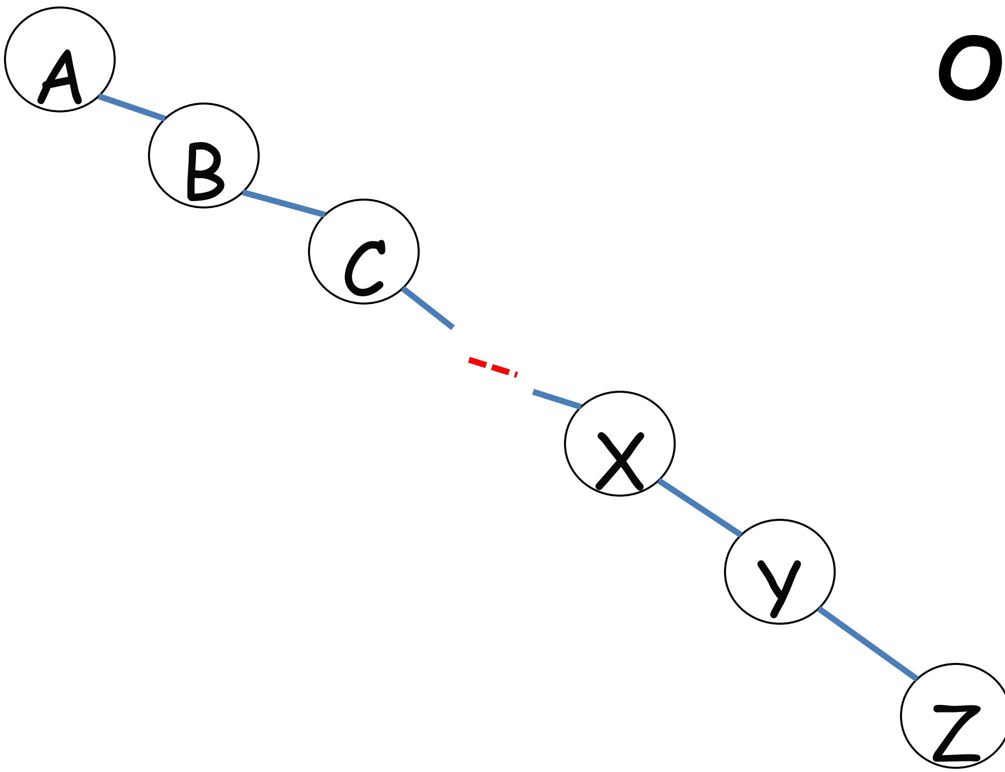
Data Structure and Algorithm (CS-102)

Dr. Sambit Bakshi

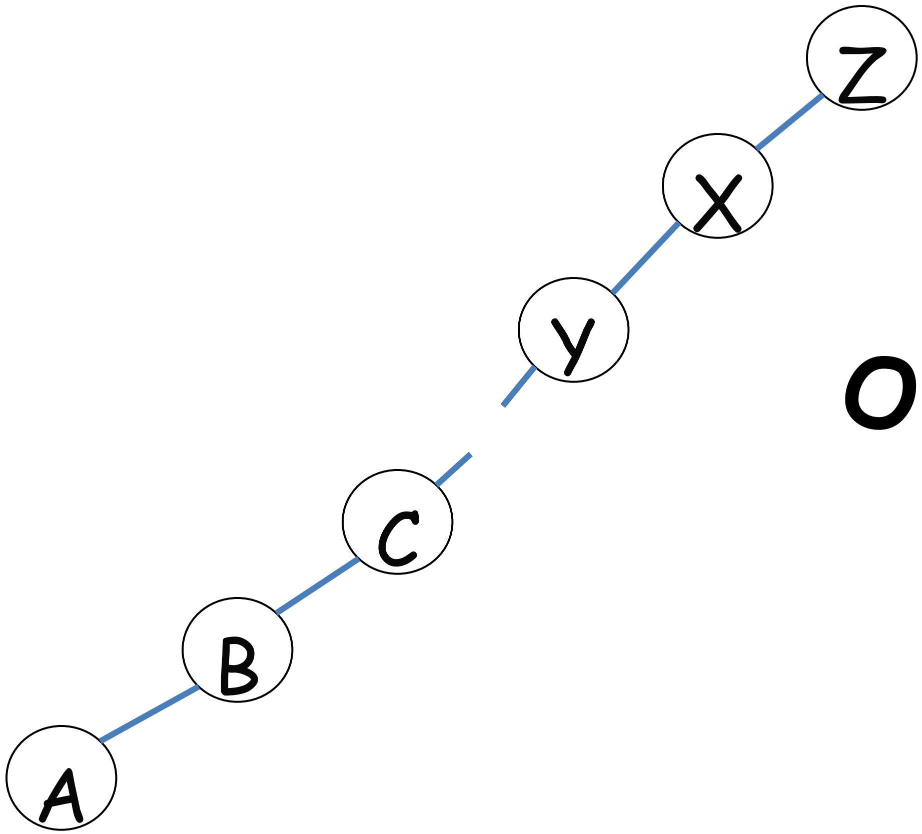
Consider the insertion of following element A, B, C, , ….,X, Y, Z into the BST



O(N)

. . .

Consider the insertion of following element Z, X, Y, , ….,C, B, A into the BST



O(N)

Balanced binary tree

- The disadvantage of a binary search tree is that its height can be as large as N-1
- This means that the time needed to perform insertion and deletion and many other operations can be O(N) in the worst case
- We want a tree with small height
- A binary tree with N node has height **at least** O(log N)
- Thus, our goal is to keep the height of a binary search tree **O(log N)**
- Such trees are called **balanced** binary search trees. Examples are AVL tree, red-black tree.

AVL tree

- An AVL tree is a binary search tree in which
  - for every node in the tree, the height of the left and right subtrees differ by **at most 1.**
  - **An empty binary tree is an AVL tree**

AVL tree

T^L left subtree of T

$h(T^L)$ Height of the subtree T^L

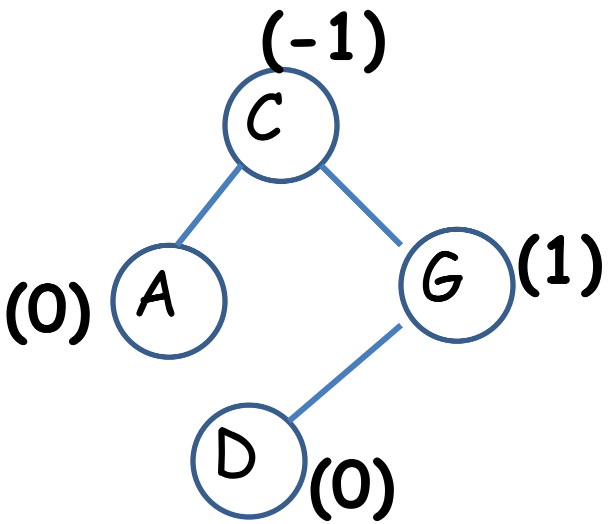
T^R Right subtree of T

$h(T^R)$ Height of the subtree T^R

T is an AVL tree iff T^L and T^R are AVL tree and $|h(T^L) - h(T^R)| <= 1$

$h(T^L) - h(T^R)$ is known as balancing factor (BF) and for an AVL tree the BF of a node can be either 0 , 1, or -1

AVL Search Tree

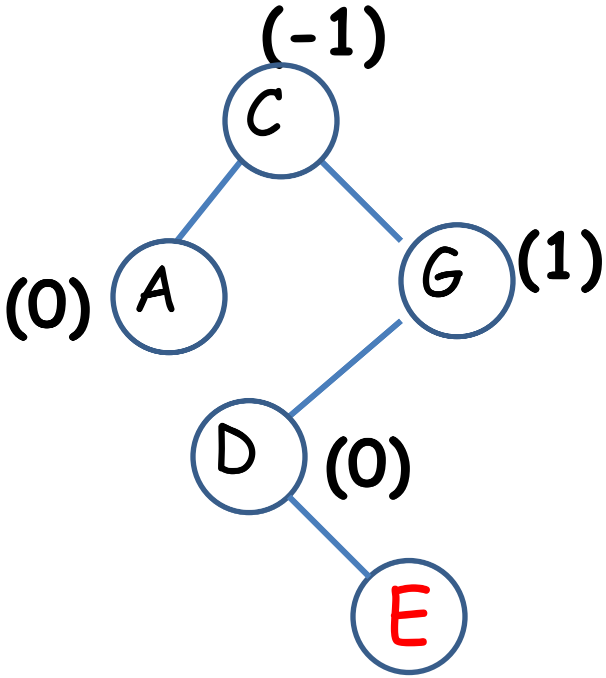


Insertion in AVL search Tree

Insertion into an AVL search tree may affect the BF of a node, resulting the BST unbalanced.

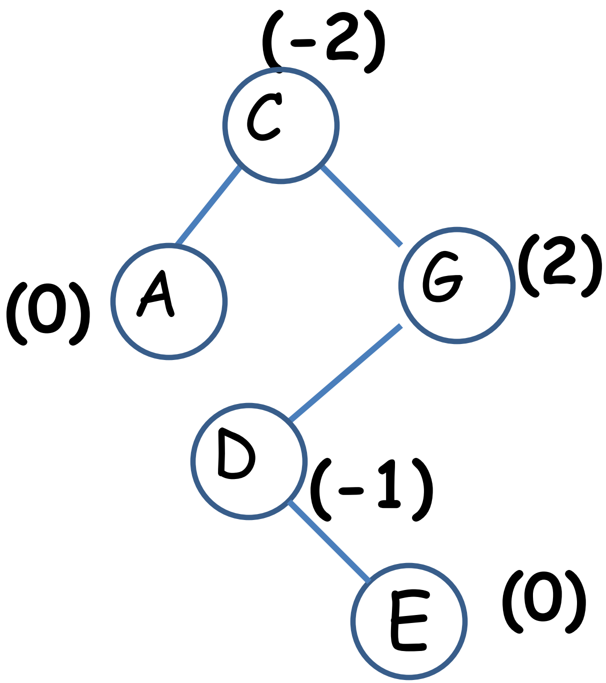
A technique called **Rotation** is used to restore the balance of the search tree

AVL Search Tree



E being inserted now

AVL Search Tree



Rotation

To perform rotation – Identify a specific node A such that:

BF(A) is neither 0, 1, or -1

and

which is the nearest ancestor to the inserted node on the path from the inserted node to the root

Rotation

Rebalancing rotation are classified as LL, LR, RR and RL

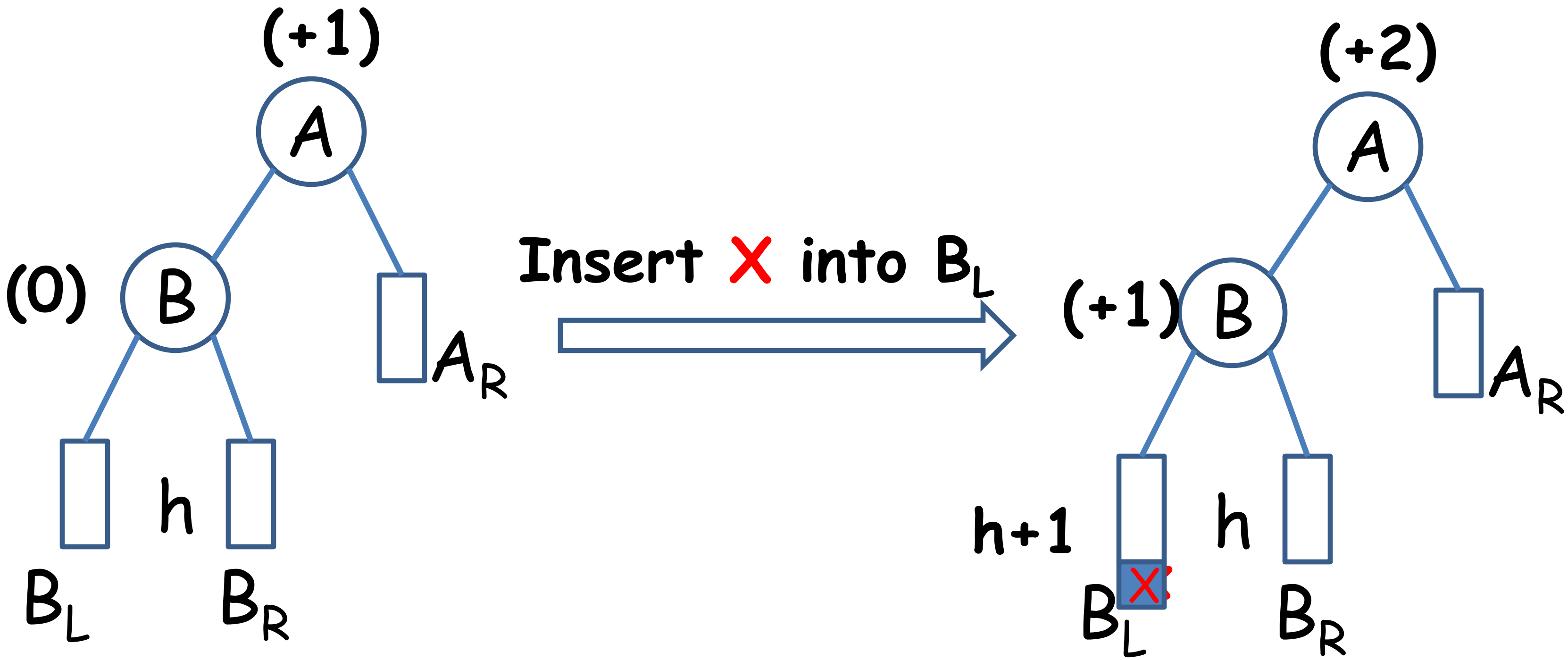
LL Rotation: Inserted node is in the left sub-tree of left sub-tree of node A

RR Rotation: Inserted node is in the right sub-tree of right sub-tree of node A

LR Rotation: Inserted node is in the right sub-tree of left sub-tree of node A

RL Rotation: Inserted node is in the left sub-tree of right sub-tree of node A

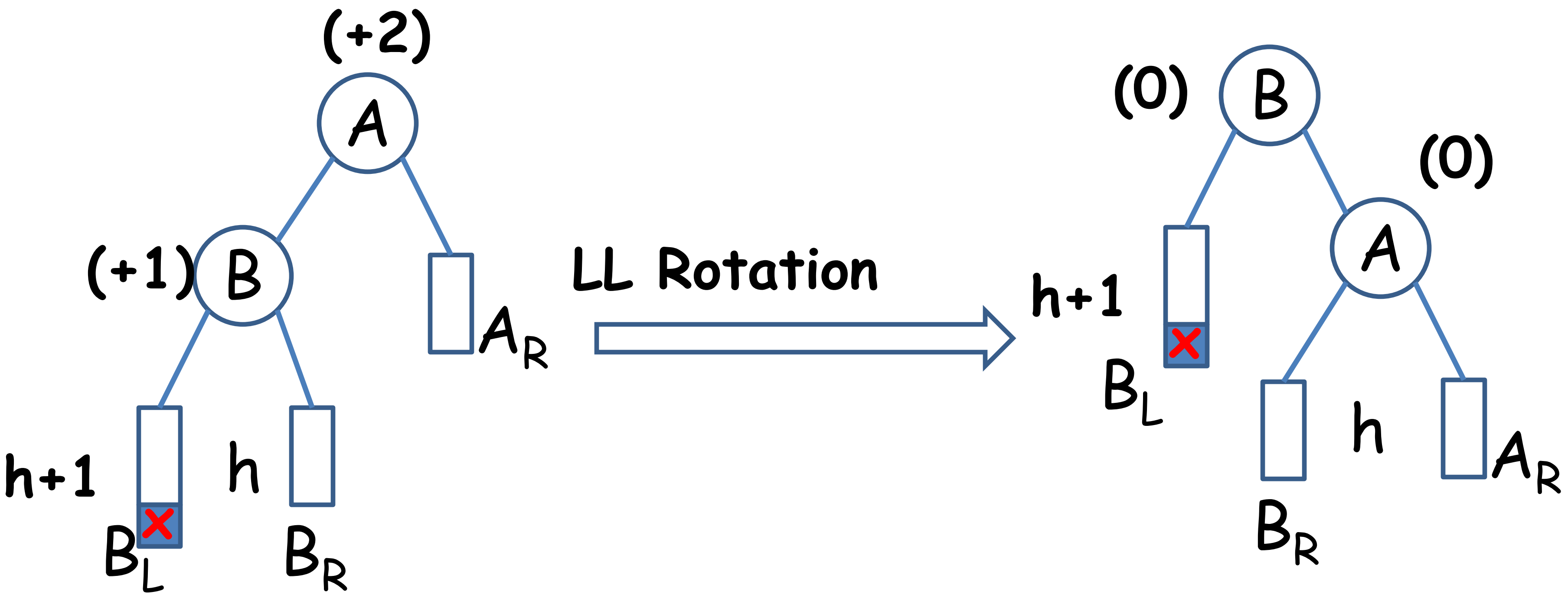
LL Rotation



Unbalanced AVL search tree after insertion

B_L : Left Sub-tree of B
B_R : Right Sub-tree of B
A_R : Right Sub-tree of A
h : Height

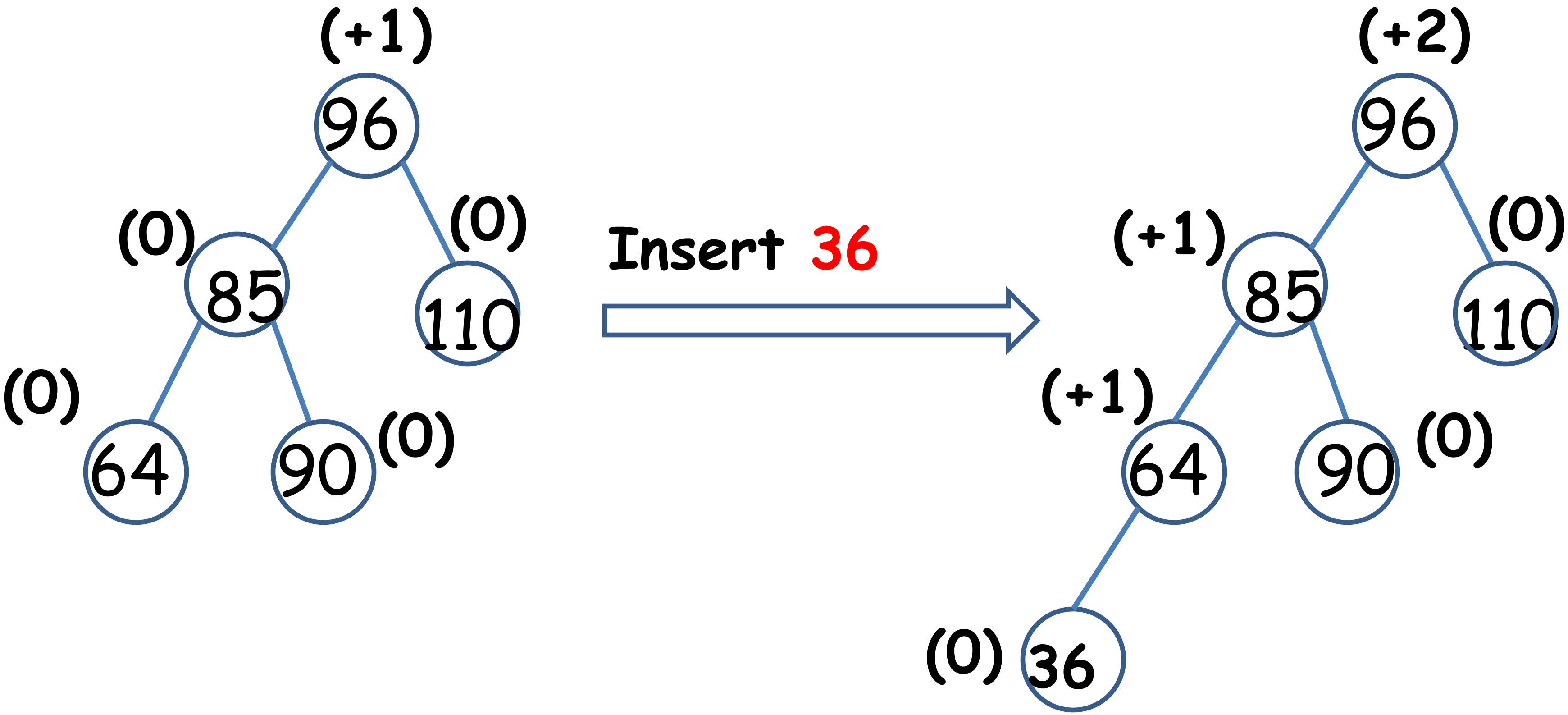
LL Rotation



Unbalanced AVL search tree after insertion

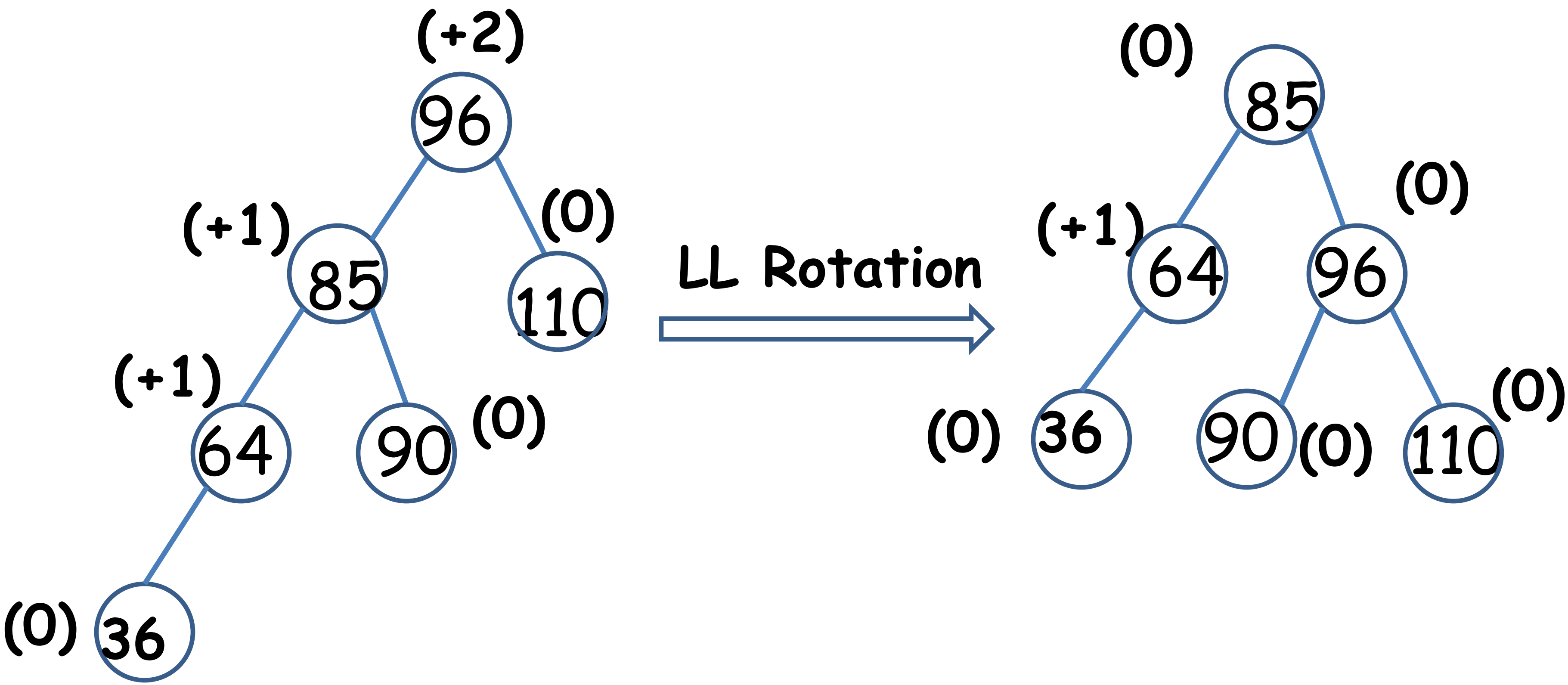
Balanced AVL search tree after rotation

LL Rotation Example



Unbalanced AVL search tree

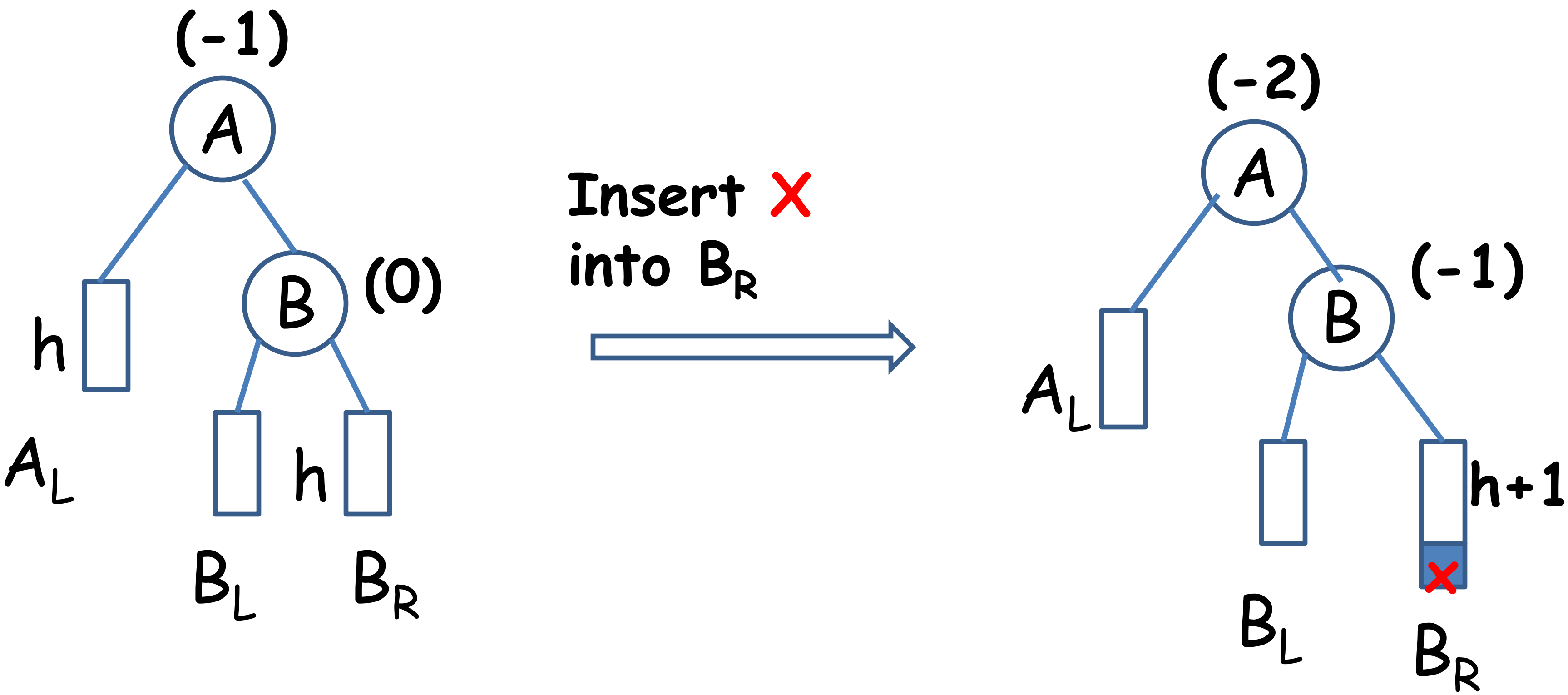
LL Rotation Example



Unbalanced AVL search tree

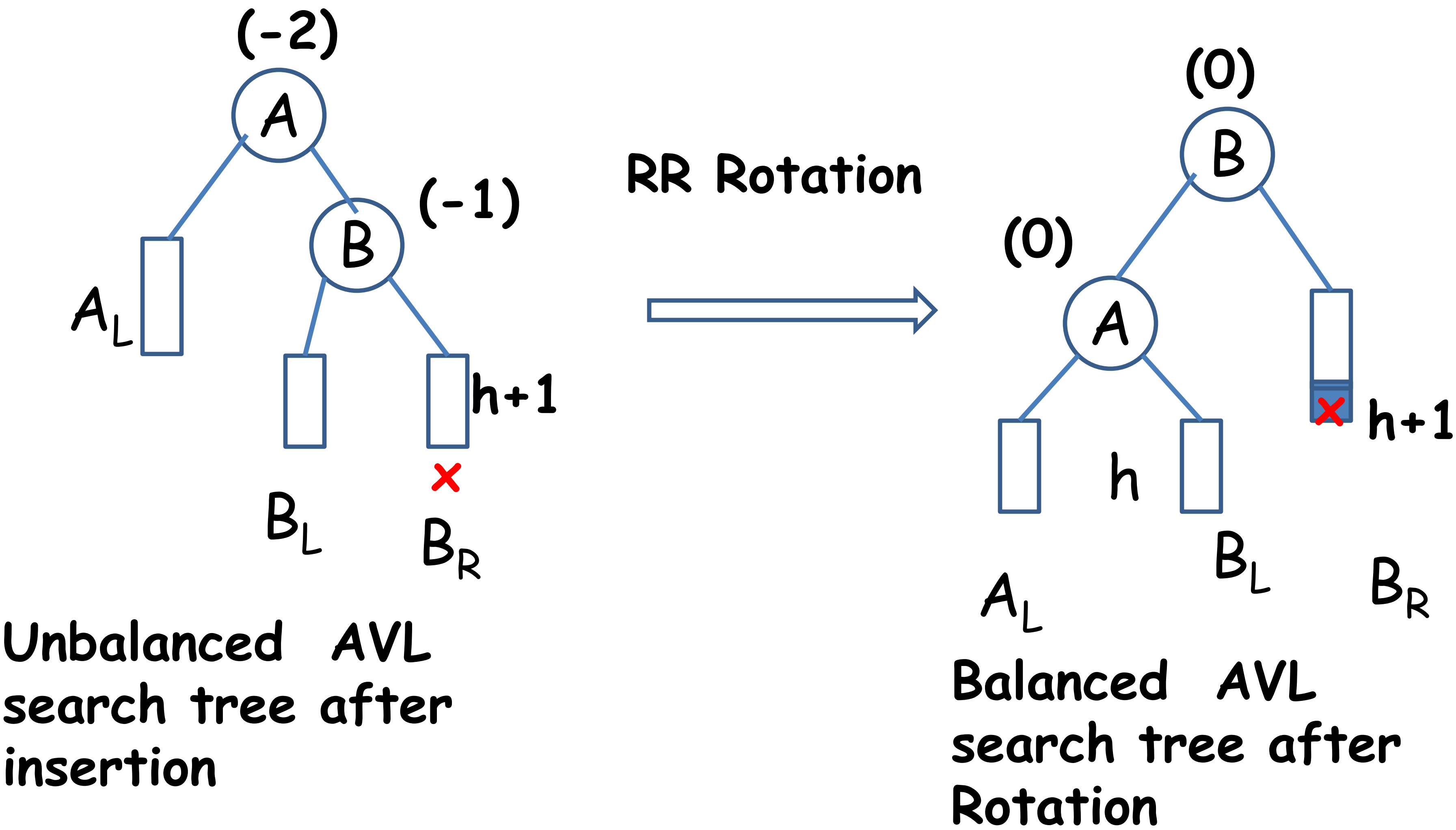
Balanced AVL search tree after LL rotation

RR Rotation

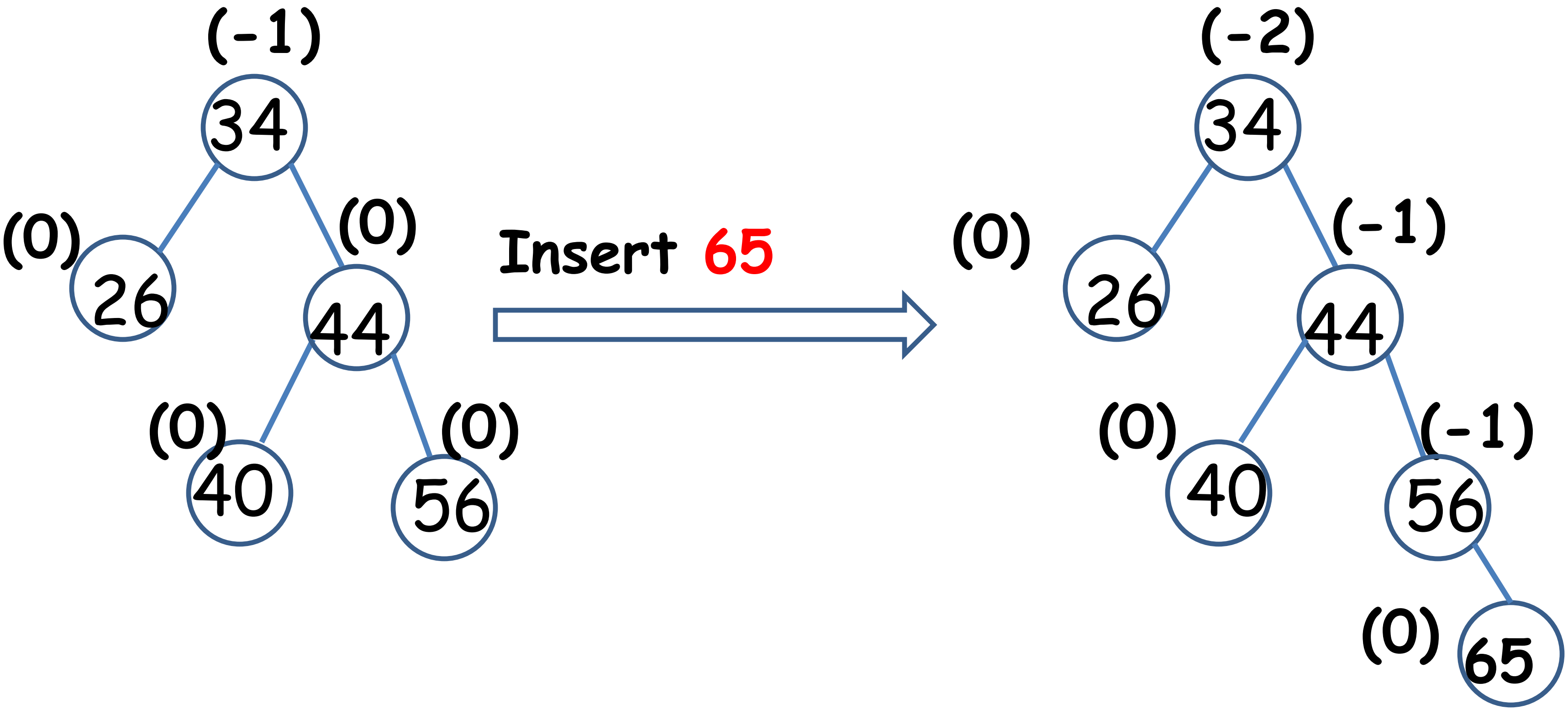


Unbalanced AVL search tree after insertion

RR Rotation

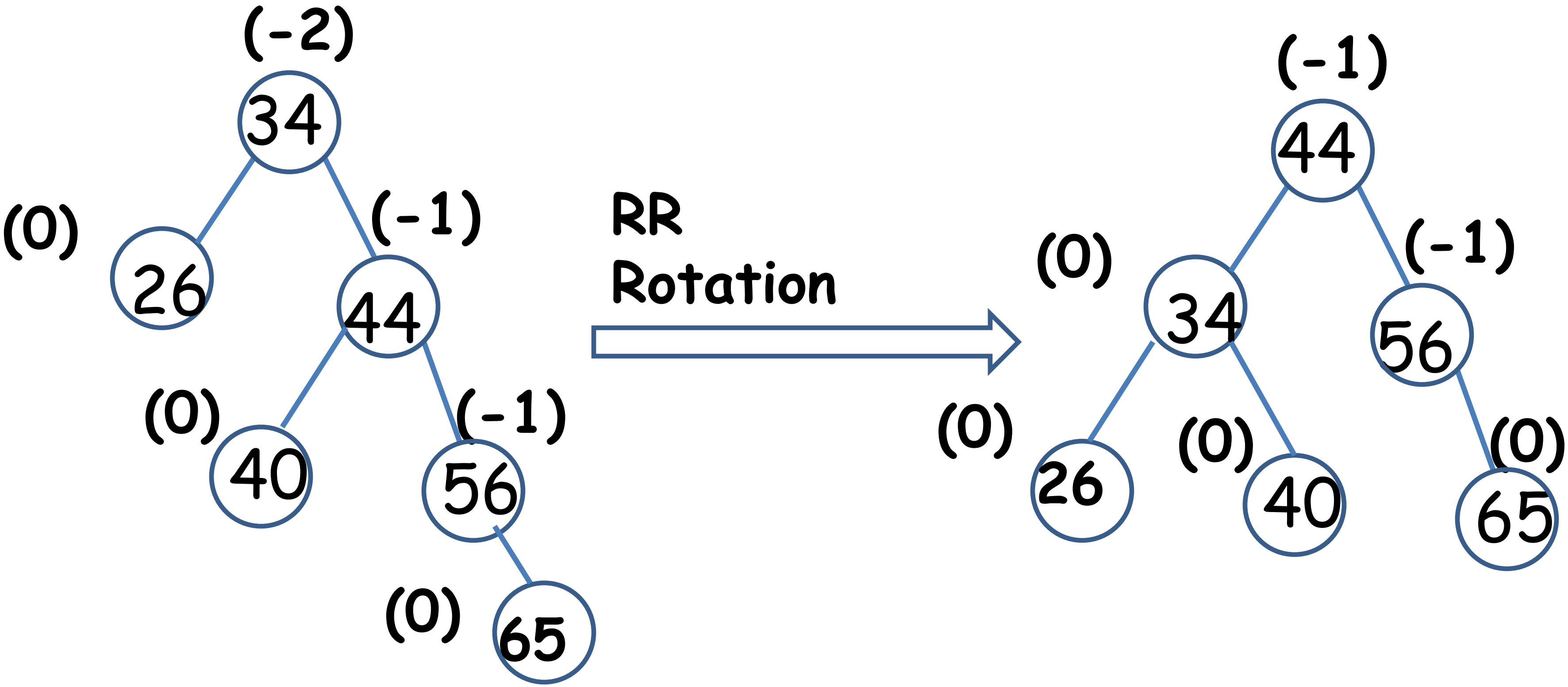

(-2)
A
(-1)
B
A_L
h+1
B_L
B_R
Unbalanced AVL search tree after insertion
RR Rotation
(0)
B
(0)
A
h+1
h
A_L
B_L
B_R
Balanced AVL search tree after Rotation

RR Rotation Example



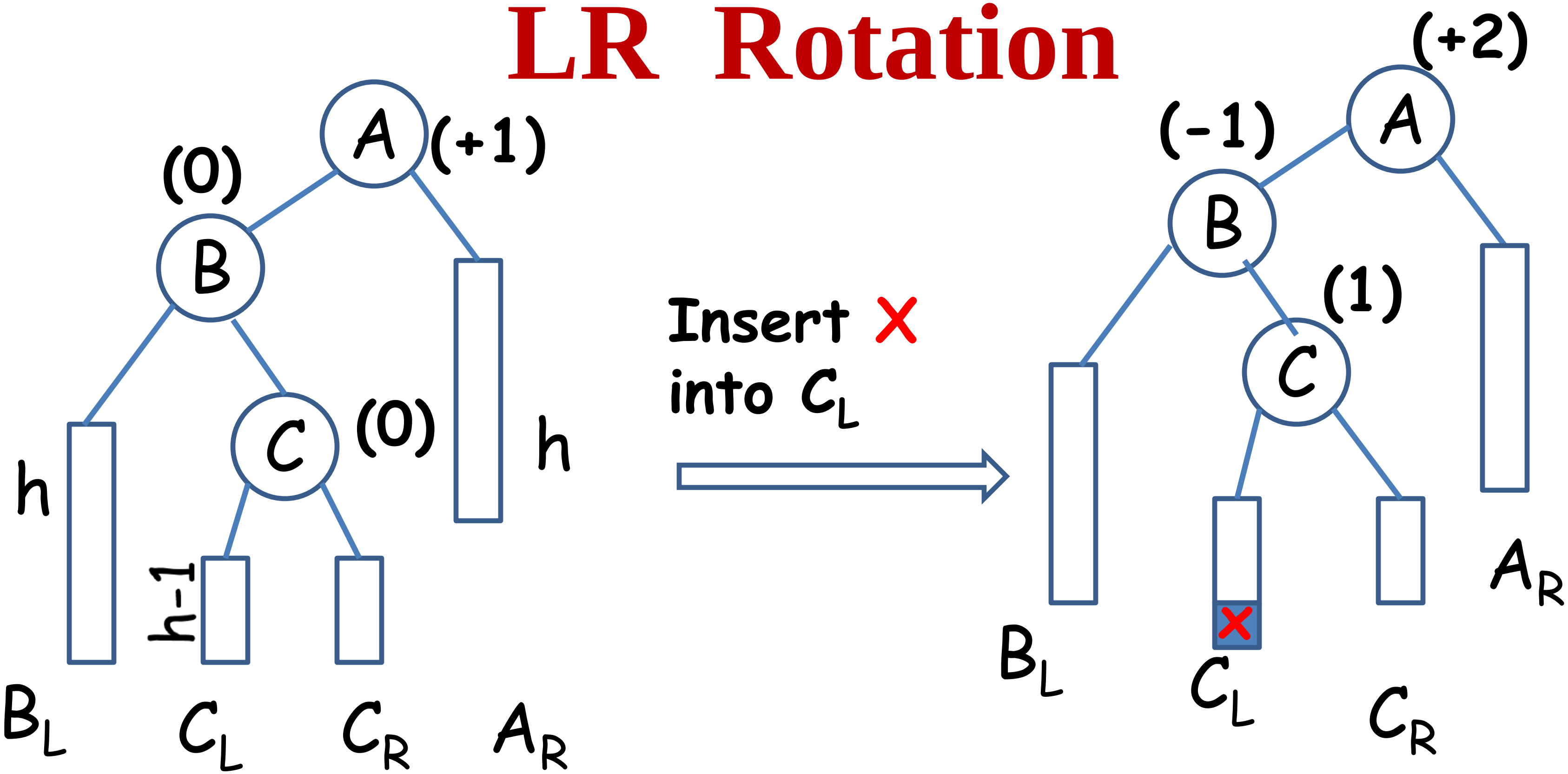
Unbalanced AVL search tree

RR Rotation Example



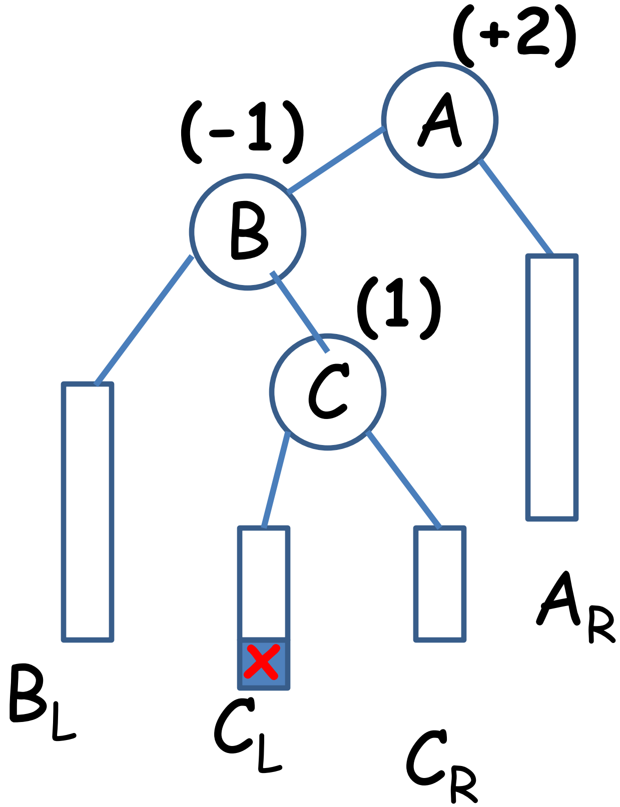
Balanced AVL search tree
after RR rotation

LR Rotation

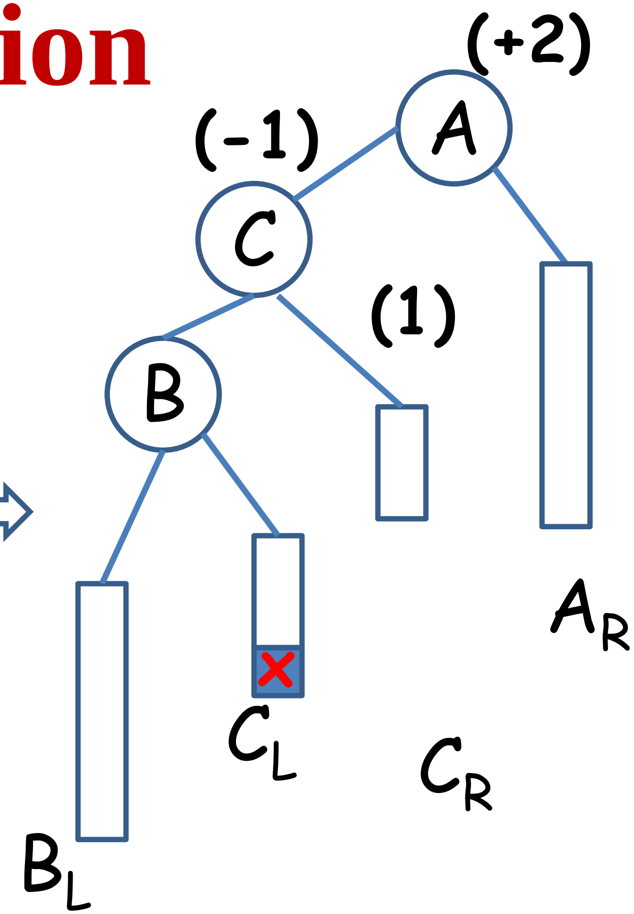


Unbalanced AVL search tree after insertion

LR Rotation

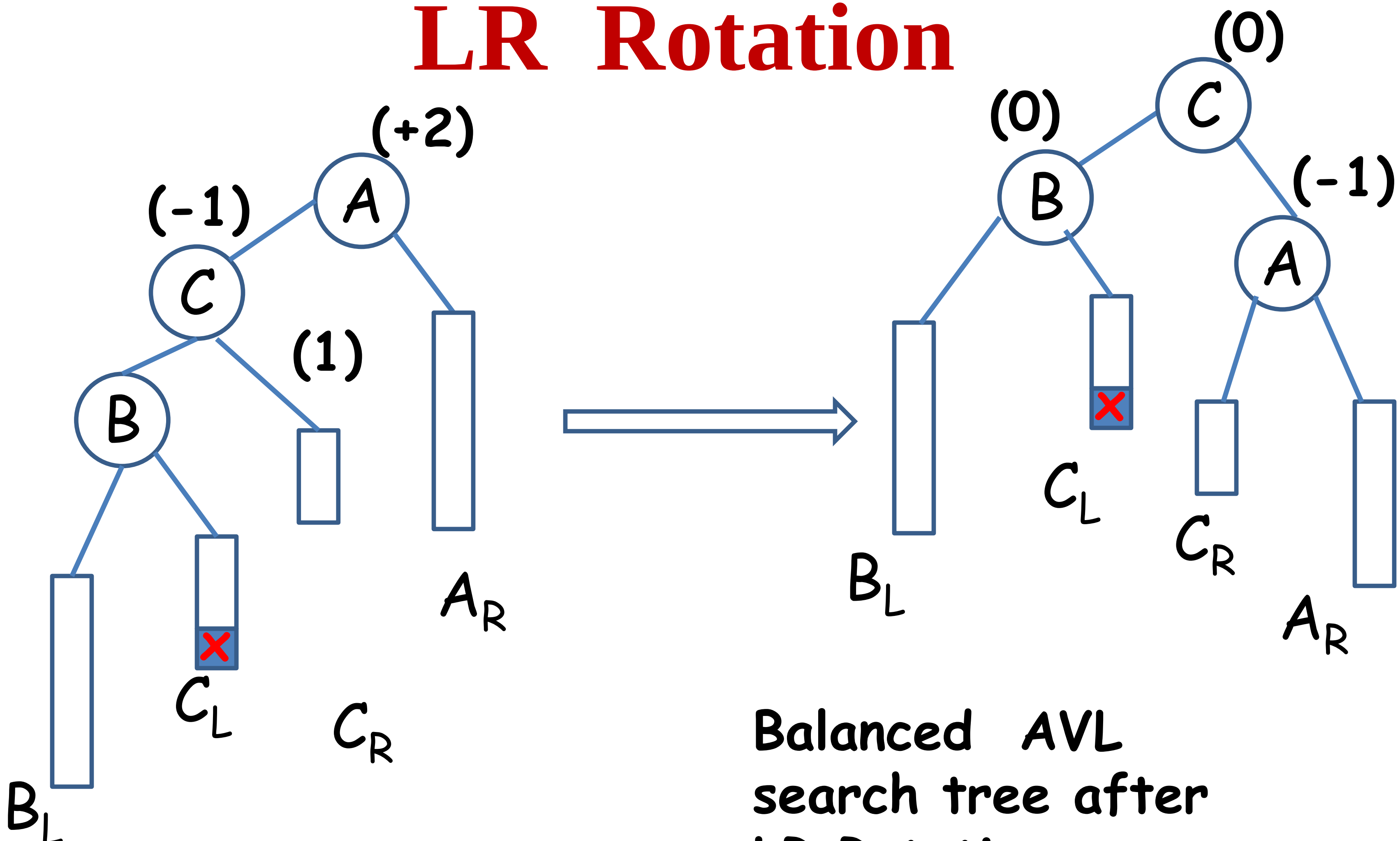


Unbalanced AVL search tree after insertion



AVL tree after Left rotation at left subtree

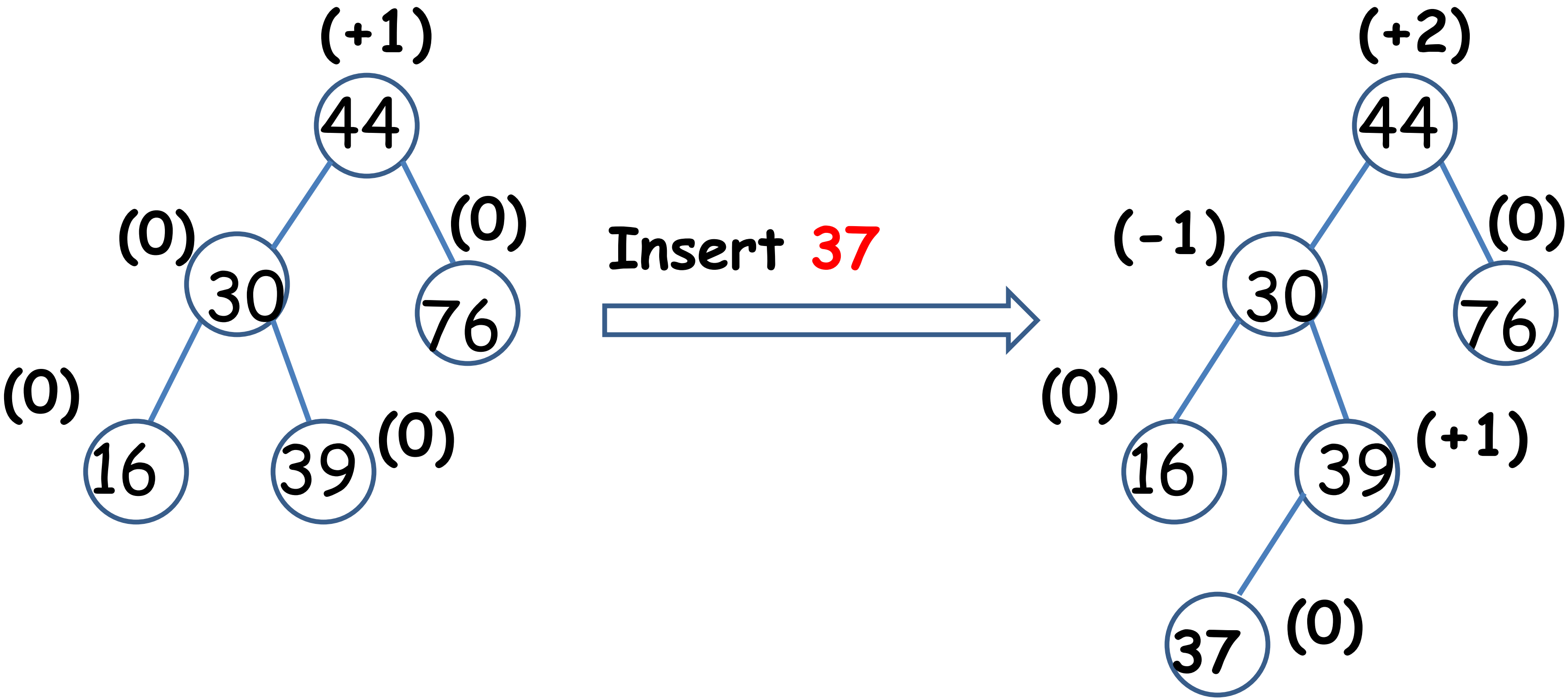
LR Rotation



AVL tree after Left rotation at left subtree

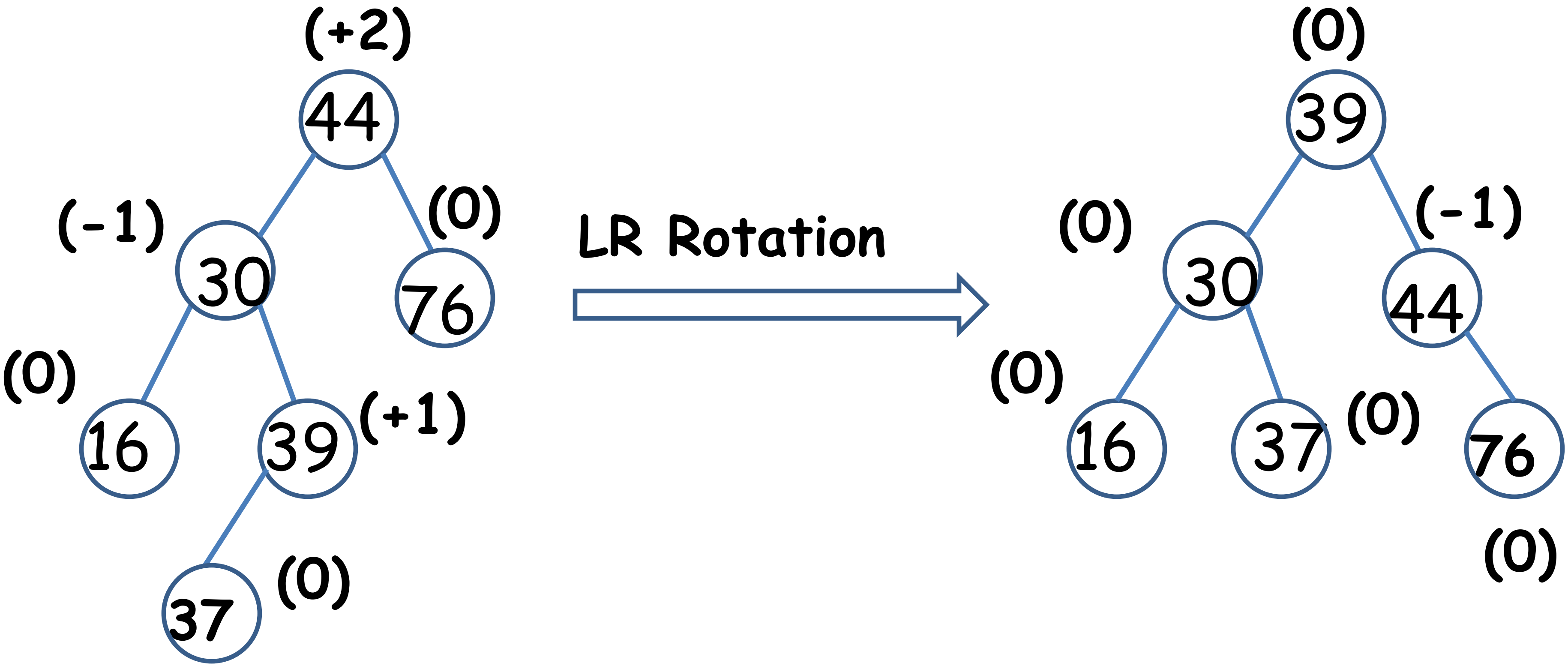
Balanced AVL search tree after LR Rotation

LR Rotation Example



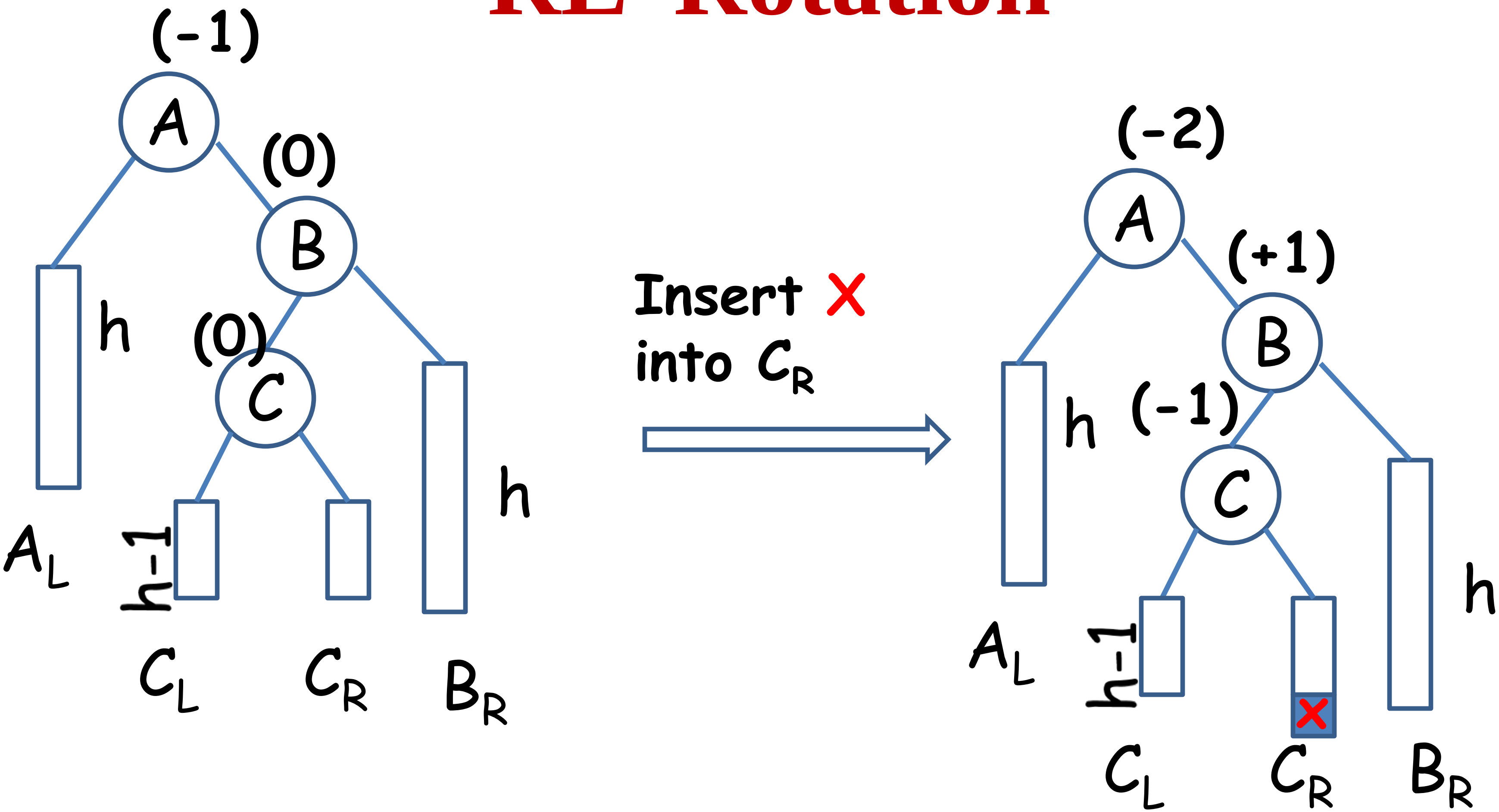
Unbalanced AVL search tree

LR Rotation Example



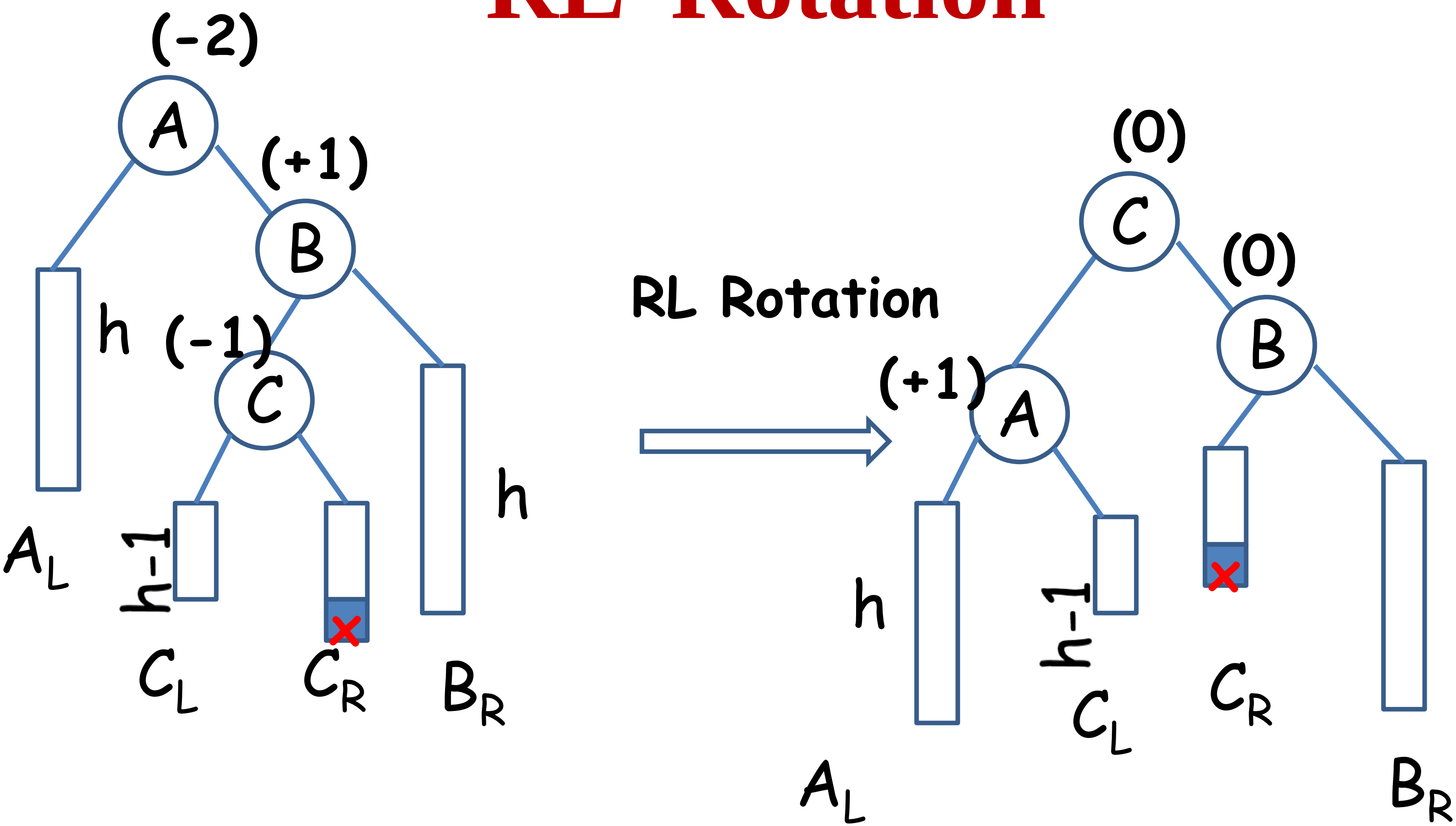
Balanced AVL search tree

RL Rotation



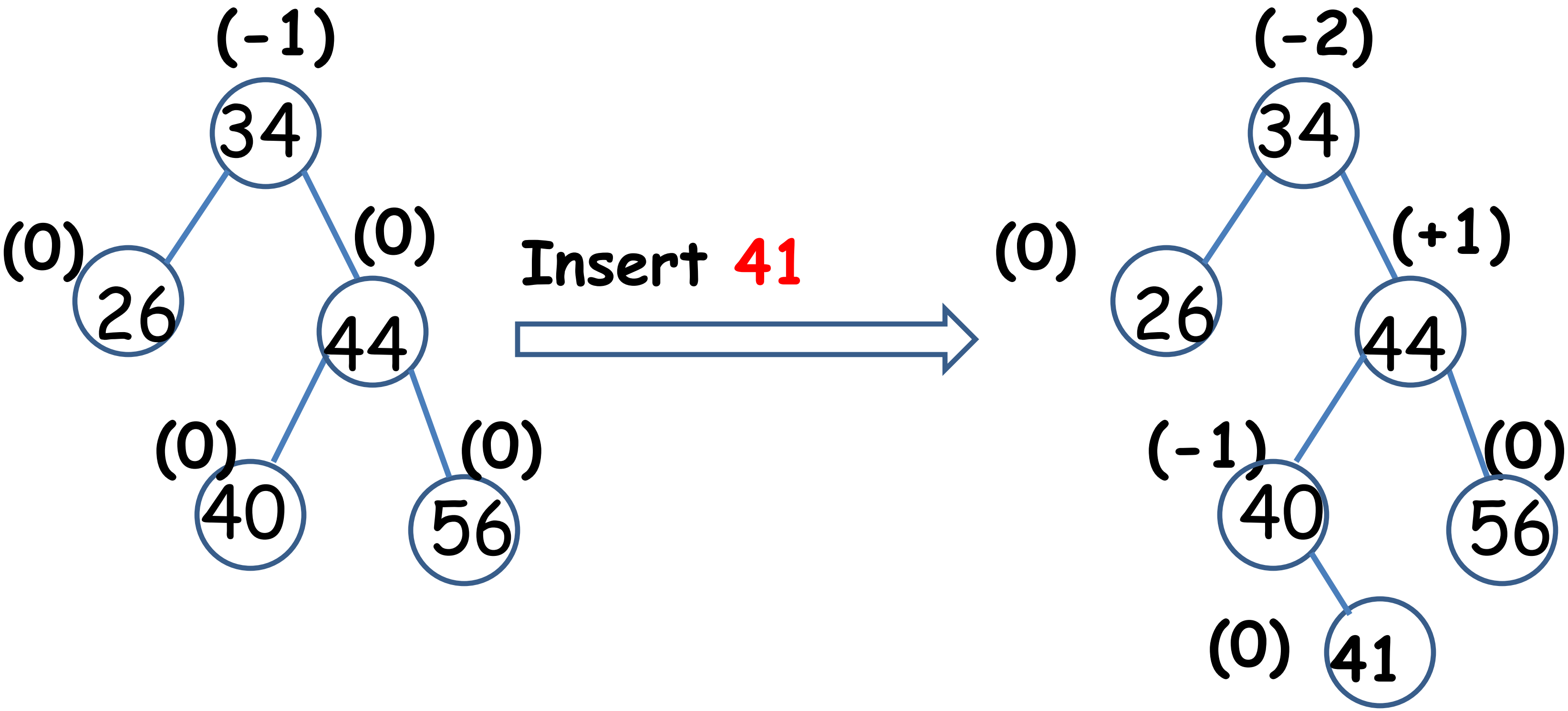
Unbalanced AVL search tree after insertion

RL Rotation



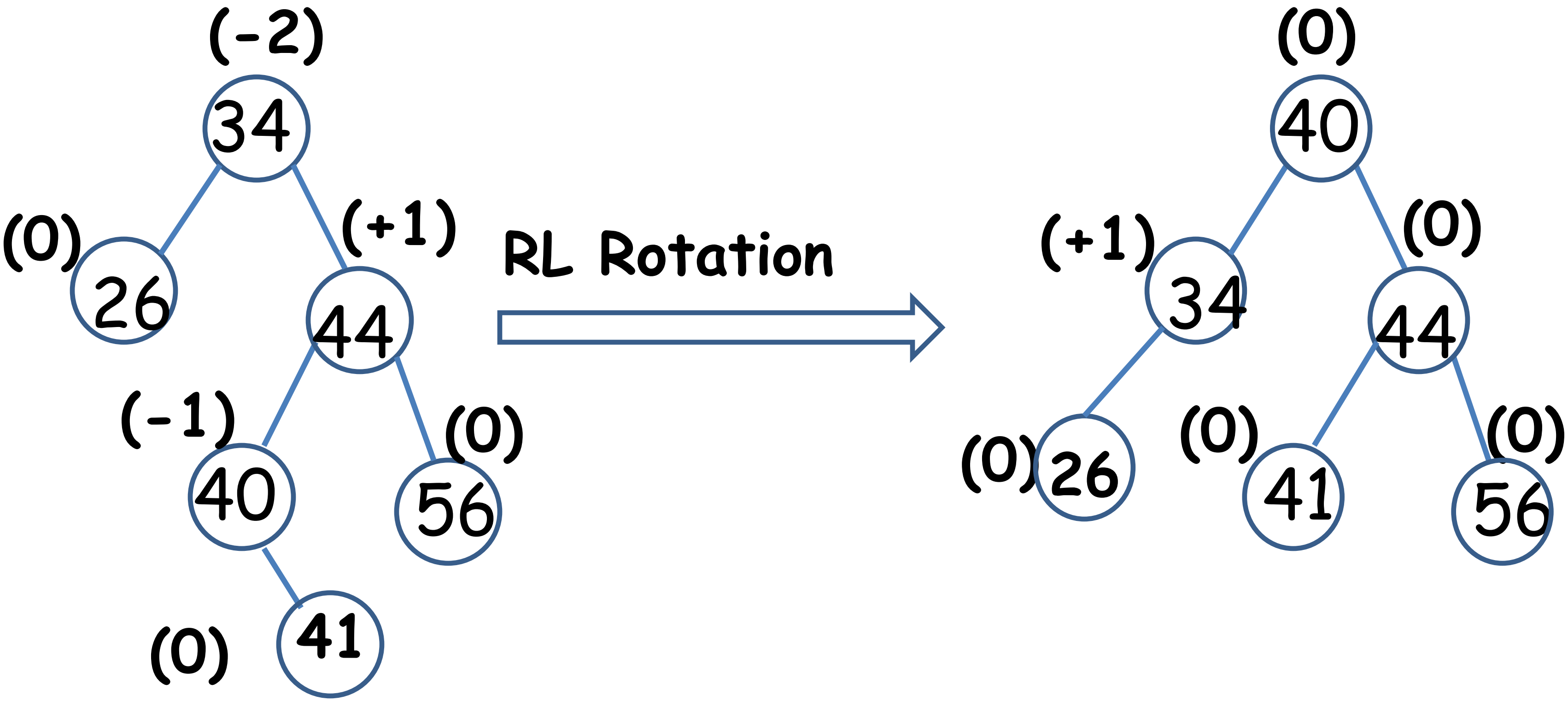
Balanced AVL search tree after RL Rotation

RL Rotation Example



Unbalanced AVL search tree

RL Rotation Example



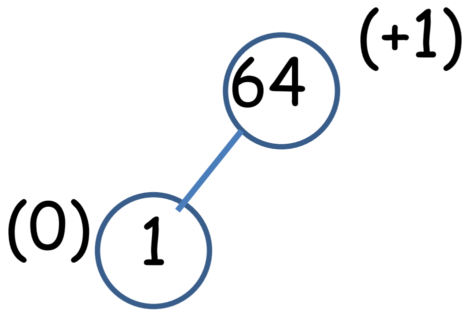
Balanced AVL search tree

AVL Tree

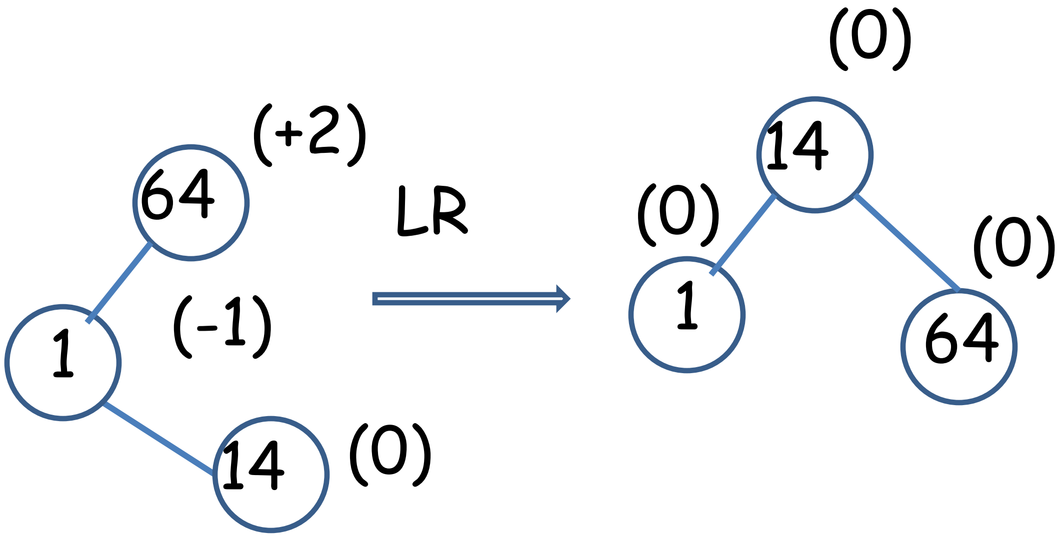
Construct an AVL search tree by inserting the following elements in the order of their occurrence

64, 1, 14, 26, 13, 110, 98, 85

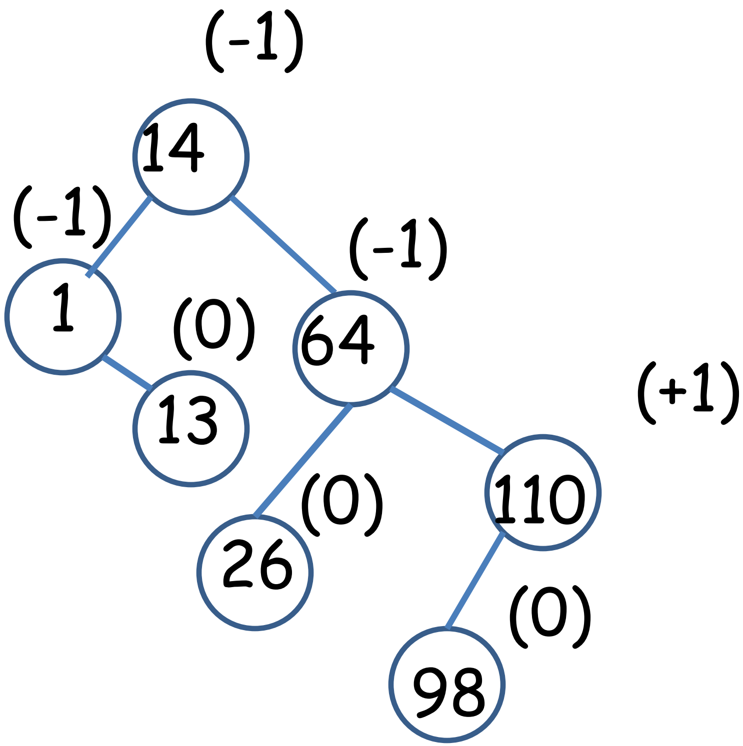
Insert 64, 1

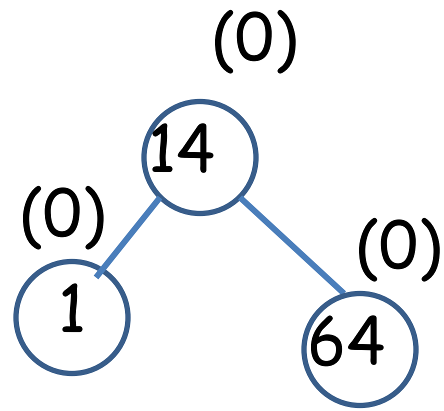


Insert 14

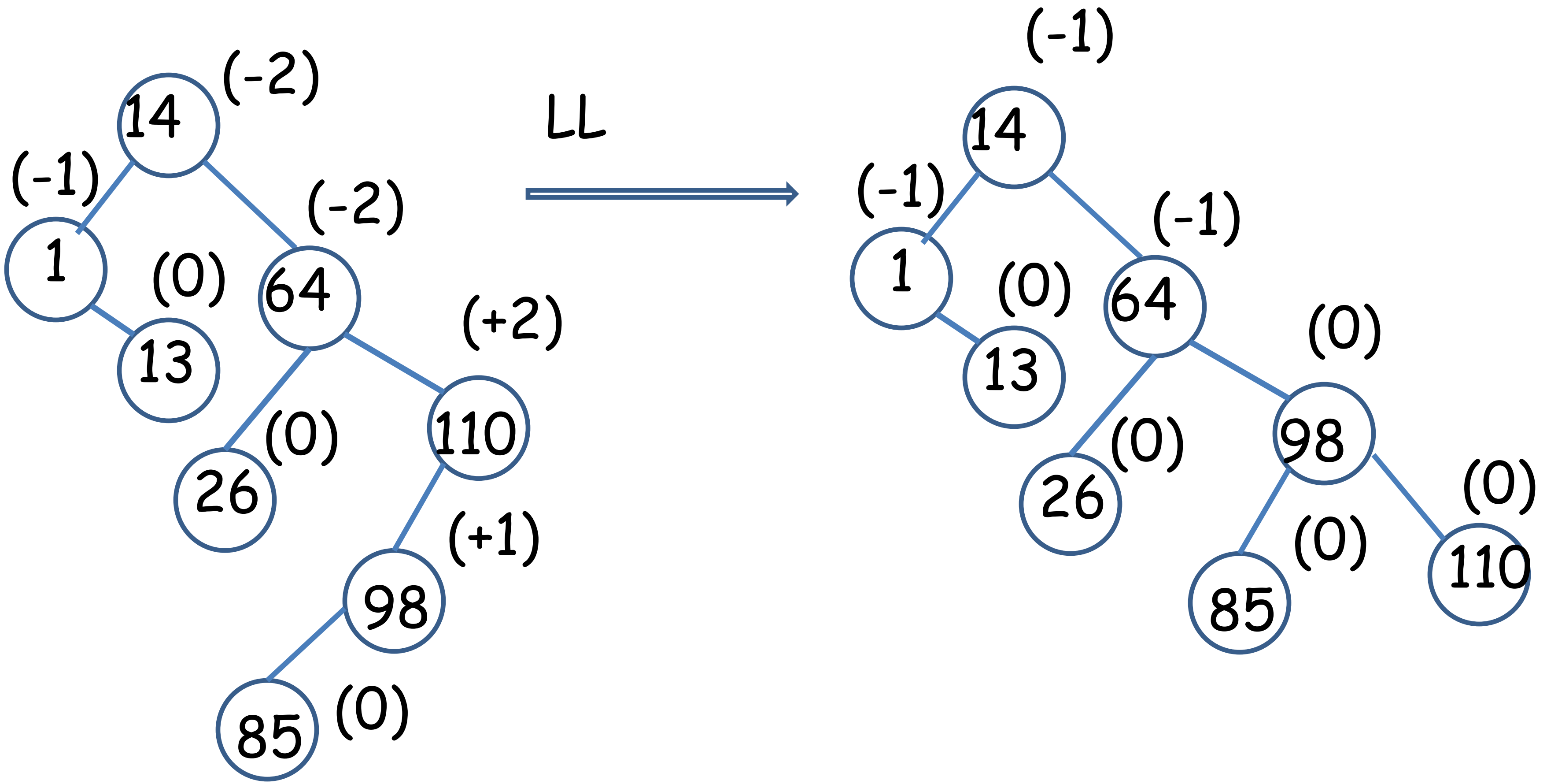


Insert 26, 13, 110,98





Insert 85



14
(-2)
(-1)
1
(0)
13
64
(-2)
26
(0)
110
(+2)
98
(+1)
85
(0)
LL
14
(-1)
(-1)
1
(0)
13
64
(-1)
26
(0)
98
(0)
85
(0)
110
(0)

Deletion in AVL search Tree

Deletion in AVL search tree proceed the same way as the deletion in binary search tree

However, in the event of imbalance due to deletion, one or more rotation need to be applied to balance the AVL tree.

AVL deletion

Let **A** be the closest ancestor node on the path from **X** (deleted node) to the root with a balancing factor +2 or -2

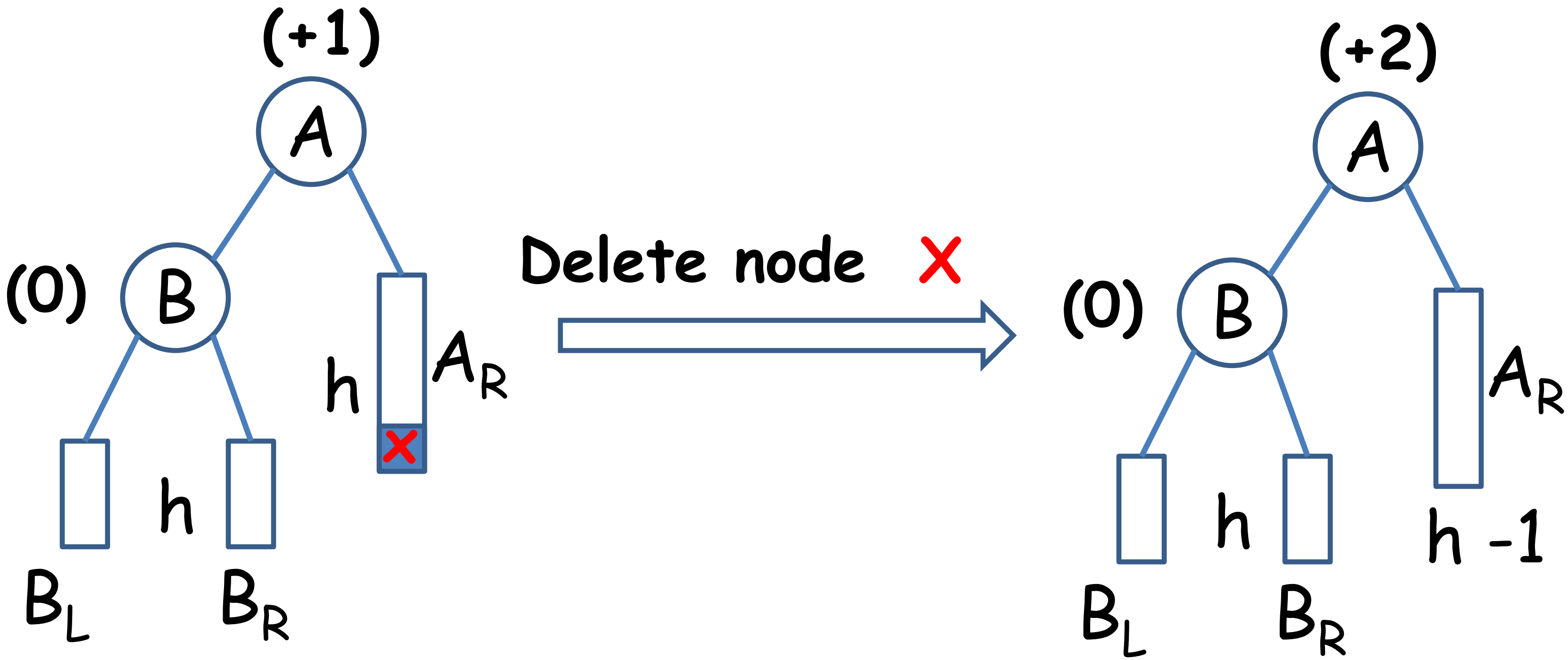
Classify the rotation as **L** or **R** depending on whether the deletion occurred on the left or right subtree of **A**

AVL Deletion

Depending on the value of **BF(B)** where **B** is the root of the right or left subtree of **A**, the R or L imbalance is further classified as R0, R1 and R -1 or L0, L1 and L-1.

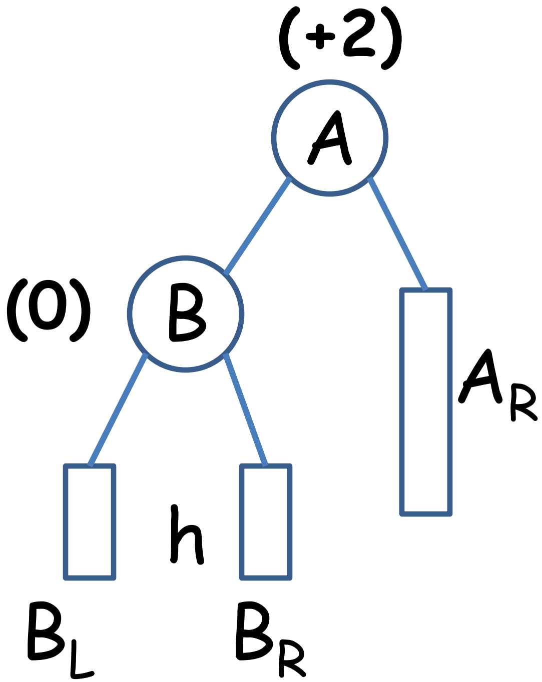
Deletion type -> Applicable rotation	Deletion type -> Applicable rotation
R0 -> LL	L0 -> RR
R1 -> LL	L1 -> RL
R-1 -> LR	L-1 -> RR

R0 Rotation



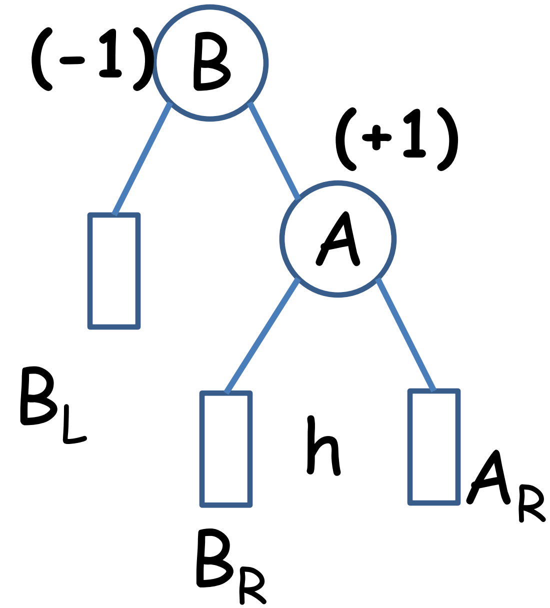
Unbalanced AVL search tree after deletion of node x

R0 Rotation



R0 Rotation

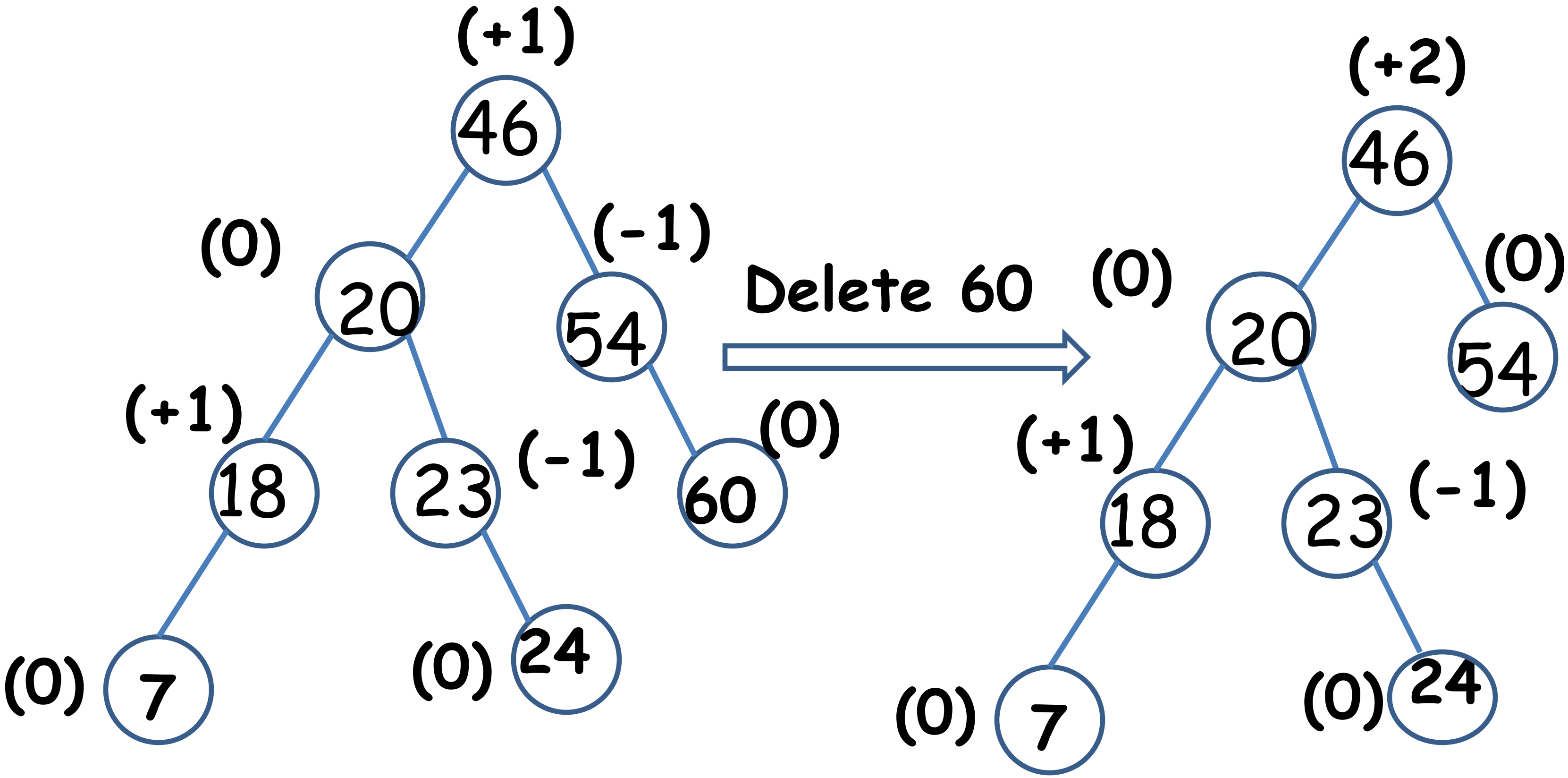
BF(B) == 0, use R0 rotation



Unbalanced AVL search tree after deletion of x

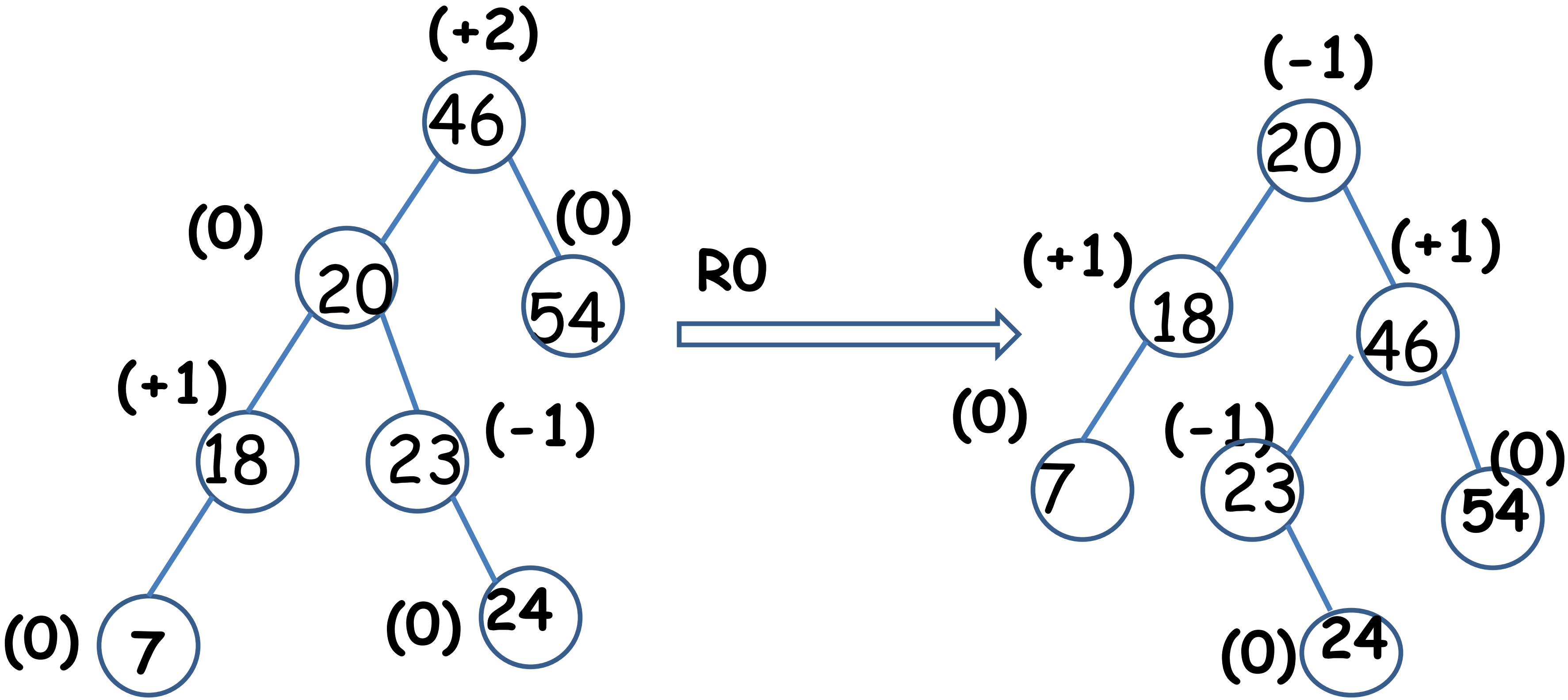
Balanced AVL search tree after rotation

R0 Rotation Example



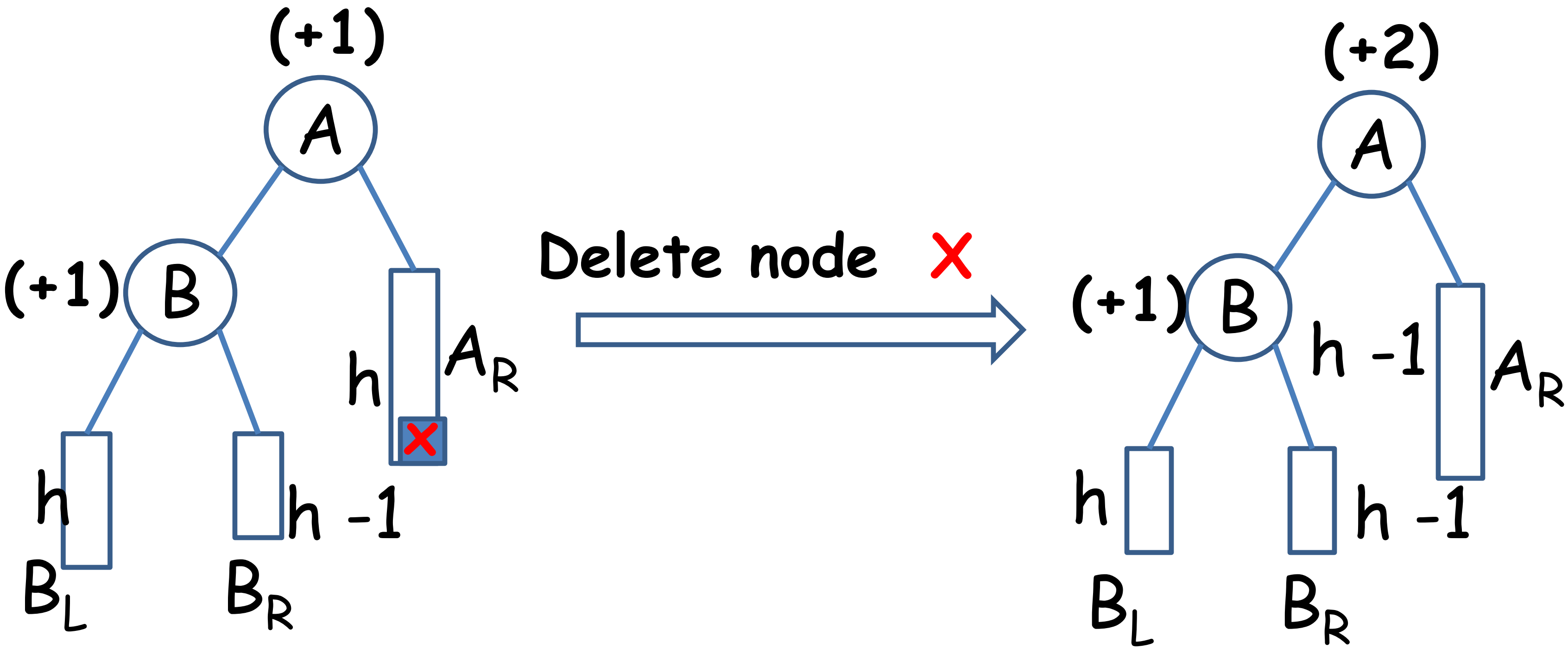
Unbalanced AVL search tree after deletion

R0 Rotation Example



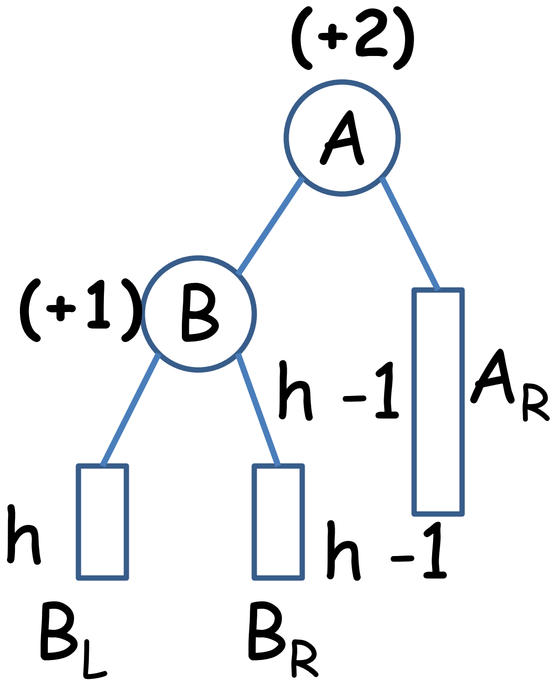
Balanced AVL search tree after deletion

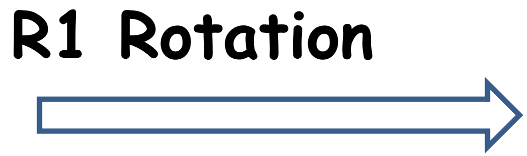
R1 Rotation

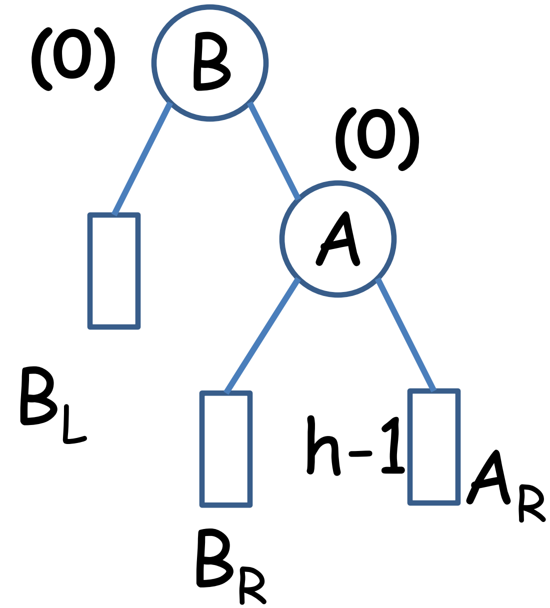


Unbalanced AVL search tree after deletion of node x

R1 Rotation



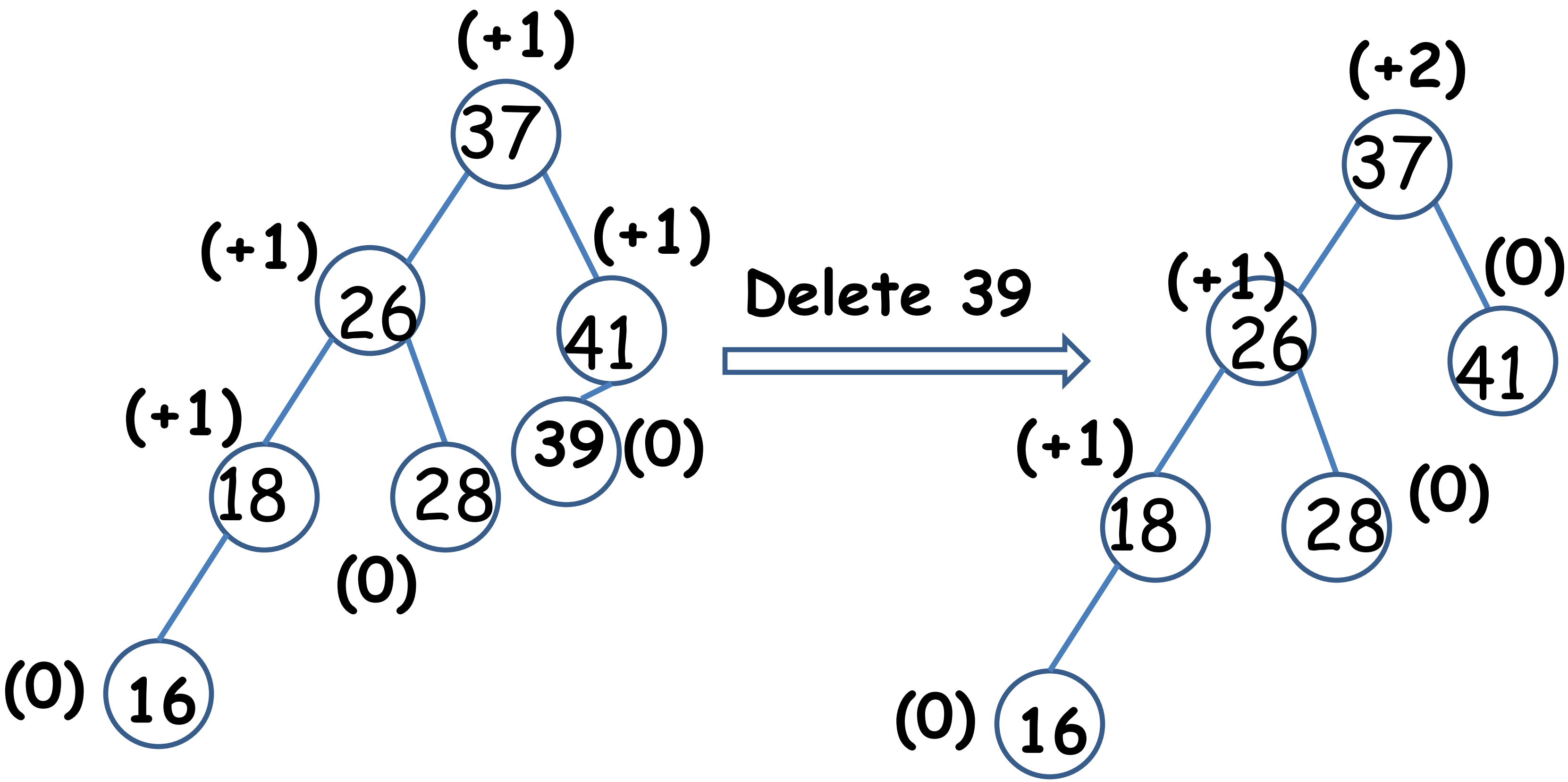




BF(B) == 1, use R1 rotation

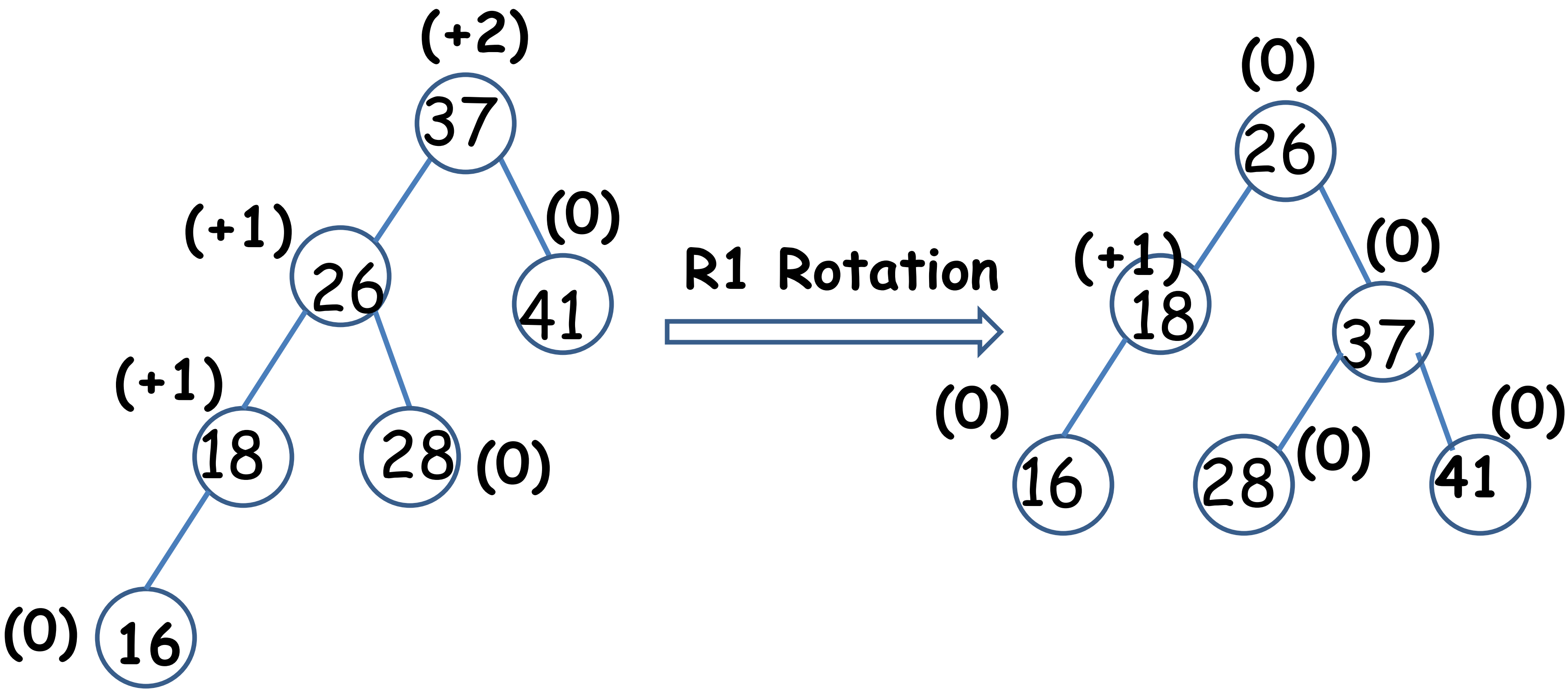
Balanced AVL search tree after rotation

R1 Rotation Example



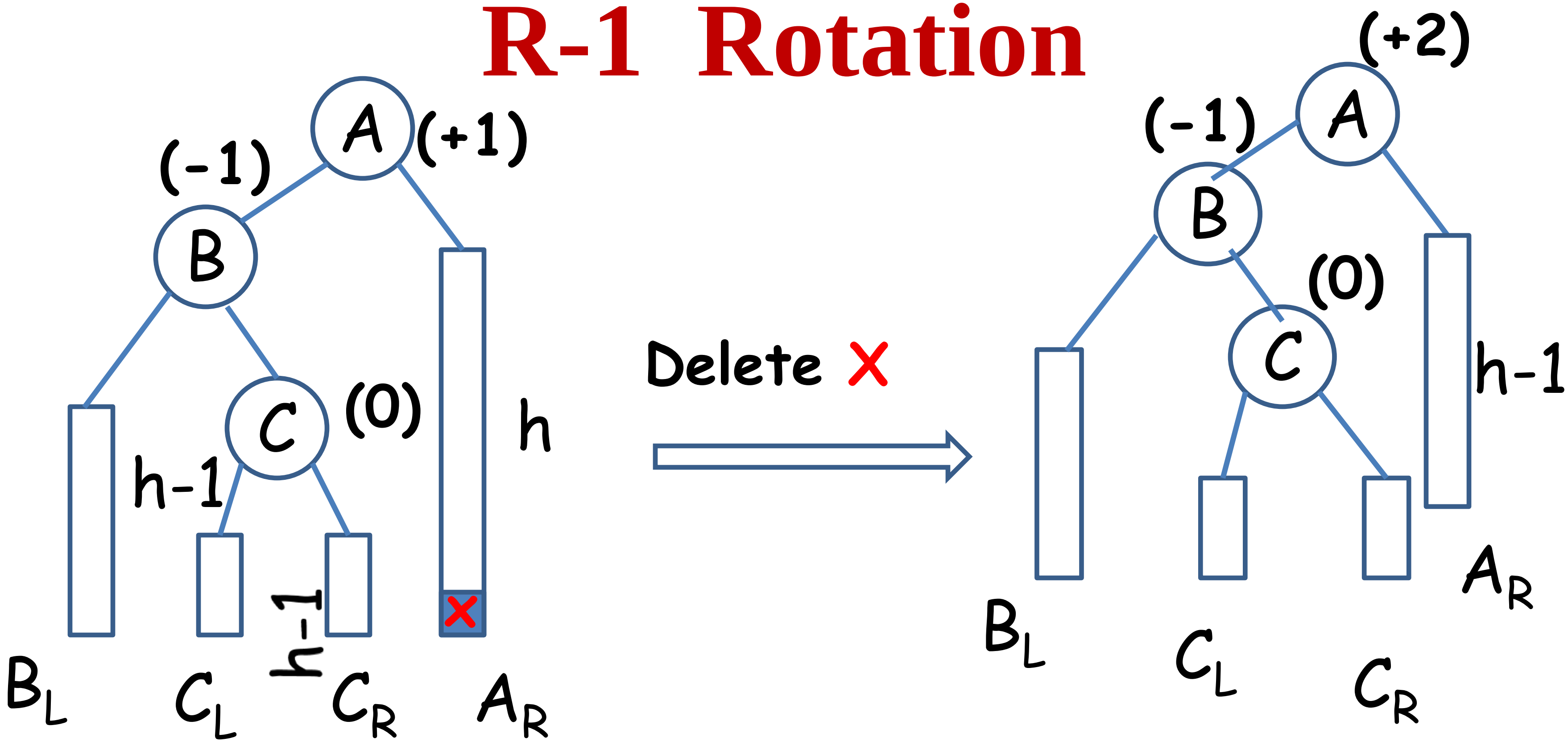
Unbalanced AVL search tree after deletion

R1 Rotation Example



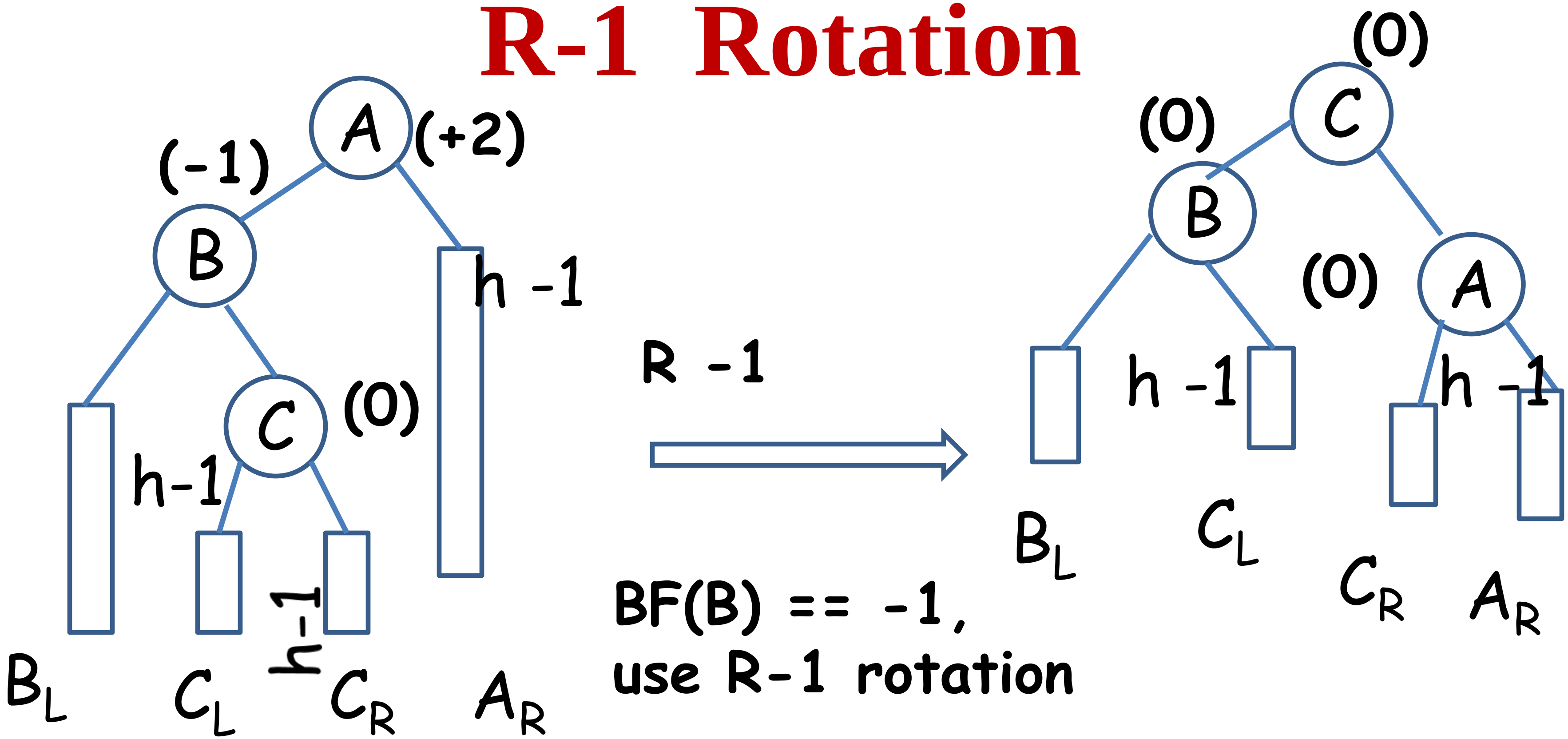
Balanced AVL search tree after deletion

R-1 Rotation



Unbalanced AVL search tree after deletion

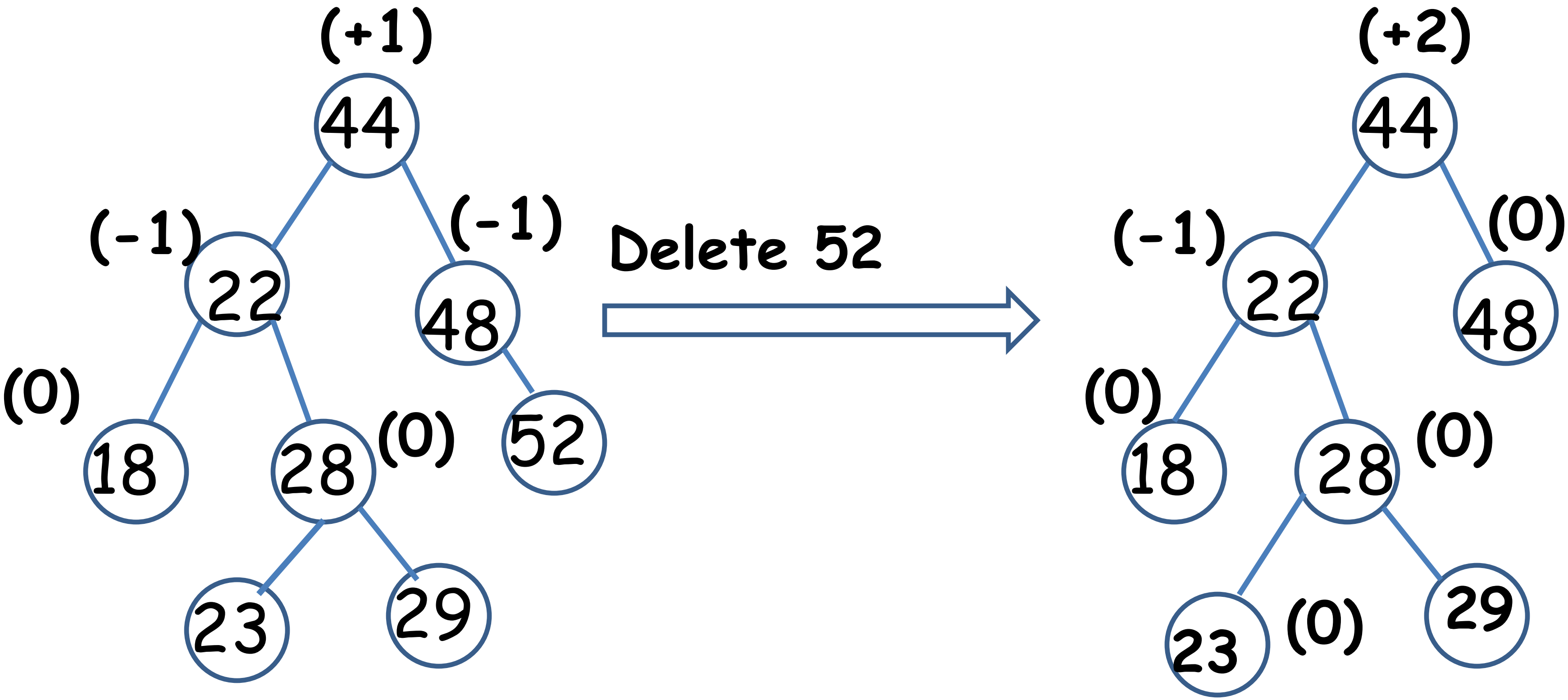
R-1 Rotation



BF(B) == -1,
use R-1 rotation

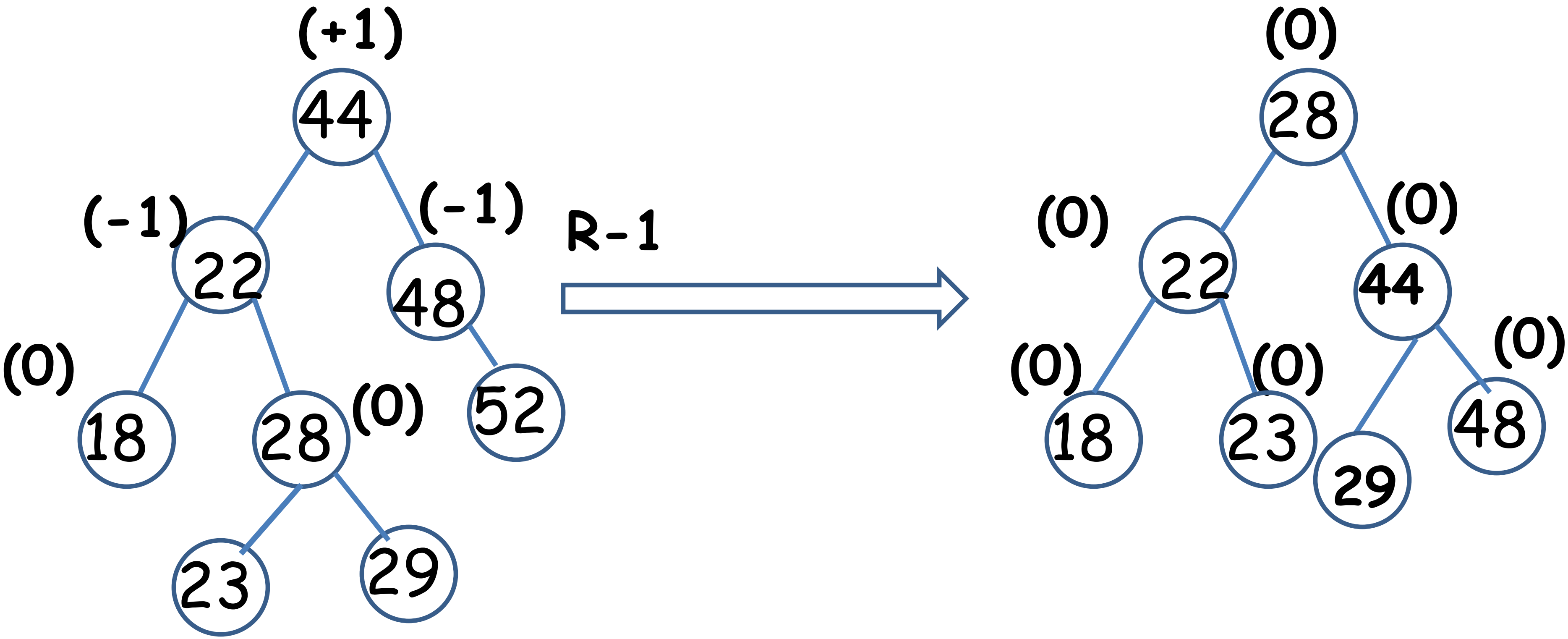
Balanced AVL
search tree after
Rotation

R-1 Rotation Example



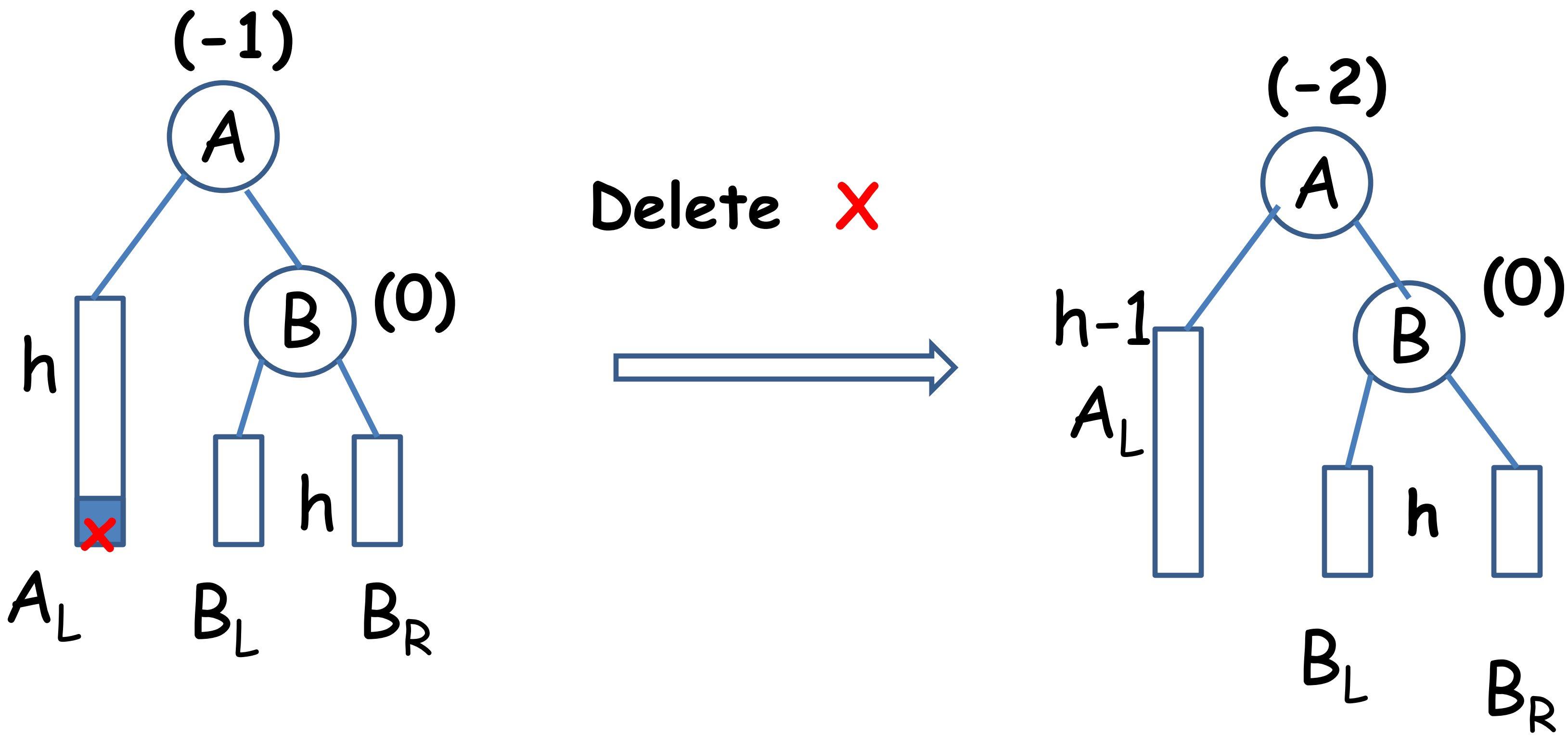
Unbalanced AVL search tree after deletion

R-1 Rotation Example



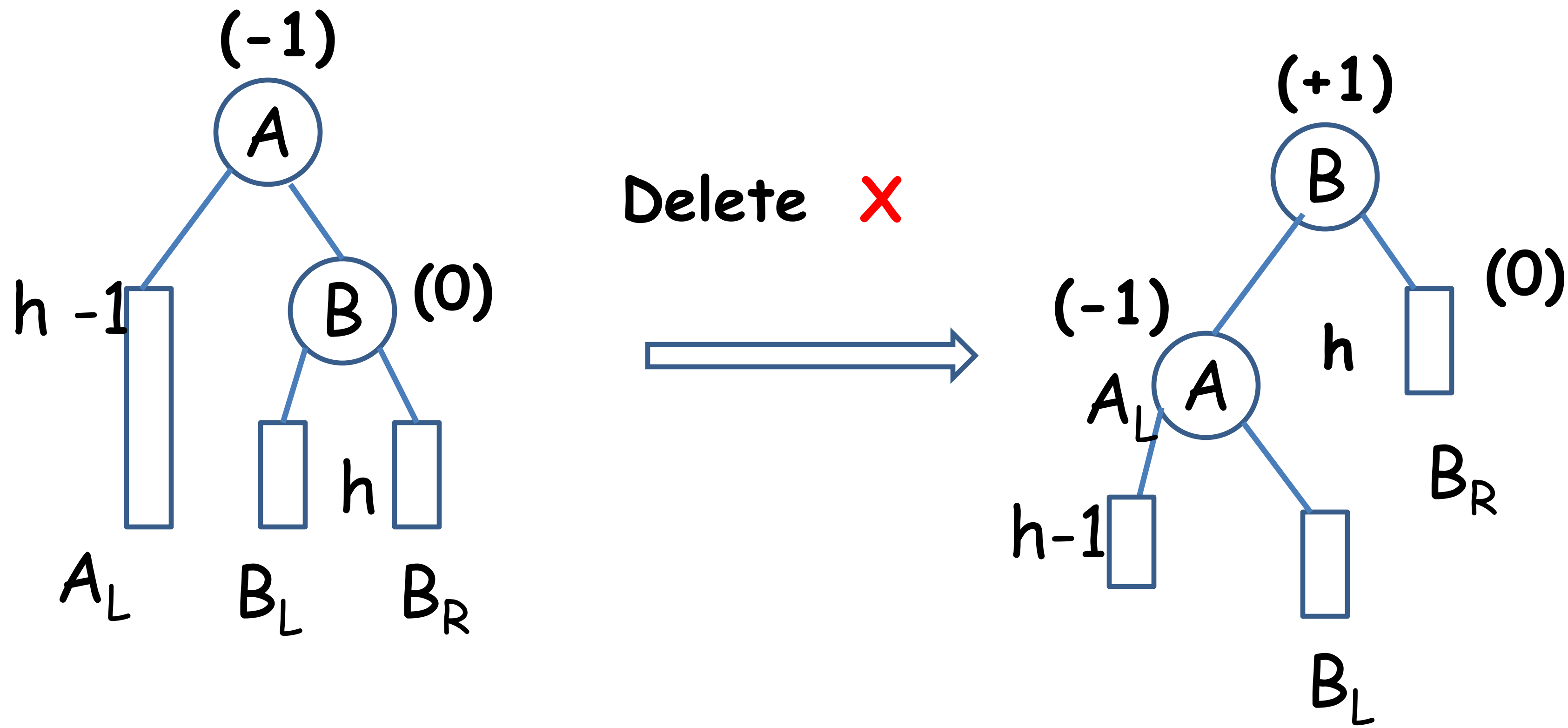
Balanced AVL search tree
after rotation

L0 Rotation



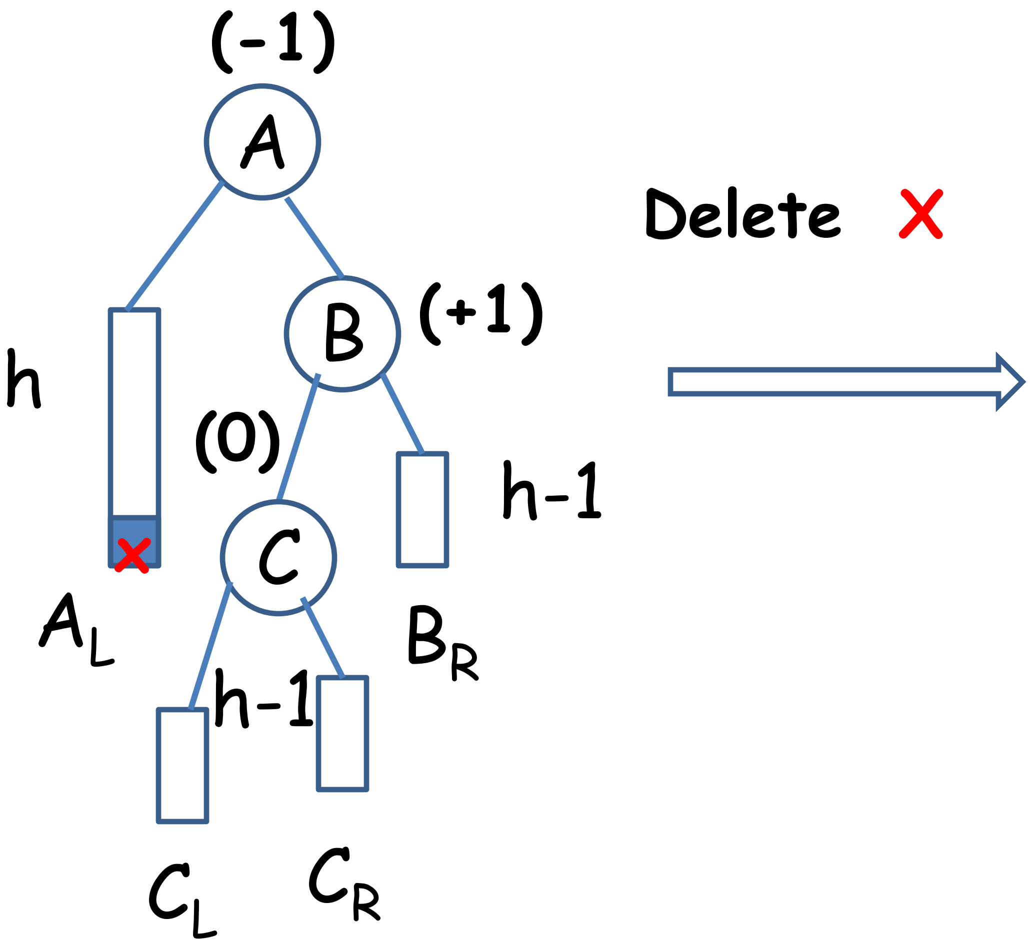
Unbalanced AVL search tree after deletion

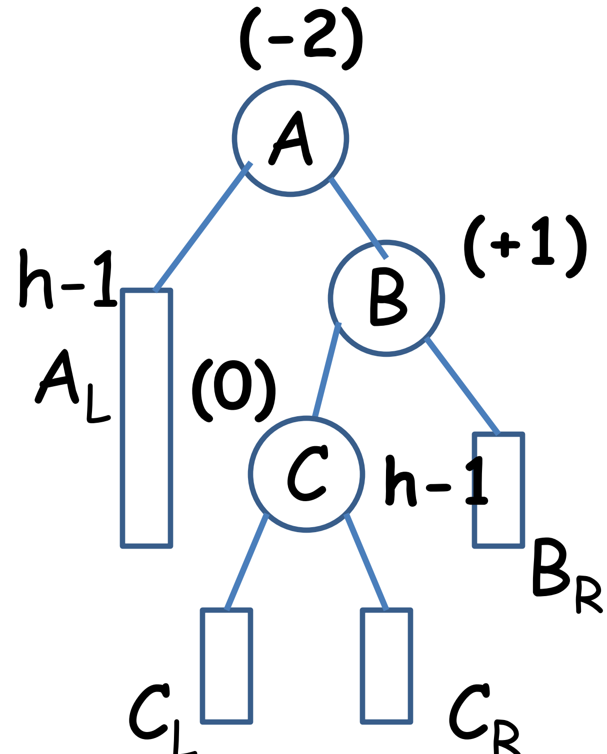
L0 Rotation



Balanced AVL search tree after deletion

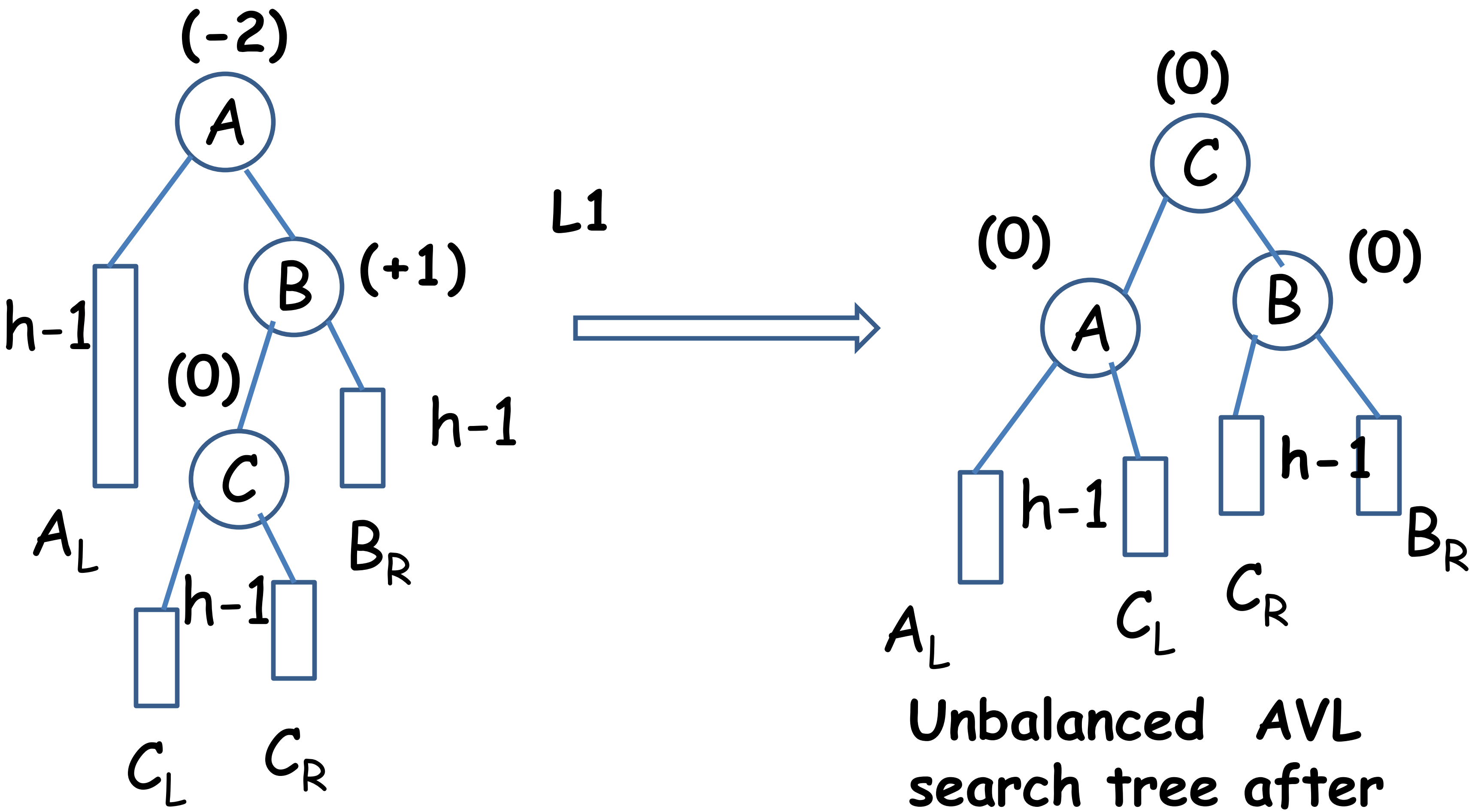
L1 Rotation





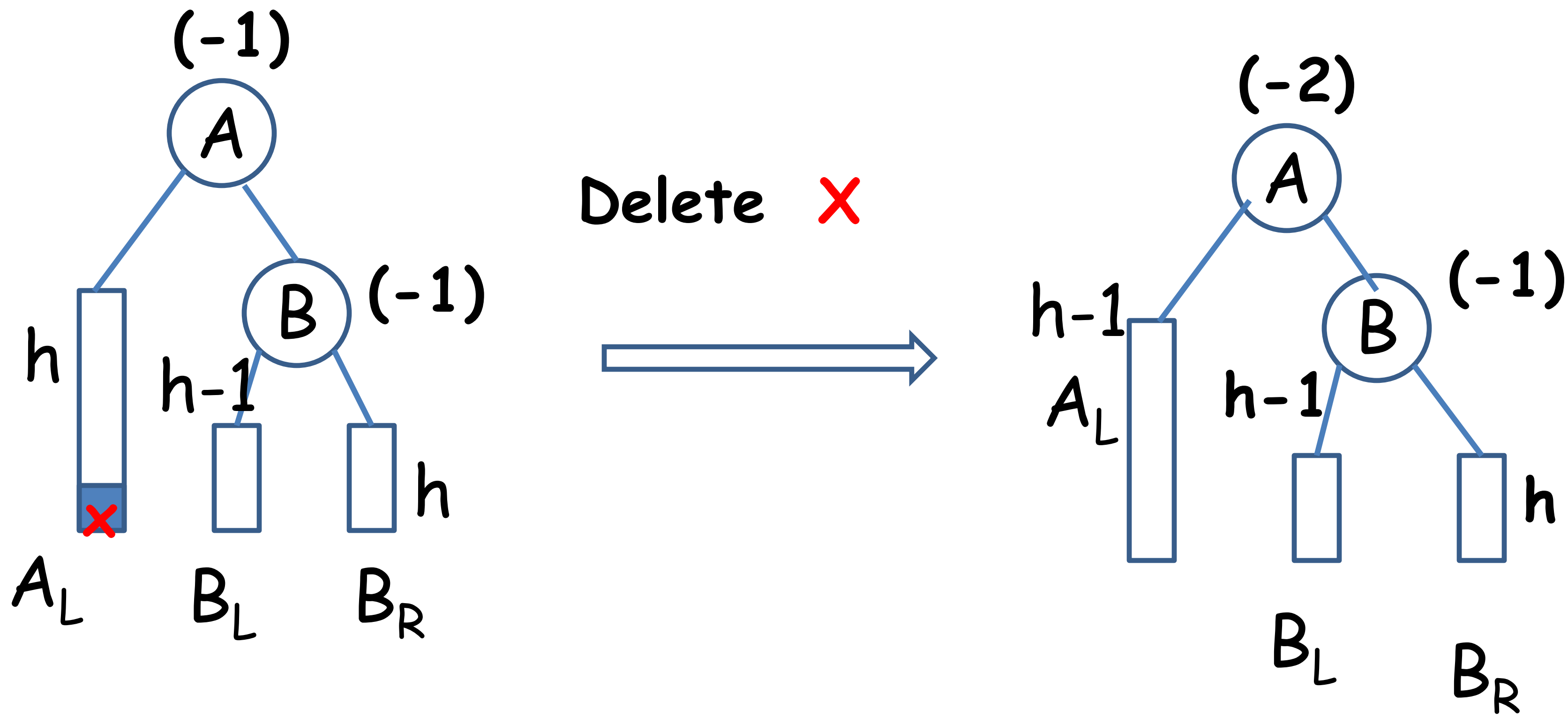
Unbalanced AVL search tree after deletion

L1 Rotation



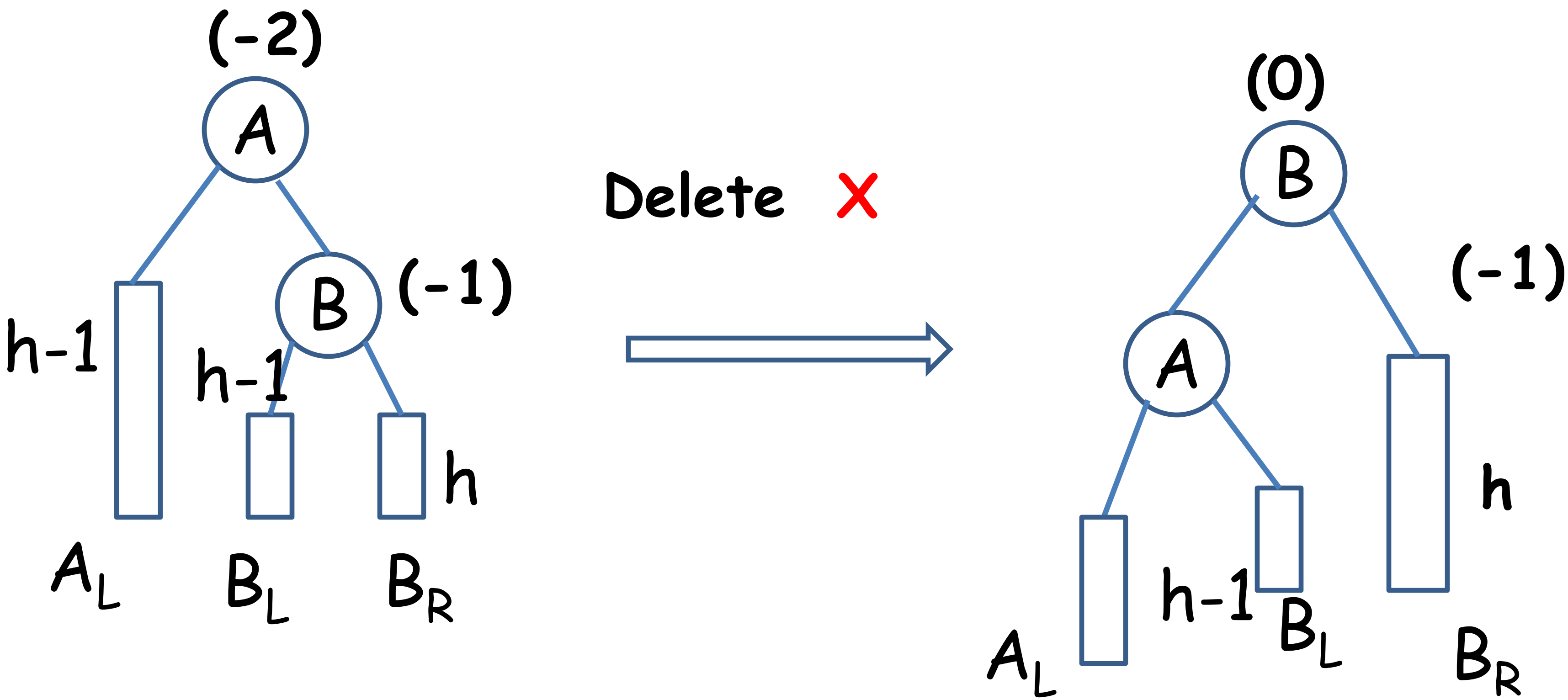
(-2)
A
(+1)
B
(0)
C
h-1
h-1
h-1
A_L
B_R
C_L
C_R
L1
(0)
C
(0)
A
(0)
B
h-1
h-1
A_L
C_L
C_R
B_R
Unbalanced AVL search tree after deletion

L-1 Rotation



Unbalanced AVL search tree after deletion

L-1 Rotation



Balanced AVL search tree after deletion