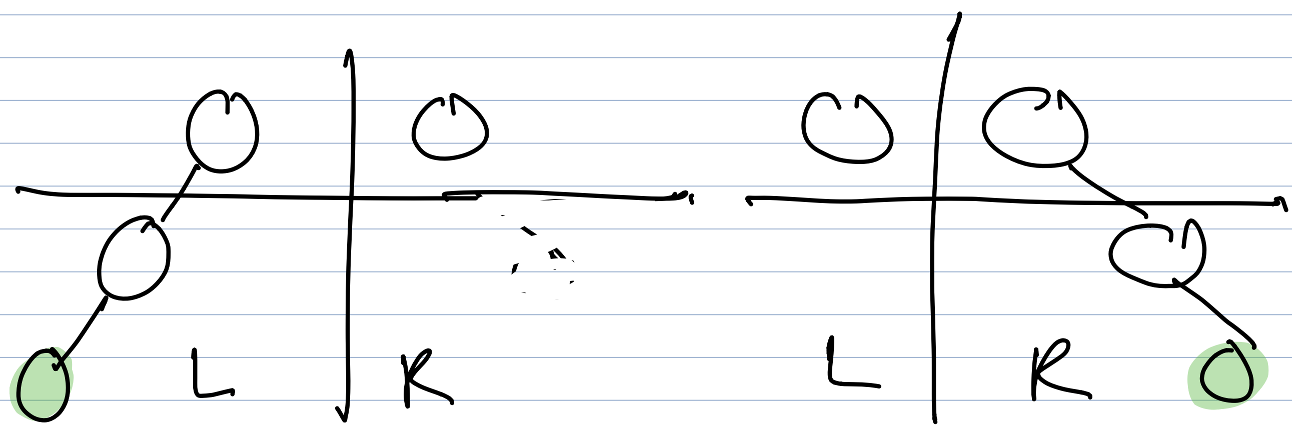
right rotate (x) requires $x \rightarrow$ left != NULL.

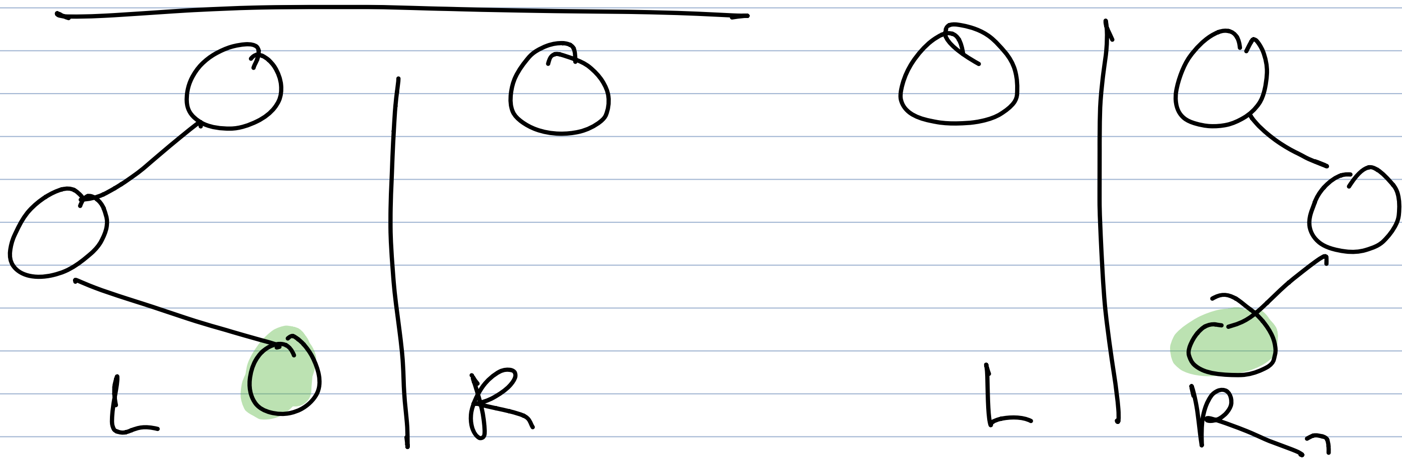



y -1
+1 x
r-1
β
α +1
x → n
n+1 α
y → n+1
n+2 β
r → n+2


50 — 0
70 — 1
95 — 2
y
x +1
90 3
100 — 3
97 4
105 — 4
80 4
93 4
75
85
91 5
5
5
50 → 0
70 → 1
65
90 → 2
x
95 → 3
y
α
80 → 3
75 4
85 → 4
93 4
91 5
β
100 → 4
97 5
105 → 5
r




L
R
L
R


L
R
L
R