DATA STRUCTURES AND ITS APPLICATIONS

Balanced Trees

Sandesh B. J

Department of Computer Science & Engineering

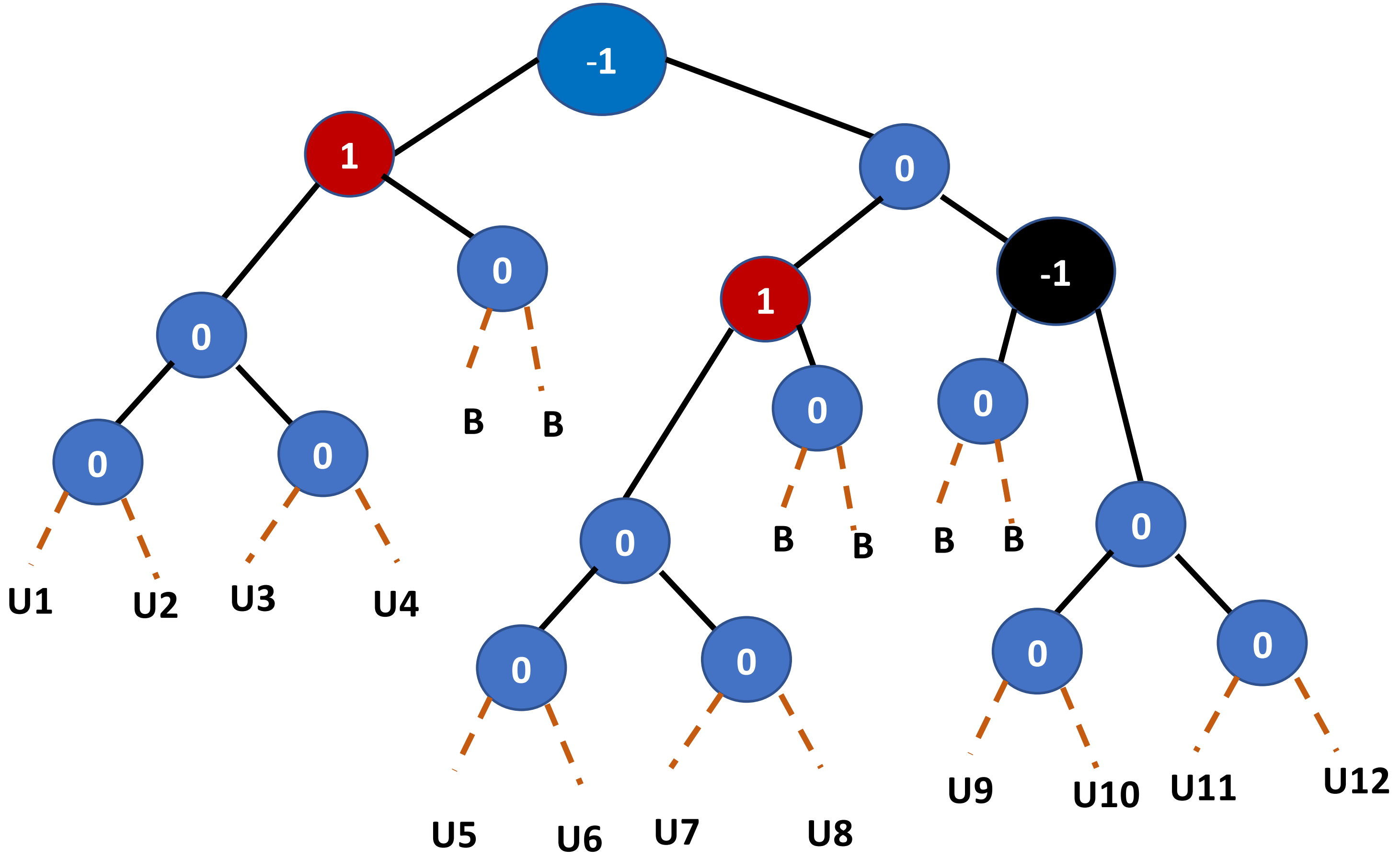
DATA STRUCTURES AND ITS APPLICATIONS

Balanced Trees

Sandesh B. J

Department of Computer Science & Engineering

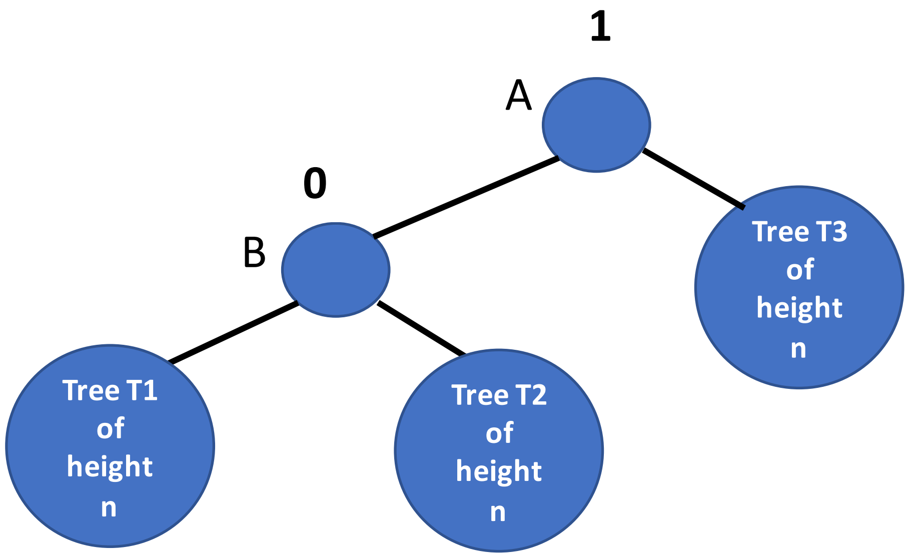
Possible insertion into AVL tree



- 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

Unbalanced Tree after inserting a node to left subtree



Balance factor(A) = (n+1) – n =1

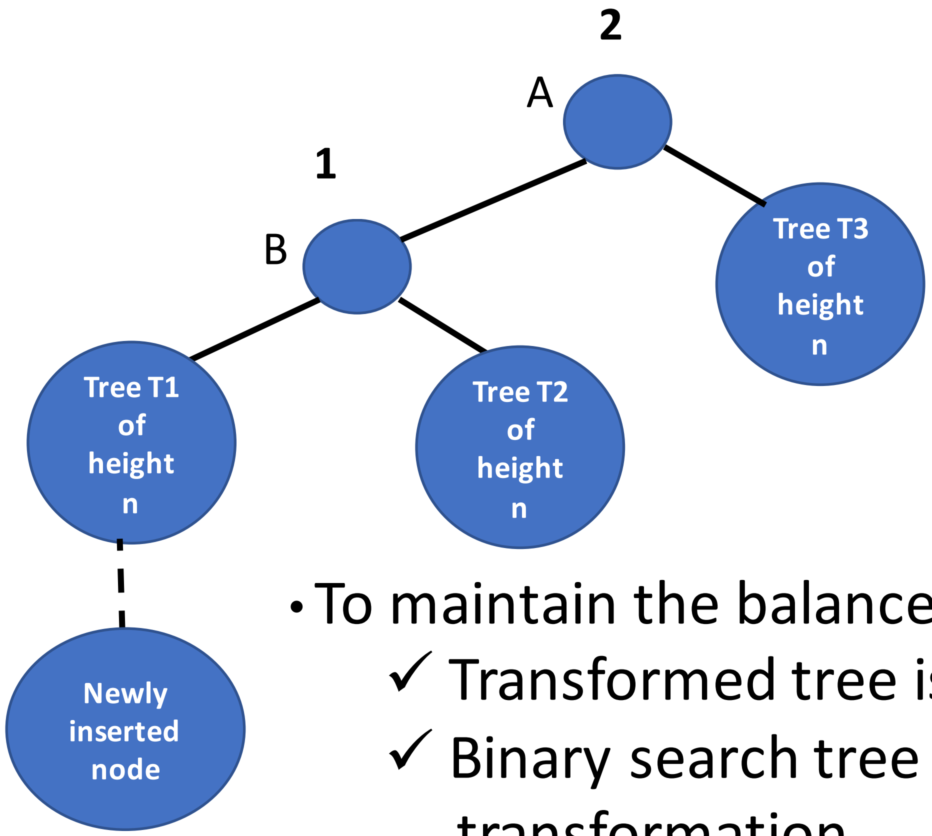
Balance factor(B) = n – n = 0

- Let us consider A is the youngest ancestor which becomes unbalanced
- Balance factor of A should 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 created node in inserted into left subtree of node B
- Changing the balance **B to 1 and A to 2**
- A is the youngest ancestor of the new node to become unbalanced

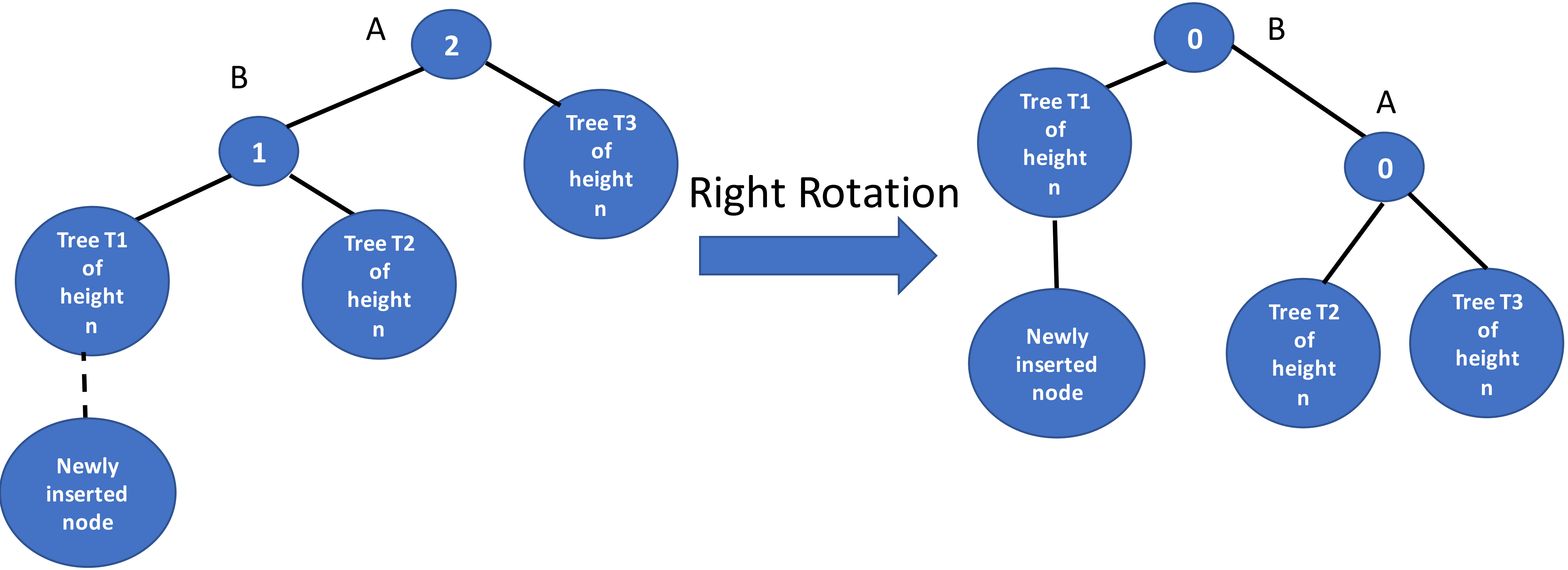


- To maintain the balance : Tree needs to be transformed
  - ✓ Transformed tree is balanced
  - ✓ Binary search tree property is maintained after transformation

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

Transformed Balanced Tree after Rotations

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

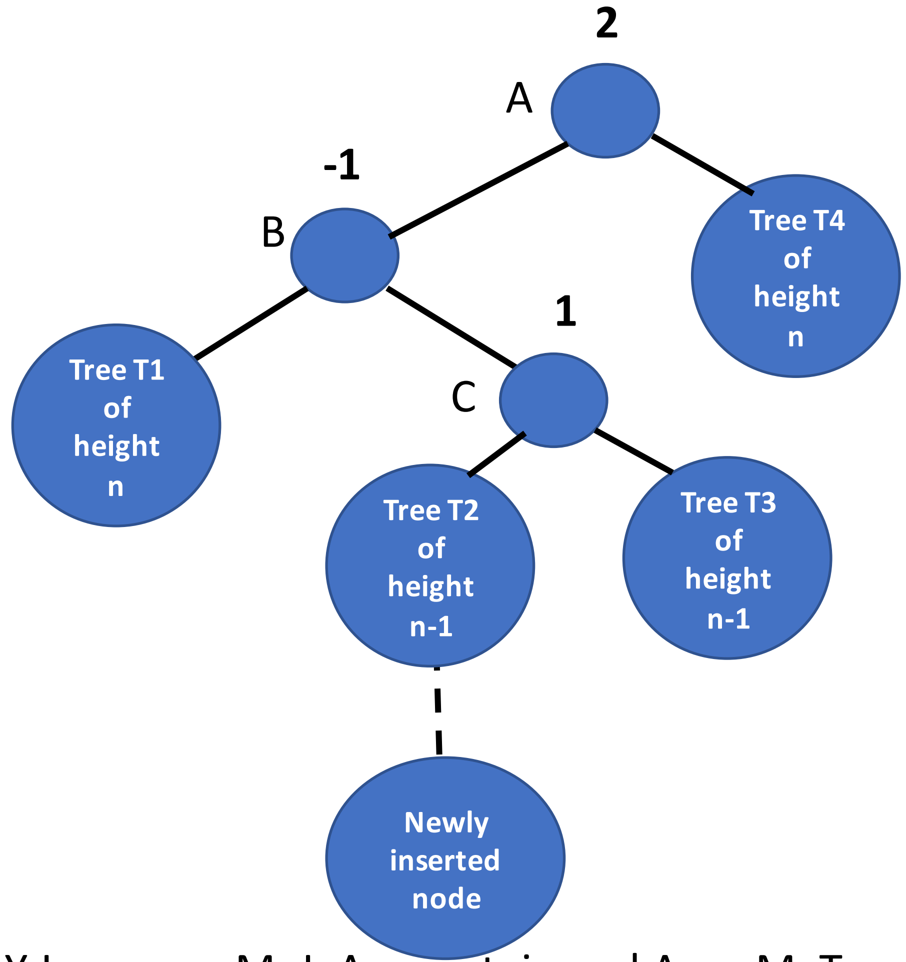


Courtesy: “Data Structures using c and c++ ” By Y Langsam, M. J. Augenstein and Aron M. Tenenbaum

Unbalanced Tree after inserting a node to right subtree

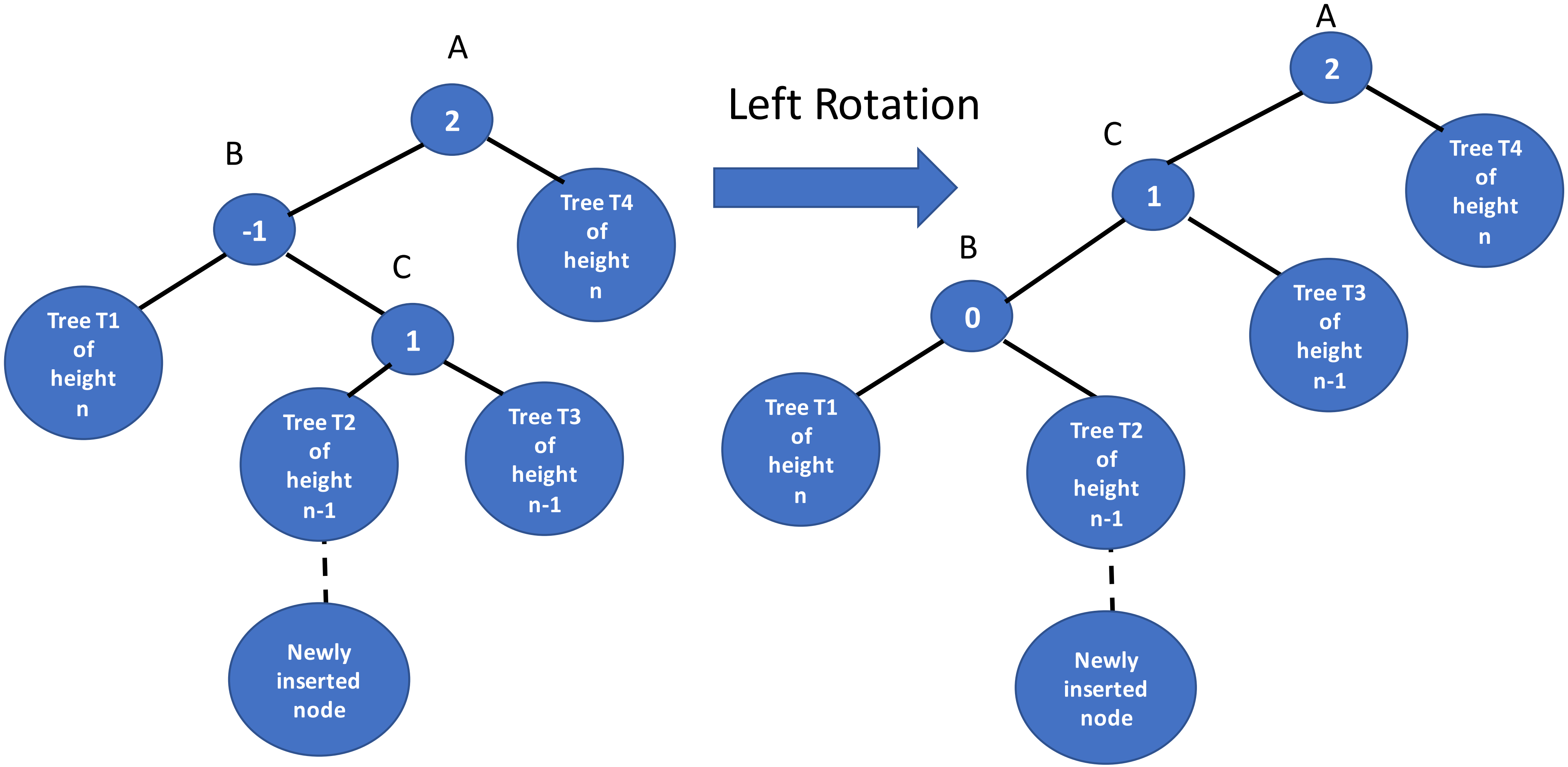
- Newly created node is inserted into right subtree changing the balance:

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



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

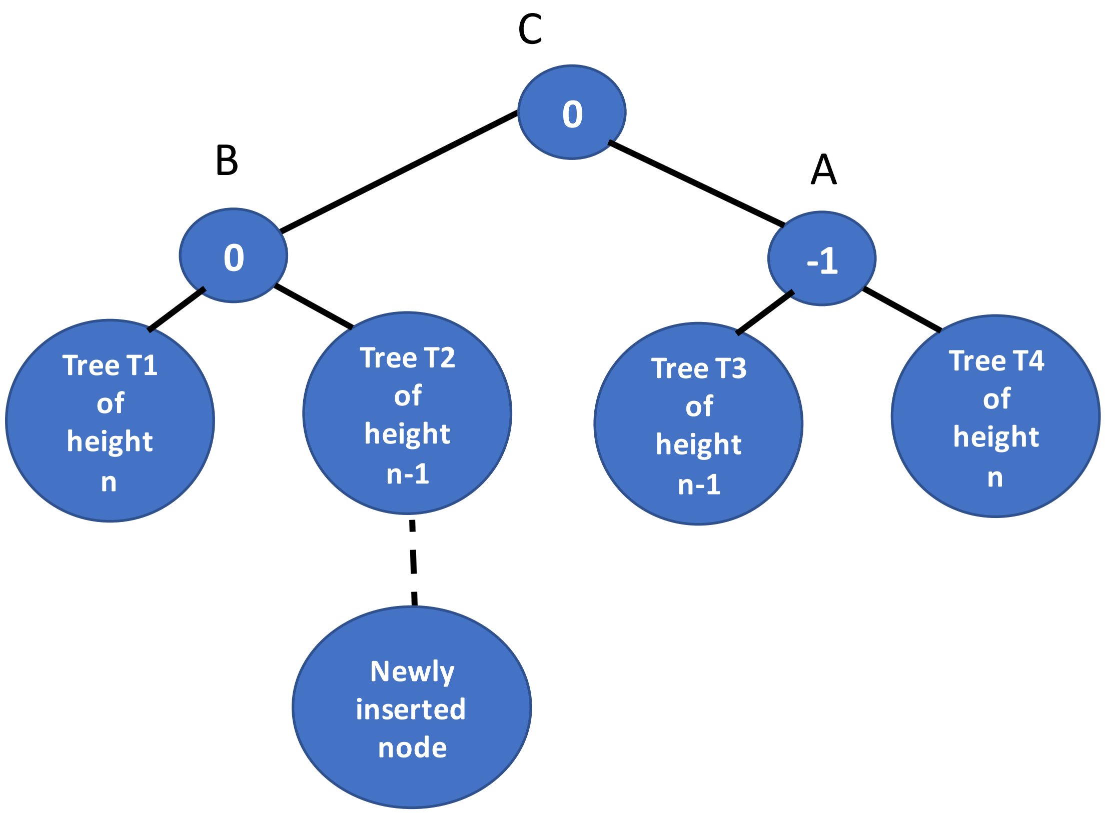
Transformed Balanced tree after Rotations



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 BST
- we need to find the youngest ancestor which becomes unbalanced

Four cases:

- Balance factor of node is greater than 1 – unbalanced node(U)
  - ✓ Left-Left case
    - if newly inserted key is less than the key in the left subtree' root
  - ✓ Left-Right case
    - if newly inserted key is greater than the key in the left subtree's root

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

AVL tree Insertion

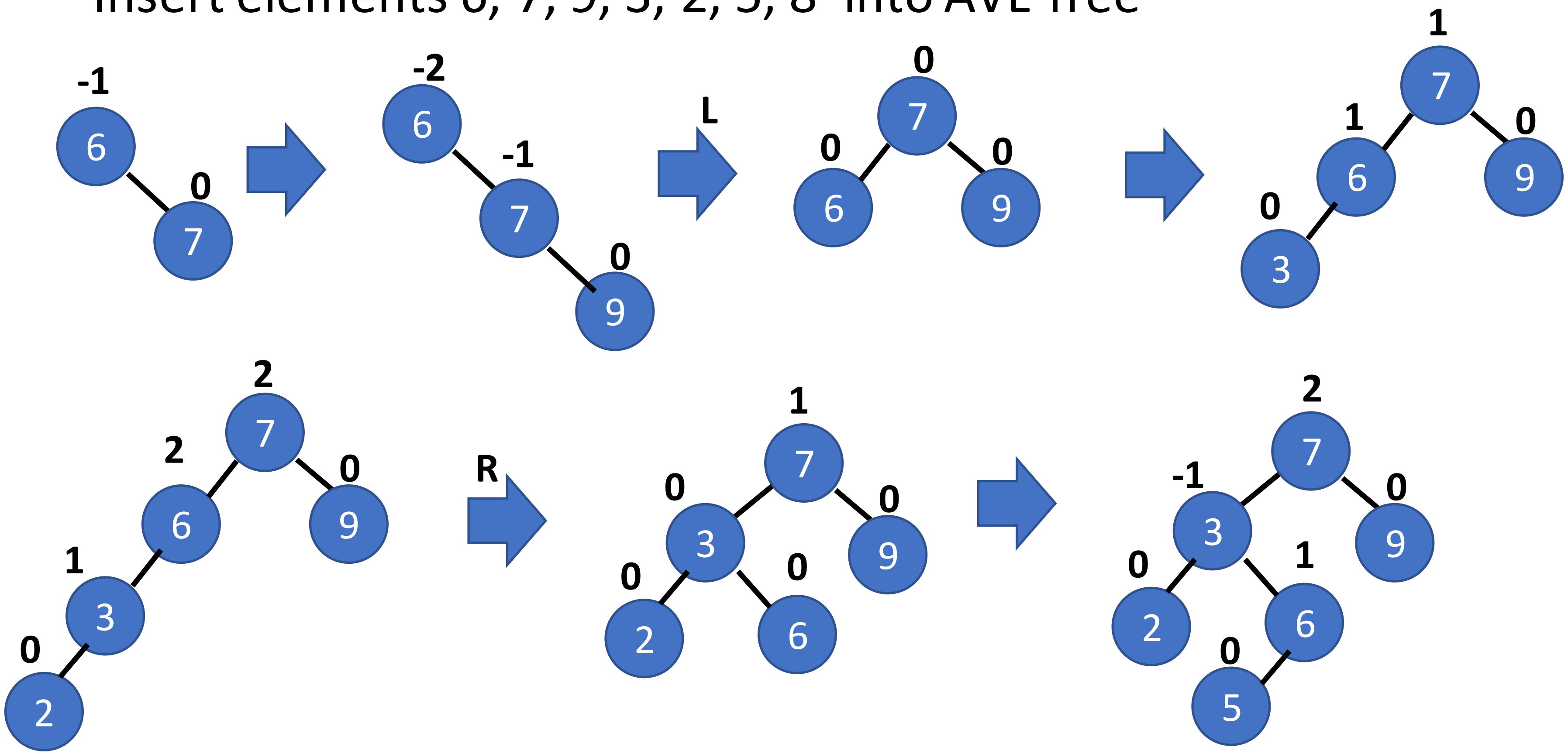
Four cases:

- Balance factor of node is lesser than -1 – unbalanced node(U)
  - ✓ Right-Right case
    - ▪ if newly inserted key is greater than the key in the right subtree' root.
  - ✓ Right-Left case
    - ▪ if newly inserted key is lesser than the key in the right subtree's root.

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

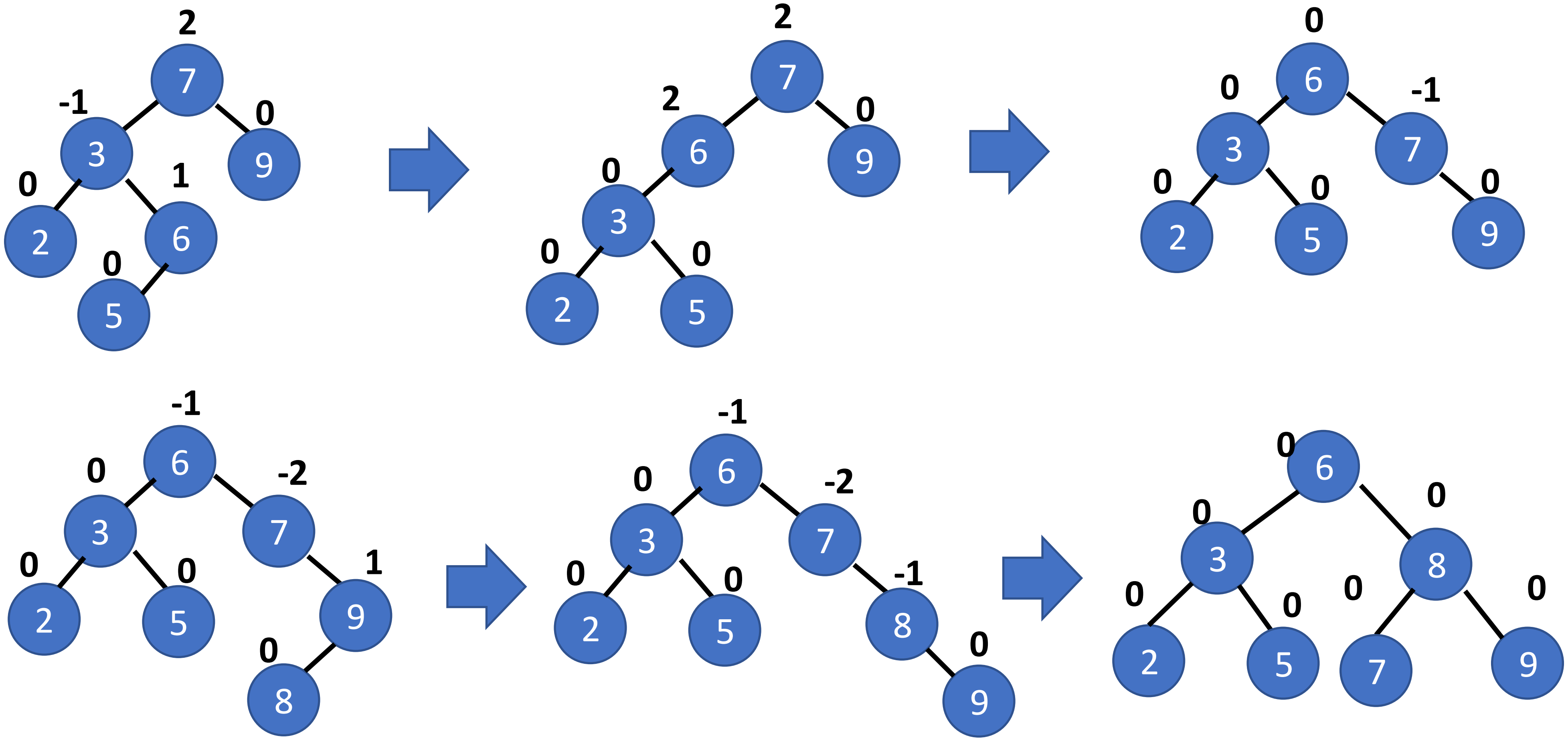
Examples – AVL Tree Insertions

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



Example – AVL Tree Insertions

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



Deletions in AVL tree

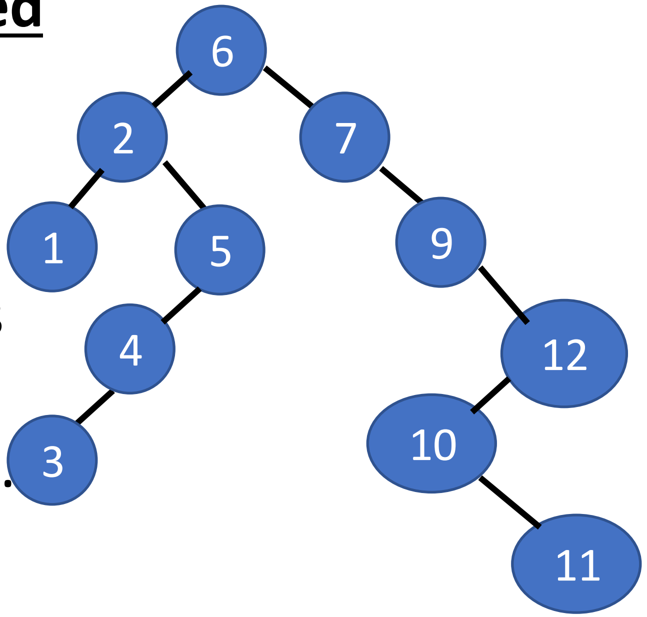
- Deletion in AVL tree is performed using standard BST Deletion
- If tree becomes unbalanced, we rebalance the tree using left or right rotation

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

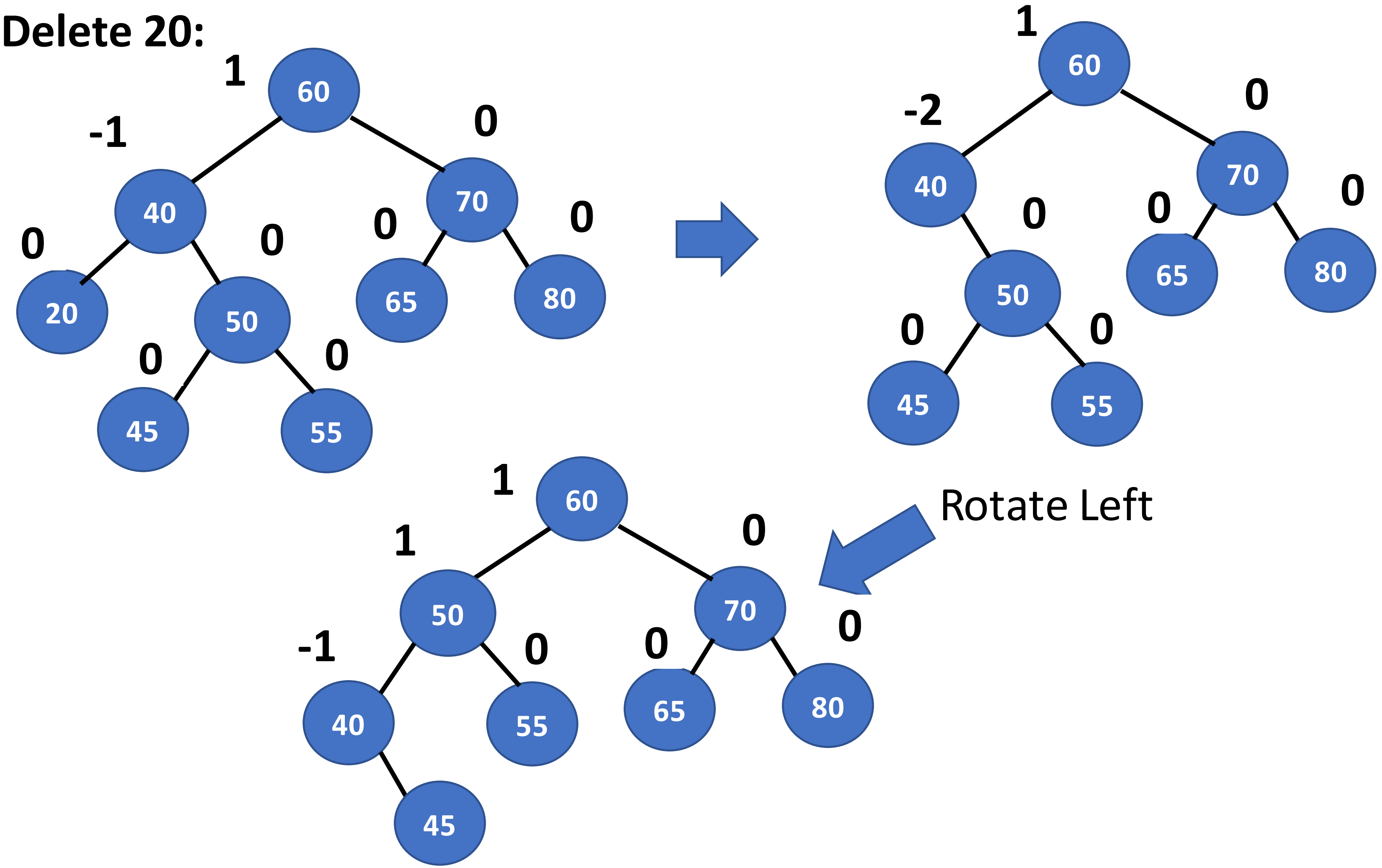
- If node X is deleted from the BST
- we need to find the youngest ancestor which becomes unbalanced

Four cases:

- Balance factor of a node is greater than 1 – unbalanced node(U)
    - ✓ Left-Left case
        - ▪ Balance factor of left subtree's root is greater than or equal to 0
    - ✓ Left-Right case
        - ▪ Balance factor of left subtree's root is less than 0
- Balance factor of unbalanced node is less than -1
    - ✓ Right-Right case
        - ▪ Balance factor of Right subtree's root is less than or equal to 0
    - ✓ Right-Left case
        - ▪ Balance factor of sub tree is grater than 0

Example – Deletions in AVL tree

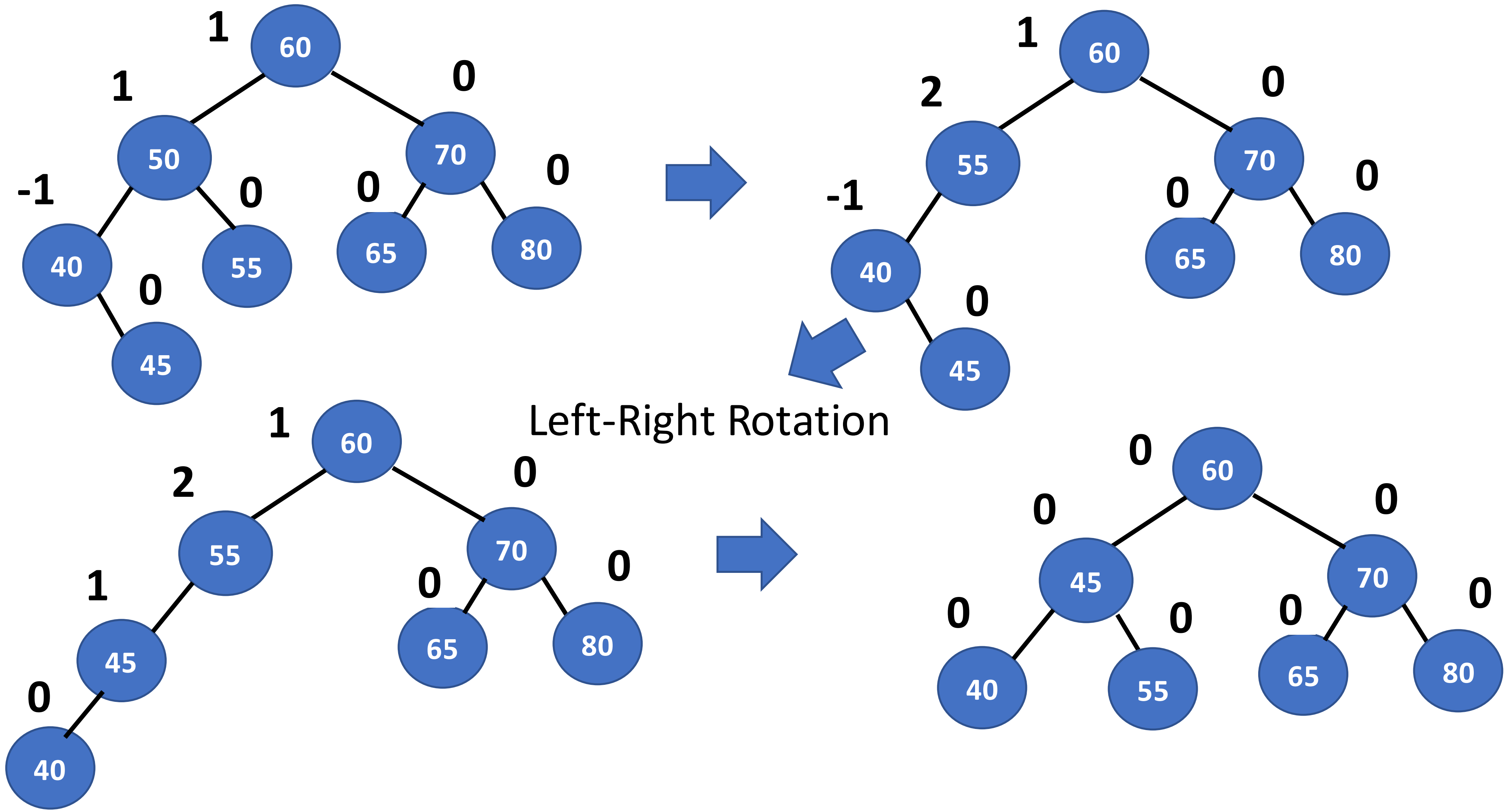
Delete 20:



Rotate Left

Example – Deletions in AVL tree

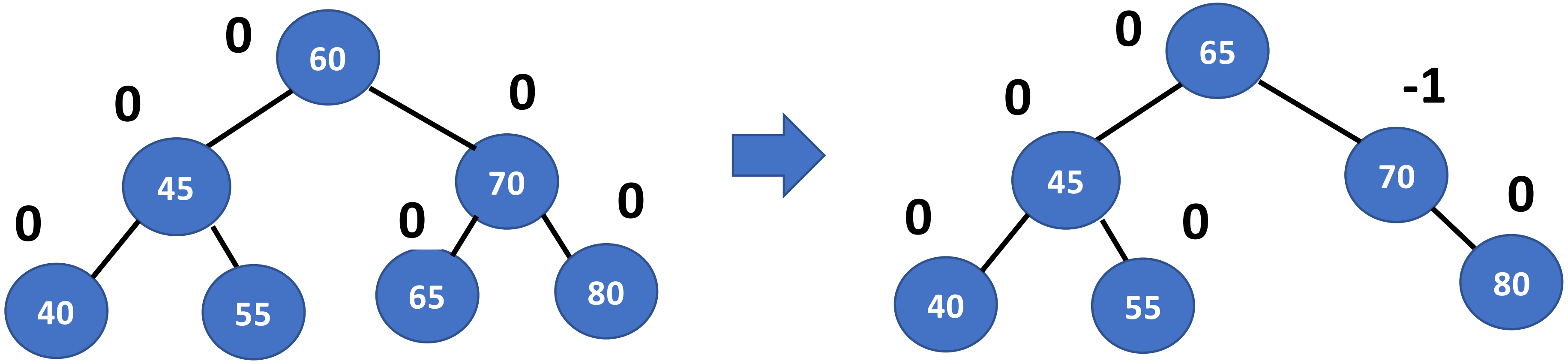
Delete 50:



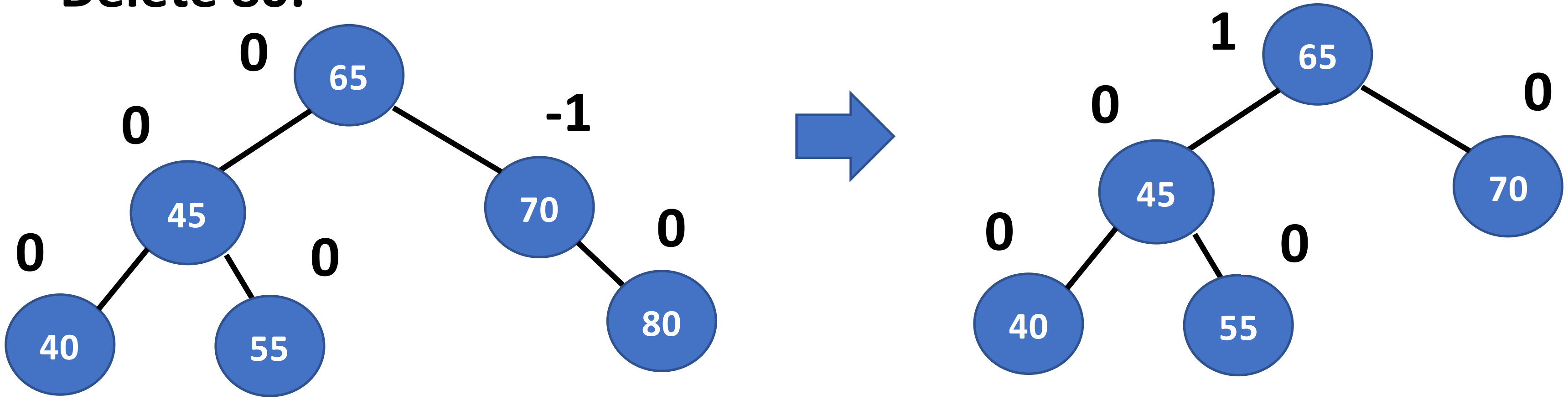
Left-Right Rotation

Example – Deletions in AVL tree

Delete 60:

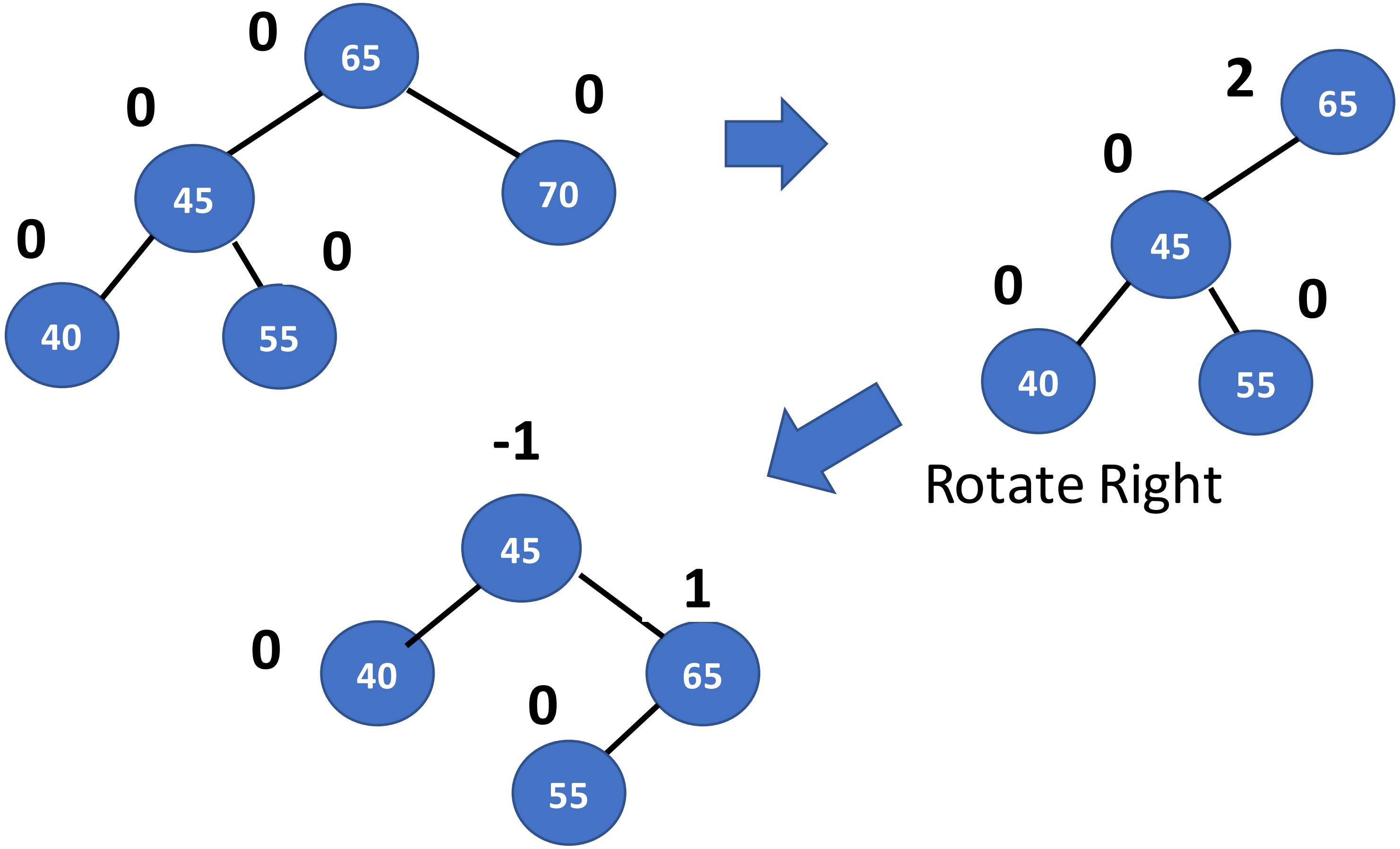


Delete 80:



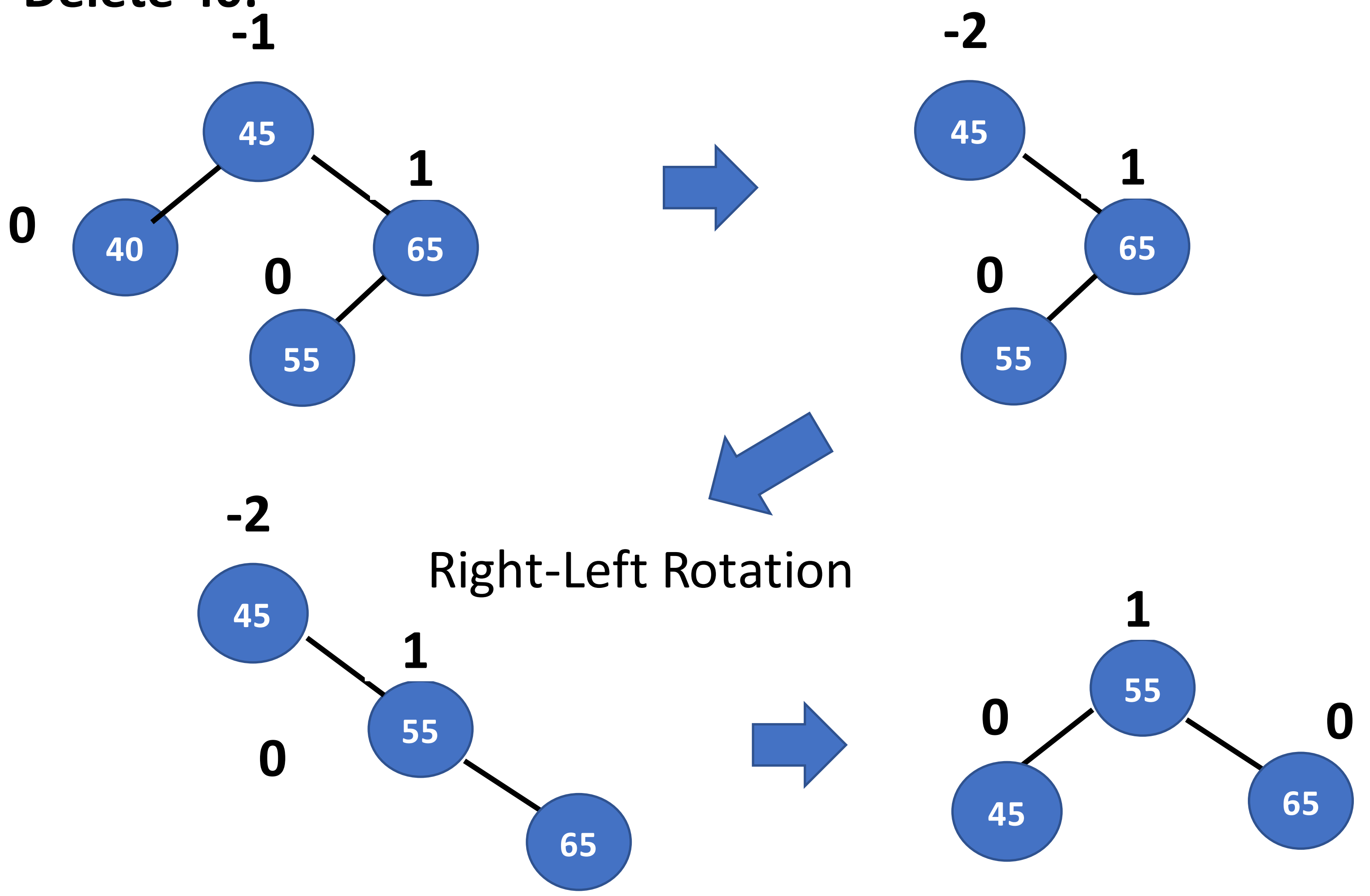
Example – Deletions in AVL tree

Delete 70:



Example – Deletions in AVL tree

Delete 40:



Right-Left Rotation

PES
UNIVERSITY
ONLINE

THANK YOU

Sandesh B. J
Department of Computer Science & Engineering

Sandesh_bj@pes.edu

+91 80 6618 6649