DATA STRUCTURES AND ITS APPLICATIONS

Balanced Trees

Sandesh B. J
Department of Computer Science & Engineering

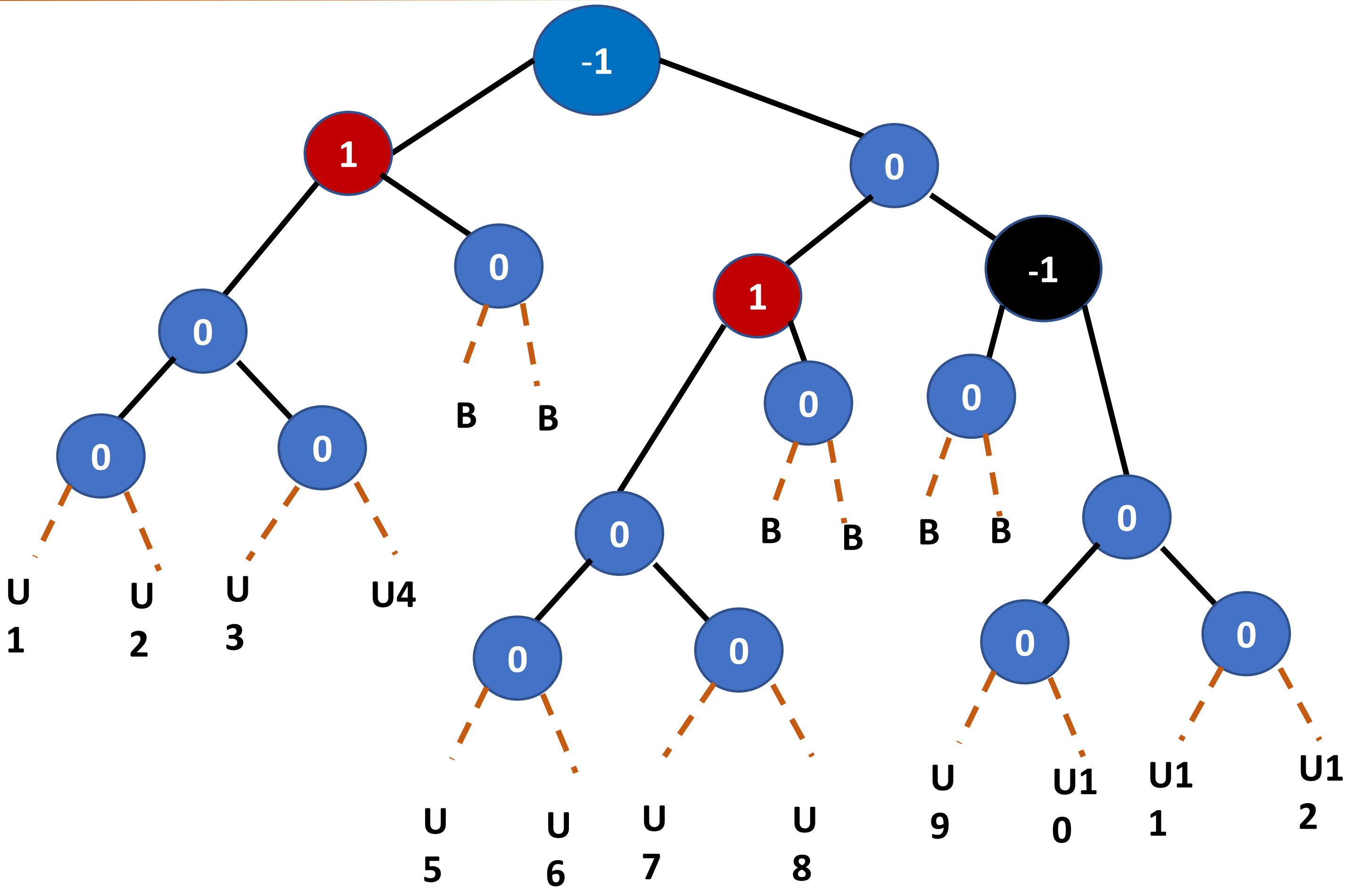
DATA STRUCTURES AND ITS APPLICATIONS

Balanced Trees

Sandesh B. J

Department of Computer Science & Engineering

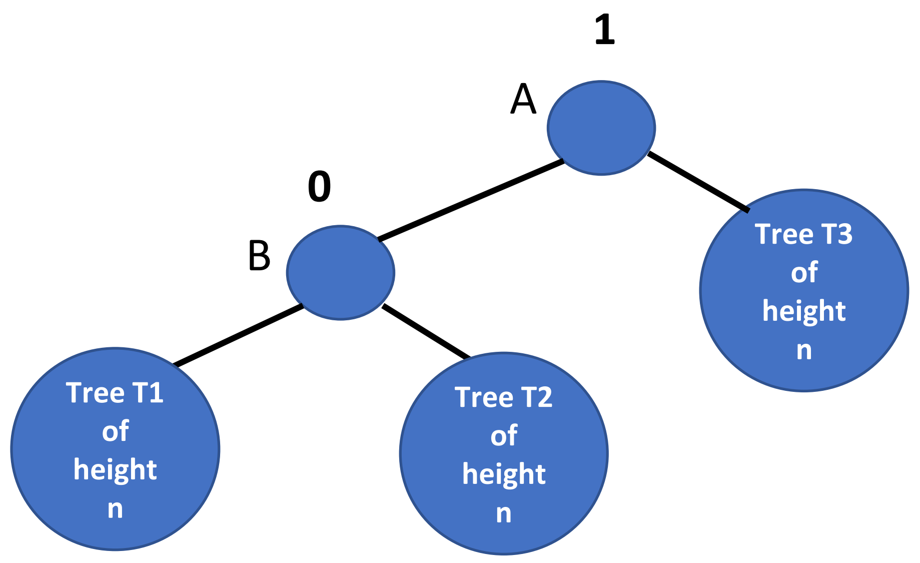
Possible insertion into AVL tree



- Unbalanced insertions are indicated by **U**
- Balanced insertions are indicated by **B**

Courtesy: "Data Structures using c and c++ " By Y Langsam, M. J. Augenstein and Aron M. Tenenbaum

AVL tree Insertions



Balance factor(A) = (n+1) – n =1

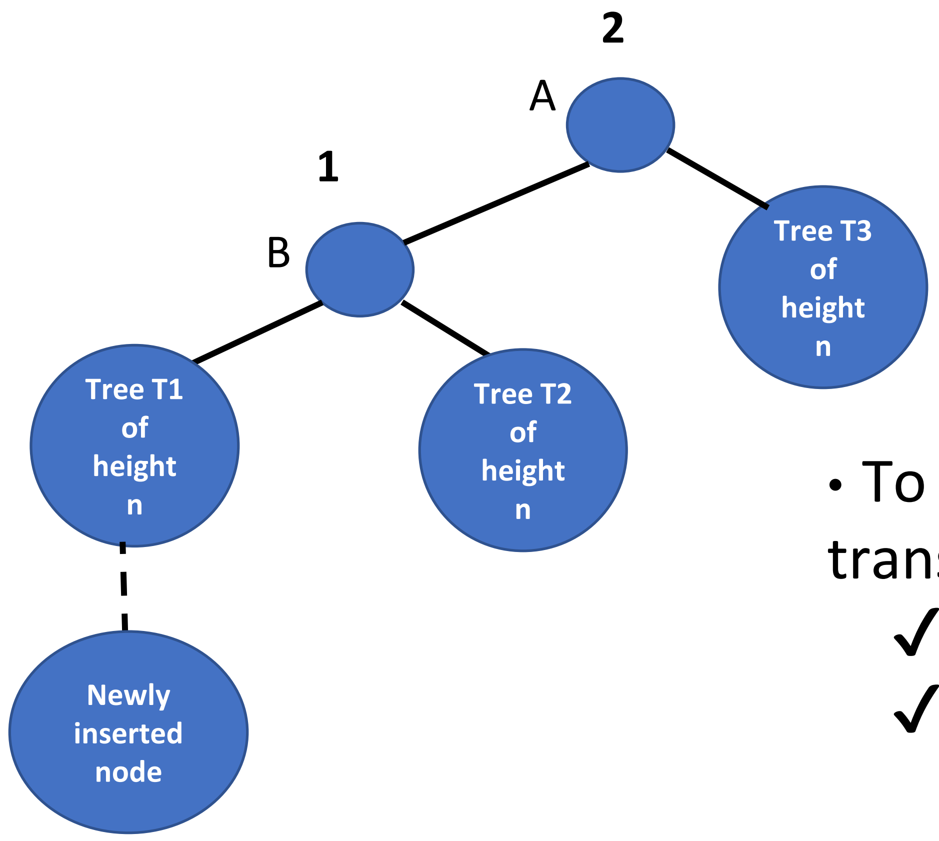
Balance factor(B) = n – n = 0

- Let us consider A is the youngest ancestor which becomes unbalanced
- Balance factor of A should be 1 before insertion
- A should have a left child B with the balance factor of 0

Courtesy: "Data Structures using c and c++ " By Y Langsam, M. J. Augenstein and Aron M. Tenenbaum

Unbalanced Tree after inserting a node to left subtree

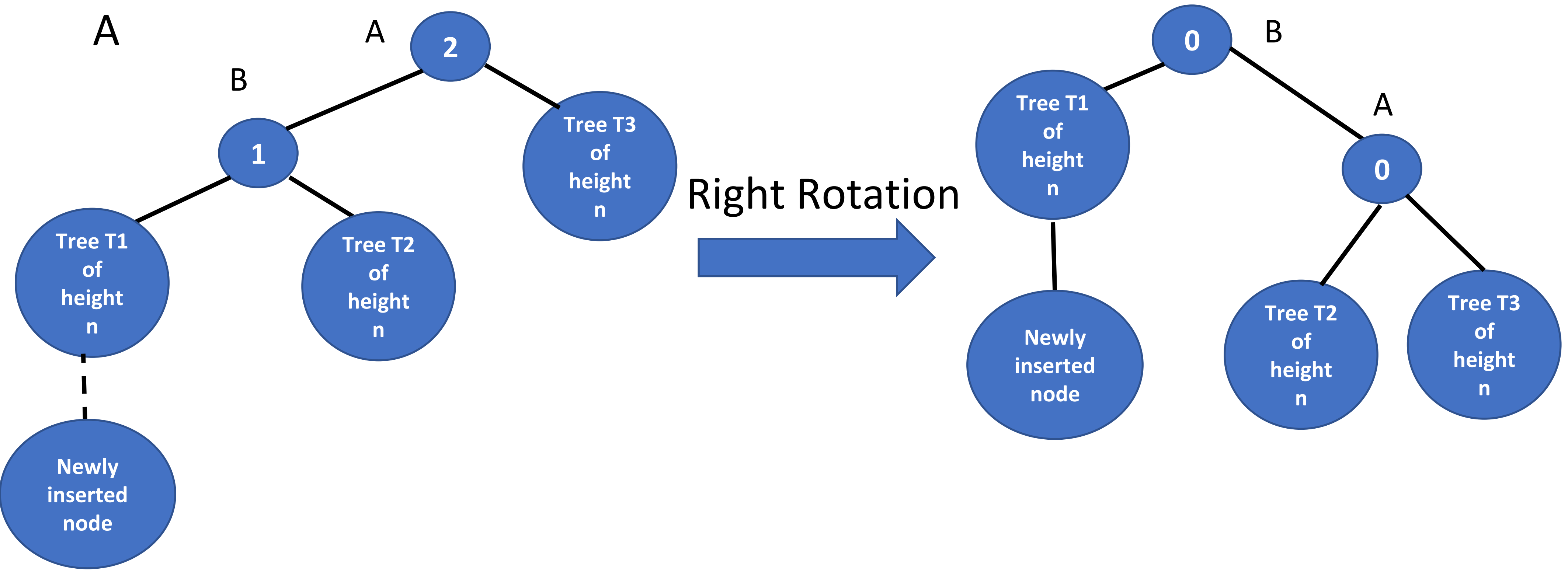
- Newly inserted node is left descendent of node A
- Changing the balance **B to 1 and A to 2**
- A is the youngest ancestor of the new node to become unbalanced



- To maintain the balance : Tree needs to be transformed
  - ✓ Transformed tree is balanced
  - ✓ Binary search tree property is maintained after transformation

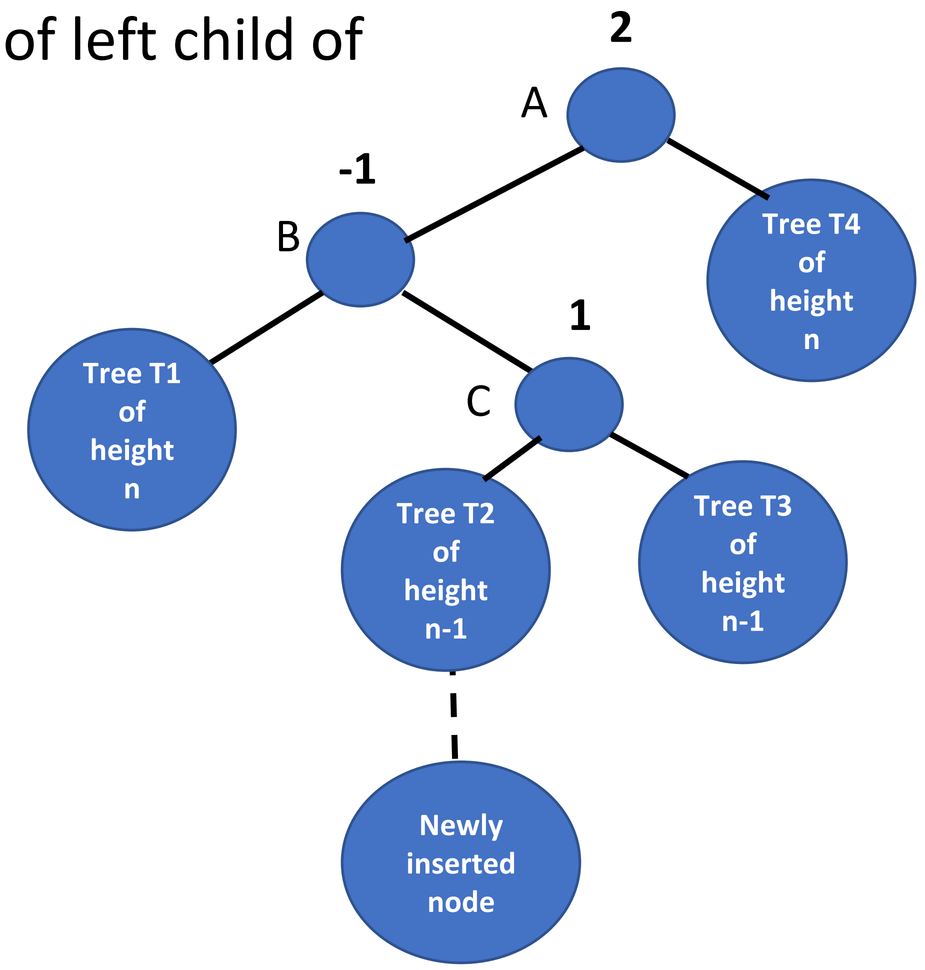
Transformed Balanced Tree after Rotations

- To maintain a balance we need to rotate sub tree B rooted at node A



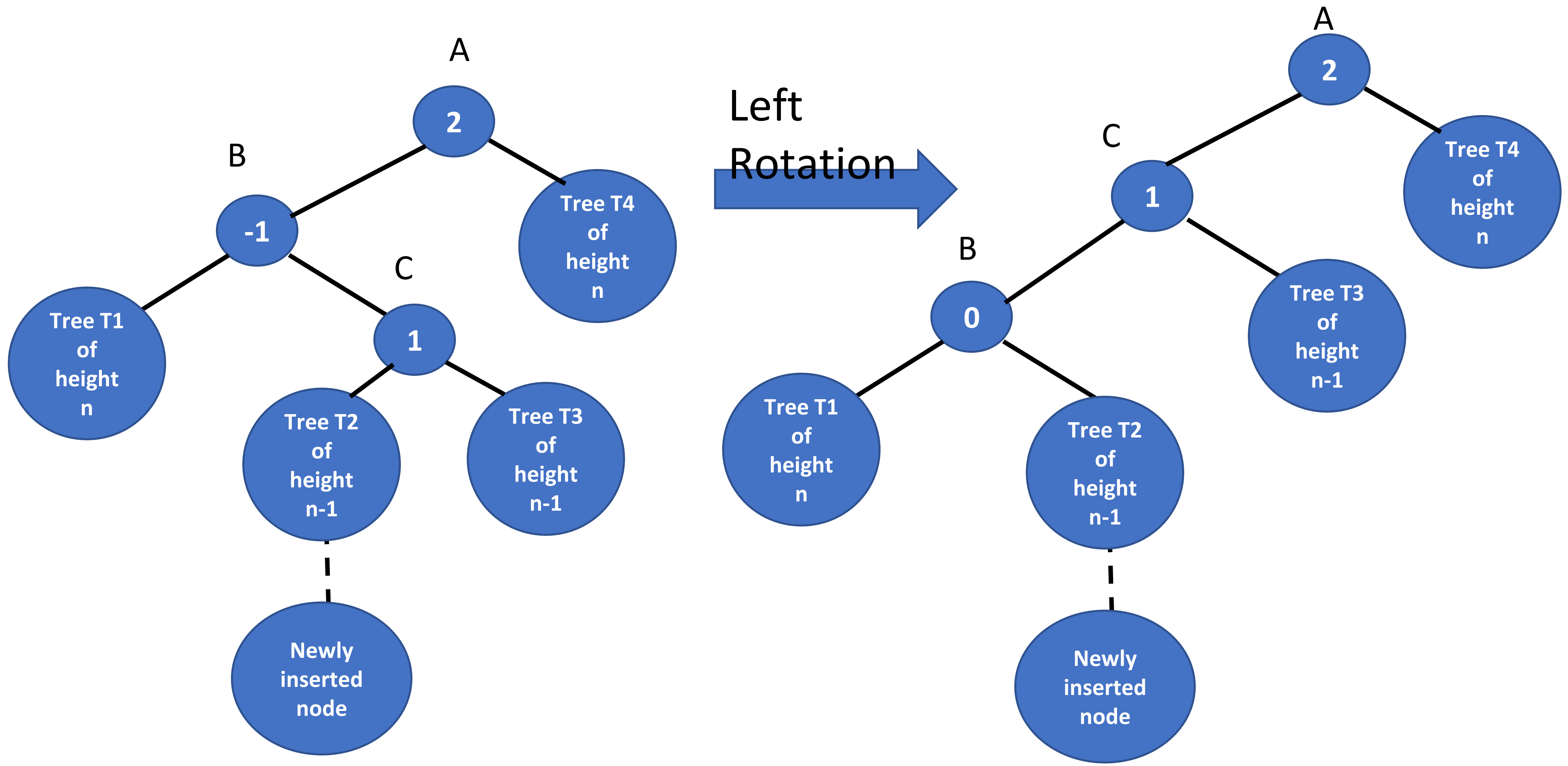
Unbalanced Tree after inserting a node to right subtree

- Newly inserted node is left descendent of the node A
- New node is inserted into right subtree of left child of A



Balance factor(C)= n-(n-1) = 1
Balance factor(B)= n-(n+1) = -1
Balance factor(A)= n+2-n= 2

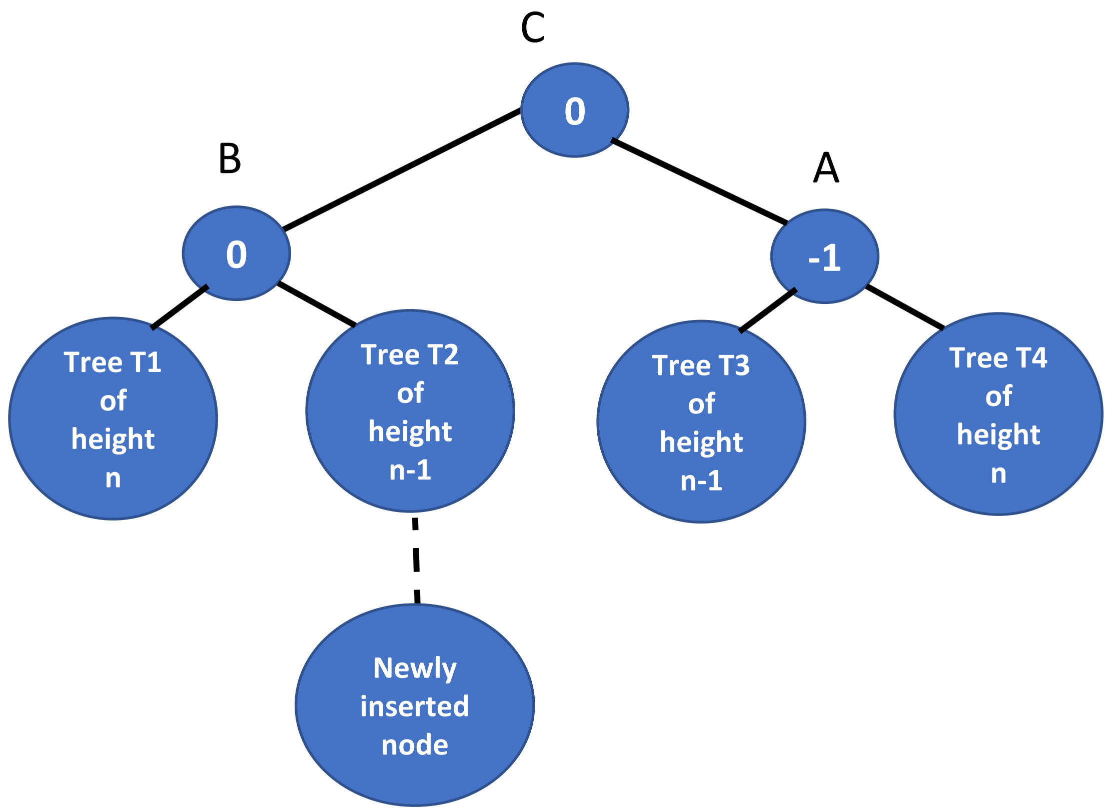
Transformed Balanced tree after Rotations



Courtesy: “Data Structures using c and c++ ” By Y Langsam, M. J. Augenstein and Aron M. Tenenbaum

Transformed Balanced tree after Rotations

Right Rotation



Courtesy: “Data Structures using c and c++ ” By Y Langsam, M. J. Augenstein and Aron M. Tenenbaum

AVL tree Insertion

- Insertion in AVL tree is performed using standard BST Insertion
- If tree becomes unbalanced, we rebalance the tree using left or right rotation
- If node X is inserted into balanced BST
- we need to find the youngest ancestor which becomes unbalanced

Four cases:

- IF(Balance factor of node) == 2 – unbalanced node(U)
  - ✓ case-1 : Left-Left case
    - IF((newly inserted key) < (key in the left subtree' root))
  - ✓ case-2: Left-Right case
    - IF((newly inserted key) > (key in the left subtree' root))

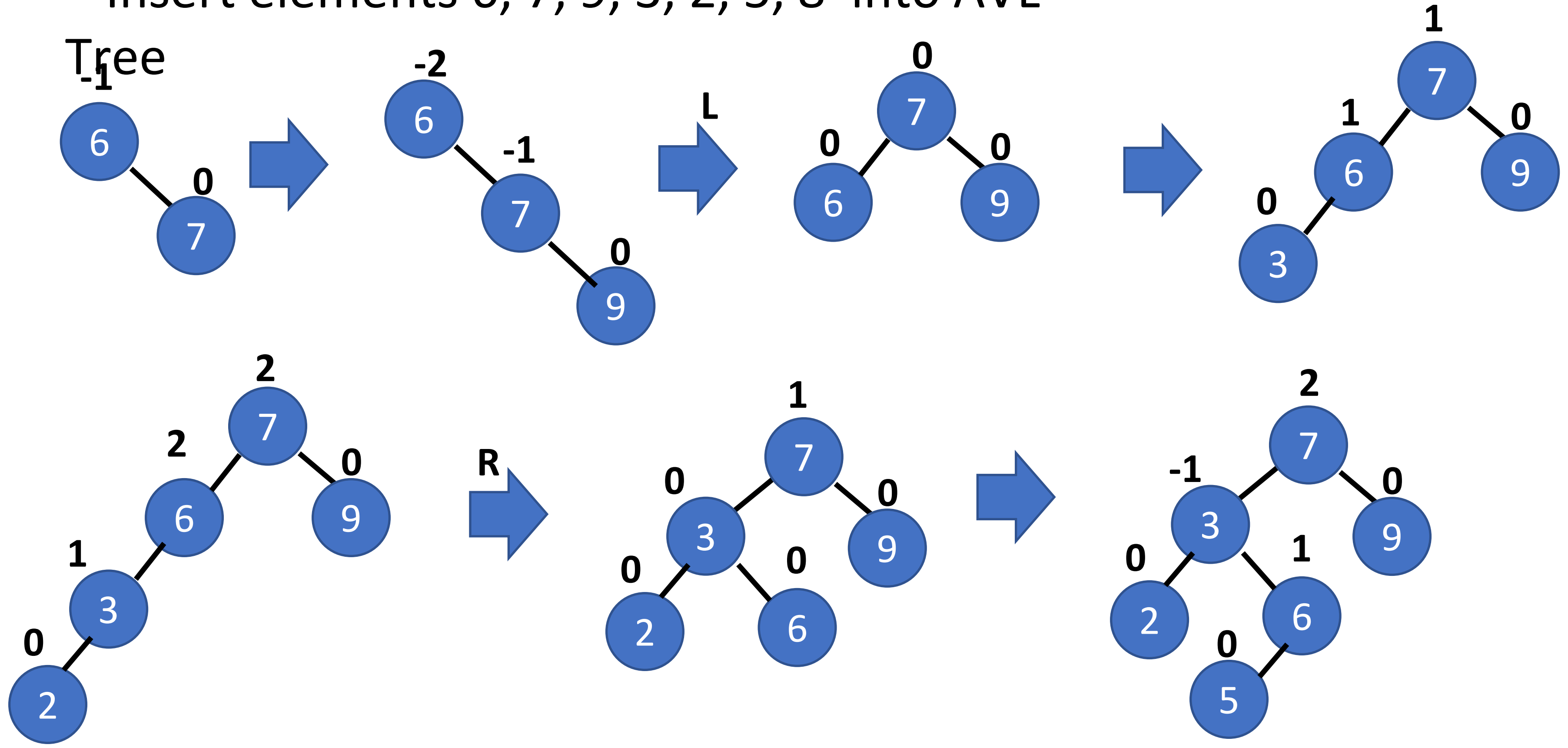
AVL tree Insertion

Four cases:

- IF((Balance factor of node)) == -2 – unbalanced node(U)
  - ✓ Case 3: Right-Right case
    - IF((newly inserted key) > (key in the right subtree' root))
  - ✓ Case 4:Right-Left case
    - IF((newly inserted key) < (key in the right subtree' root))

Examples – AVL Tree Insertions

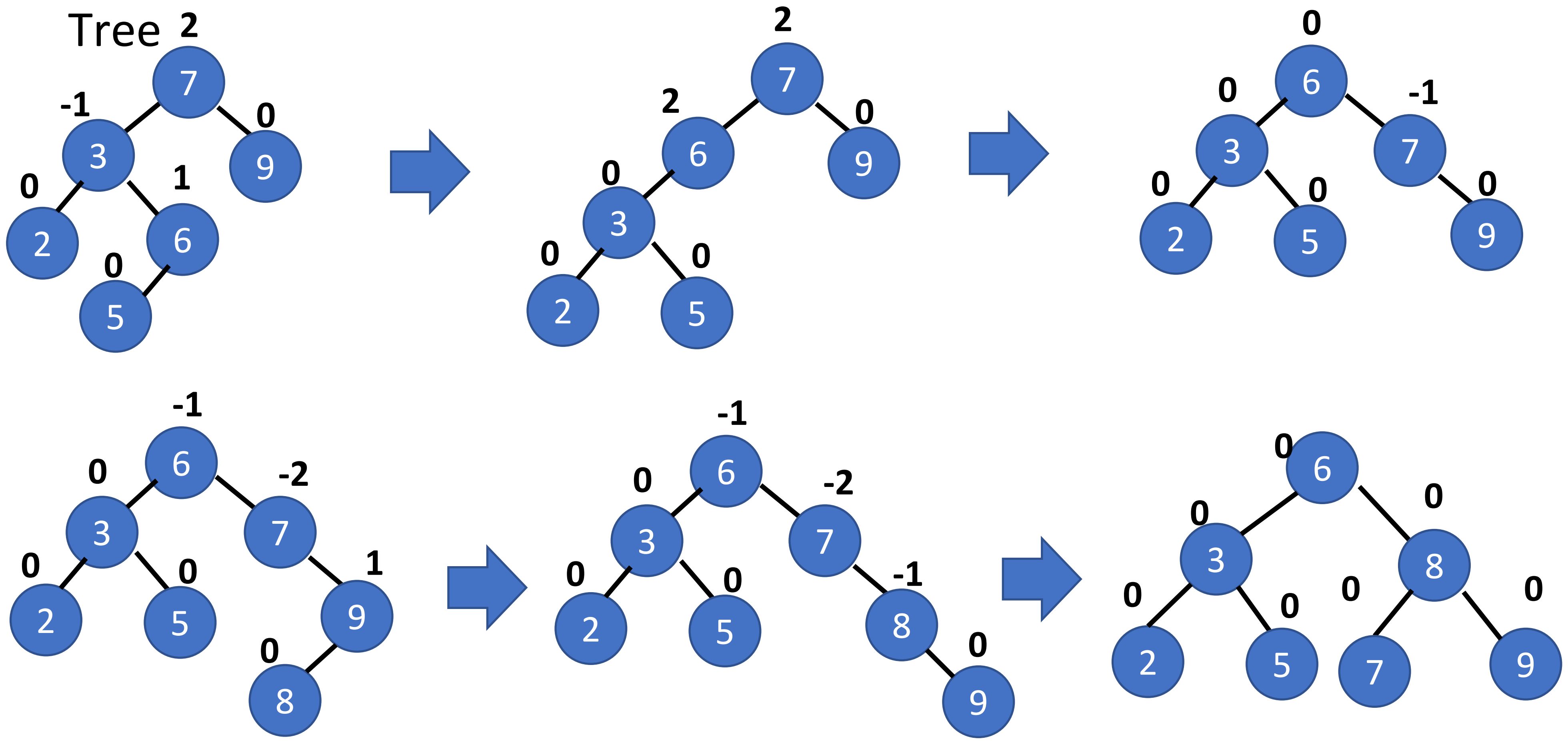
- Insert elements 6, 7, 9, 3, 2, 5, 8 into AVL Tree



DATA STRUCTURES AND ITS APPLICATIONS

Example – AVL Tree Insertions

- Insert elements 6, 7, 9, 3, 2, 5, 8 into AVL Tree



Deletions in AVL tree

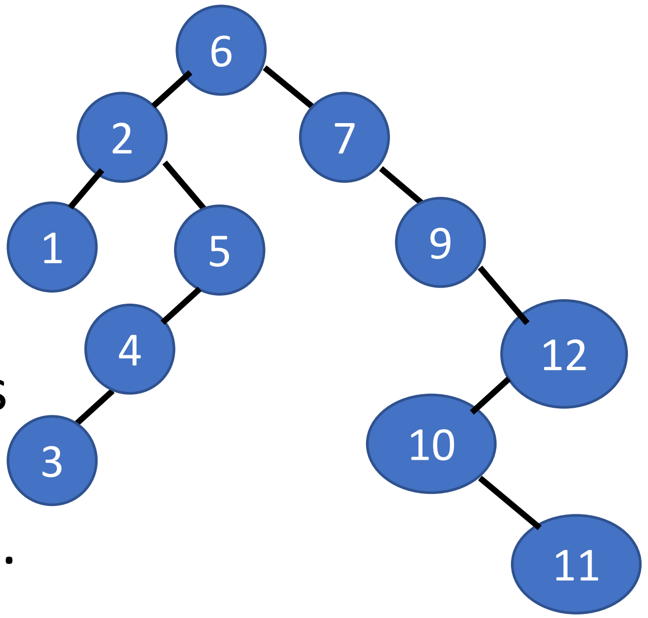
- Deletion in AVL tree is performed using standard BST Deletion
- If tree becomes unbalanced, we rebalance the tree using left or right rotation

BST Deletion: 3- case: Node to be delted

Case 1: Does not have any children

Case 2: has either left or right subtrees

Case 3: has both left and right subtrees.

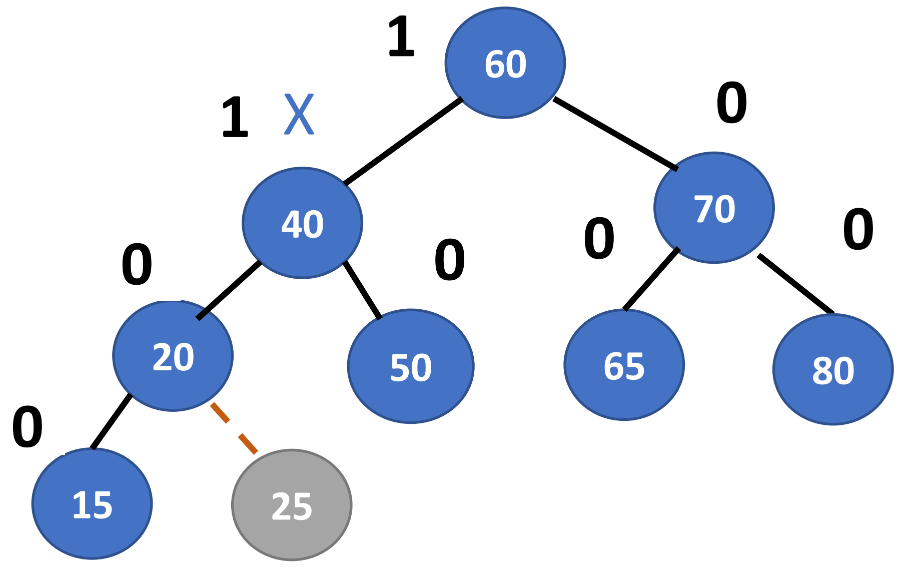


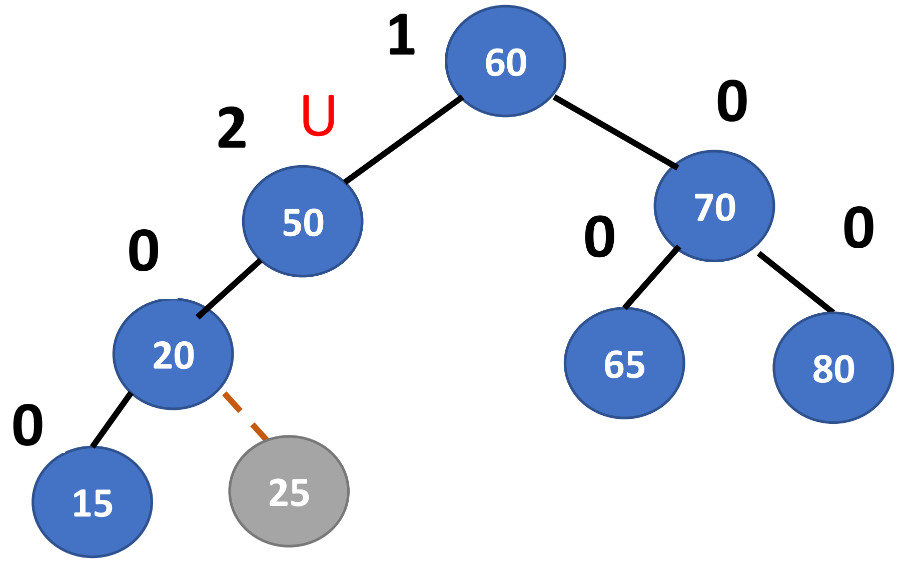
Deletions in AVL tree

- If node X is deleted from the BST
- we need to find the youngest ancestor which becomes unbalanced

Four cases: Case-1

- IF((Balance factor of a node) == 2) - unbalanced node(U)
  - ✓ Left-Left case:
    - IF(Balance factor of left subtree's root) >= 0

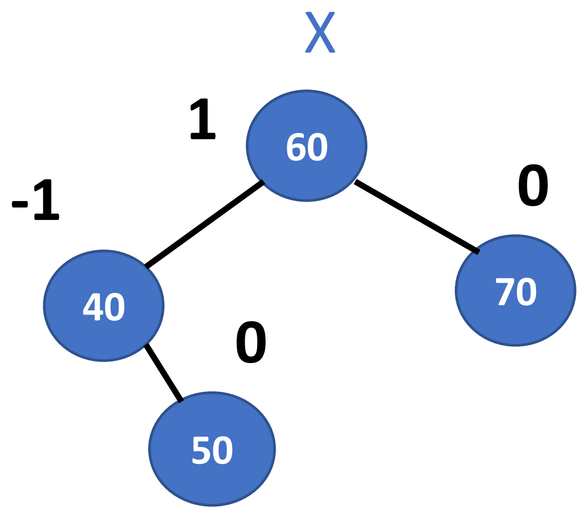


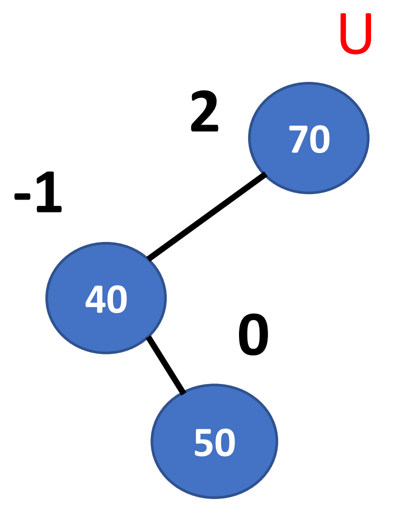


Deletions in AVL tree

Case-2:

- IF(Balance factor of a node == 2) – unbalanced node(U)
  - ✓Left-Right case
    - IF(Balance factor of left subtree's root) < 0

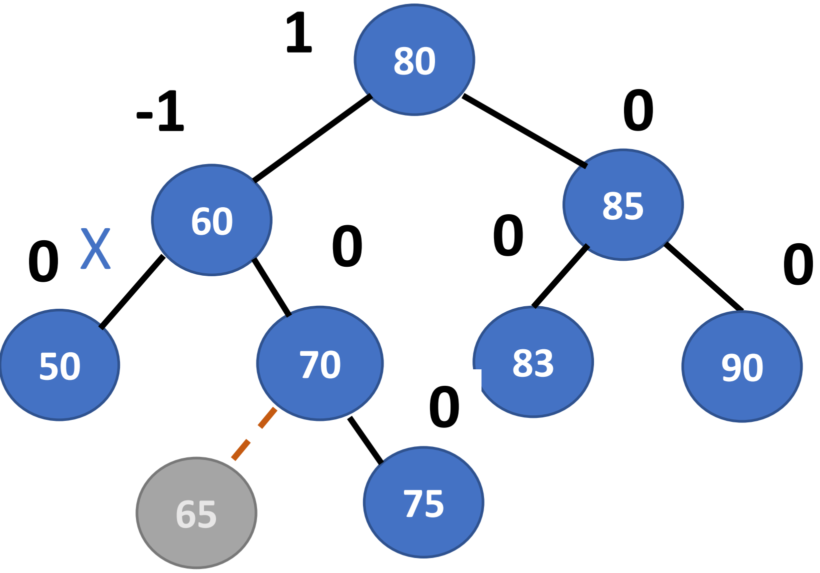


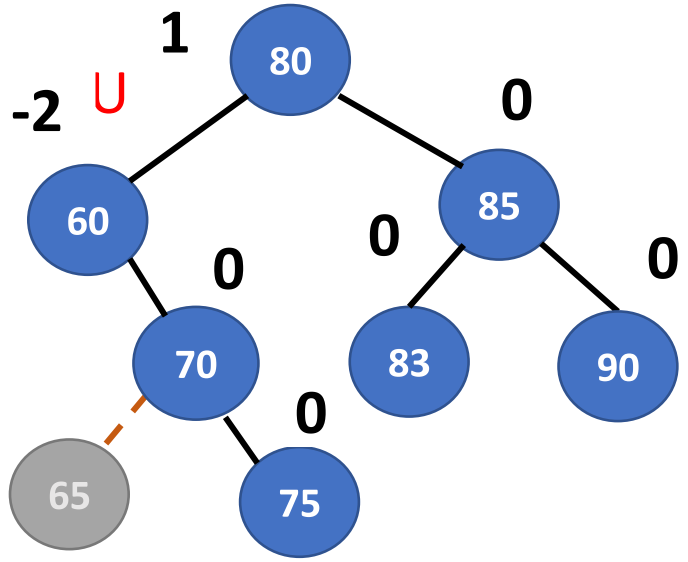


Deletions in AVL tree

Case-3:

- If ((Balance factor of a node) == -2) unbalanced node(U)
  - ✓Right-Right case
    - IF(Balance factor of Right subtree's root) <= 0





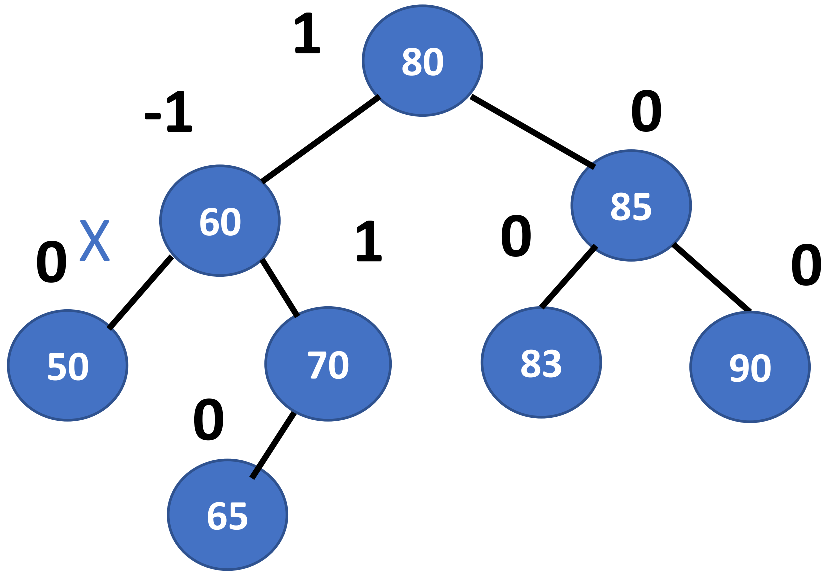
Deletions in AVL tree

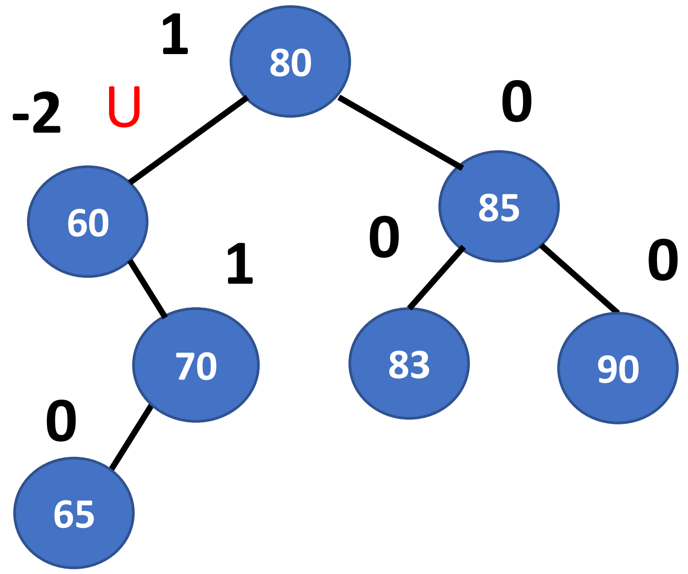
Case-4:

IF((Balance factor of unbalanced node) == -2) unbalanced node(U)

✓Right-Left case

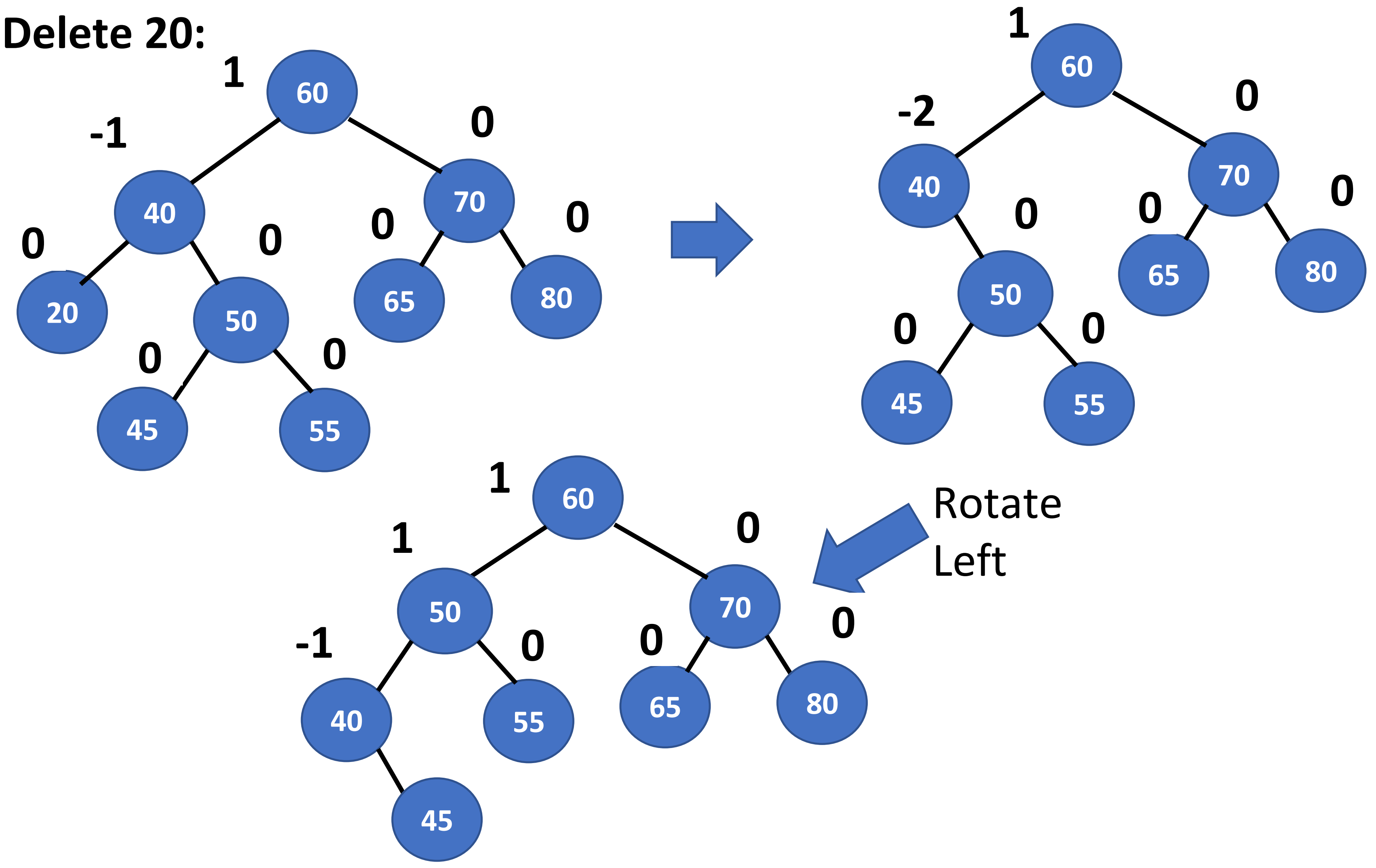
- IF(Balance factor of Right sub tree's root) > 0





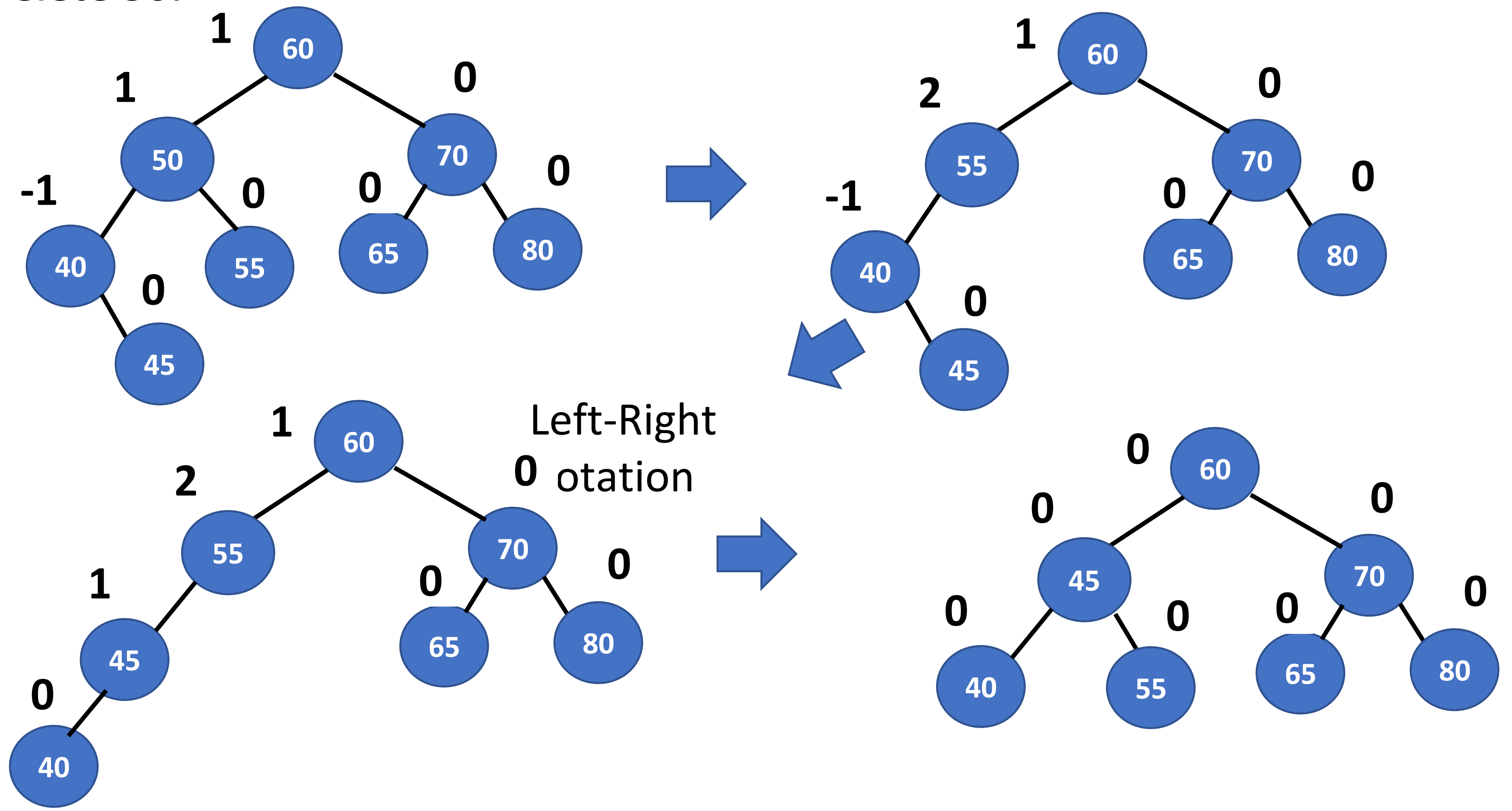
Example – Deletions in AVL tree

Delete 20:



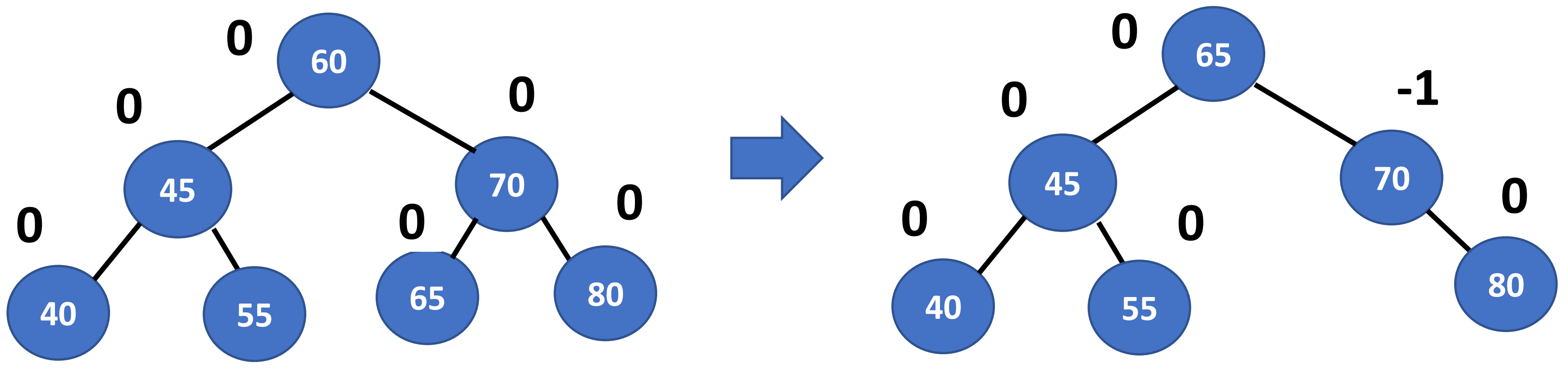
Example – Deletions in AVL tree

Delete 50:

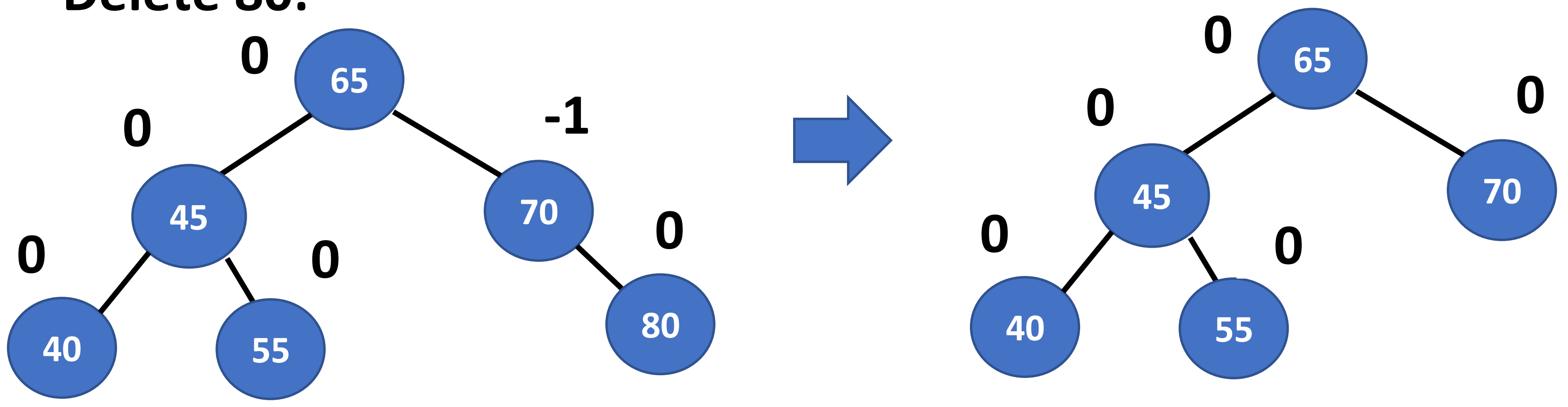


Example – Deletions in AVL tree

Delete 60:

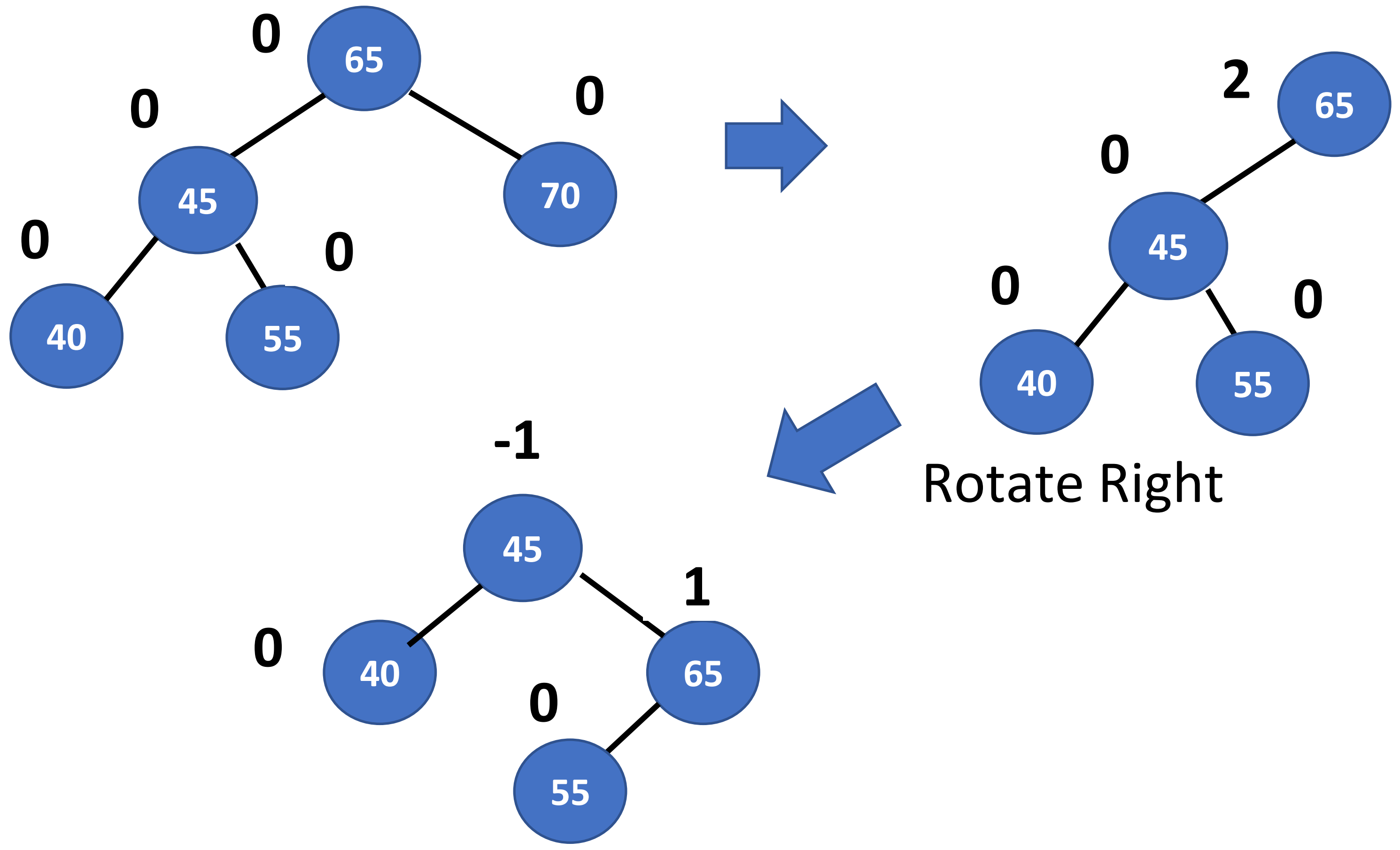


Delete 80:



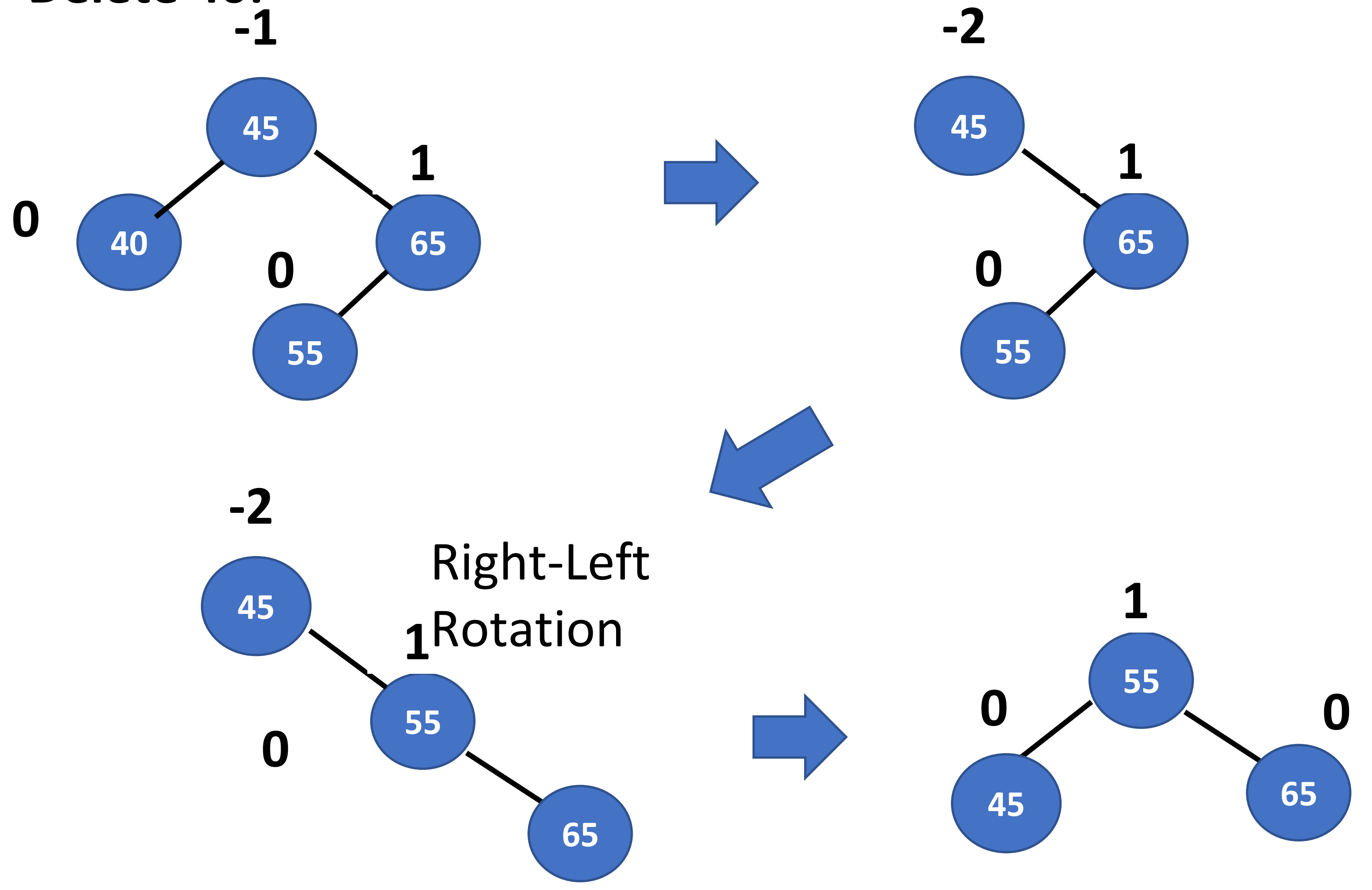
Example – Deletions in AVL tree

Delete 70:



Example – Deletions in AVL tree

Delete 40:



1. During insertion in an AVL tree, imbalance occurs when the balance factor becomes:

A) −1 or +1
B) −2 or +2
C) 0
D) None of the above

PES UNIVERSITY

1. During insertion in an AVL tree, imbalance occurs when the balance factor becomes:

A) −1 or +1
B) −2 or +2
C) 0
D) None of the above

(Soln: In an AVL tree, imbalance occurs when the balance factor of any node becomes less than −1 or greater than +1, i.e., when it reaches −2 or +2.)

2. Insertion into the left subtree of the left child of a node causes which imbalance?
A) LL case
B) RR case
C) LR case
D) RL case

2. Insertion into the left subtree of the left child of a node causes which imbalance?

A) LL case
B) RR case
C) LR case
D) RL case

3. After inserting nodes 10, 20, 30 in sequence into an empty AVL tree, the root becomes:

A) 10
B) 20
C) 30
D) 15

3. After inserting nodes 10, 20, 30 in sequence into an empty AVL tree, the root becomes:

A) 10
B) 20
C) 30
D) 15

4. After deleting a node from an AVL tree, which of the following is TRUE?

A) Balance factor may become −2 or +2 before rebalancing

B) Rebalancing is never required

C) Deletion requires more rotations than insertion in worst case

D) Both A and C

4. After deleting a node from an AVL tree, which of the following is TRUE?

A) Balance factor may become −2 or +2 before rebalancing

B) Rebalancing is never required

C) Deletion requires more rotations than insertion in worst case

D) Both A and C

THANK YOU

Sandesh B. J
Department of Computer Science & Engineering

Sandesh_bj@pes.edu