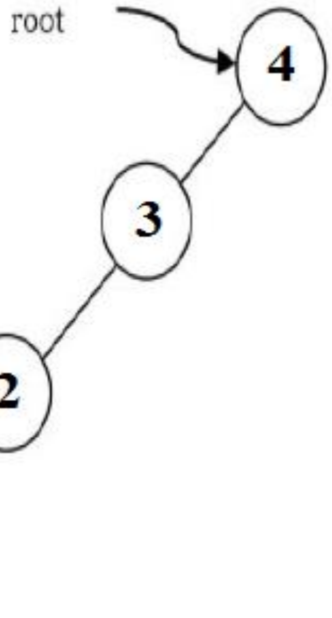
CS-2001 Data Structures

AVL Tree

Problem

- When a **Binary Search Tree (BST)** is *skewed*, it's worst-time complexity becomes the same as that of Linear Search **i.e. O(n)**

- Searching ***1*** in the given tree takes n comparisons



Solution

- In case of skewed Binary Search Trees, we might as well be using Linear Search

- This bad worst case behavior can be avoided by using an idea called *height balancing*

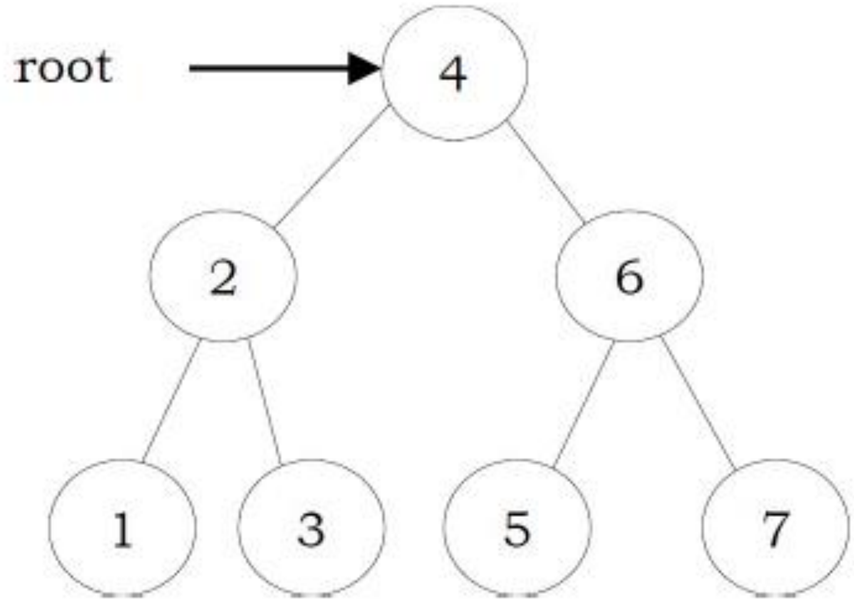
Balanced Trees

- The height balanced trees are represented with **HB(k)**, where k is the difference between *left subtree* height and *right subtree* height.

- Sometimes ***k*** is called *balance factor*

Full Balanced BST

- In HB(k), if **k = 0** (if balance factor is zero), then we call such a binary search tree as ***Full Balanced Binary Search Tree***



Self-balancing Binary Tree

- A **self-balancing BST** or height-balanced BST is a binary search tree that attempts to keep its height as small as possible at all times (after every node insertion)

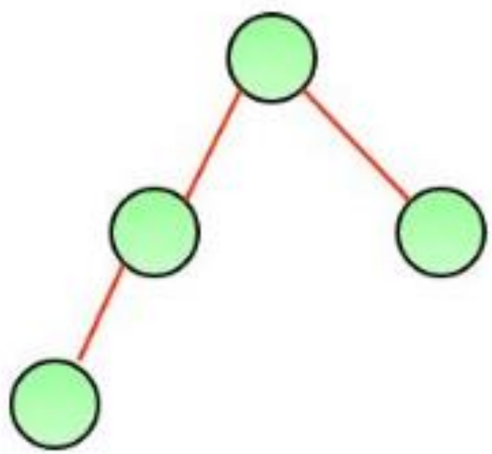
- **Example:**
  - ***AVL Tree***
  - *Red Black Tree*
  - *B+ Trees (balanced but not binary trees)*

AVL Trees

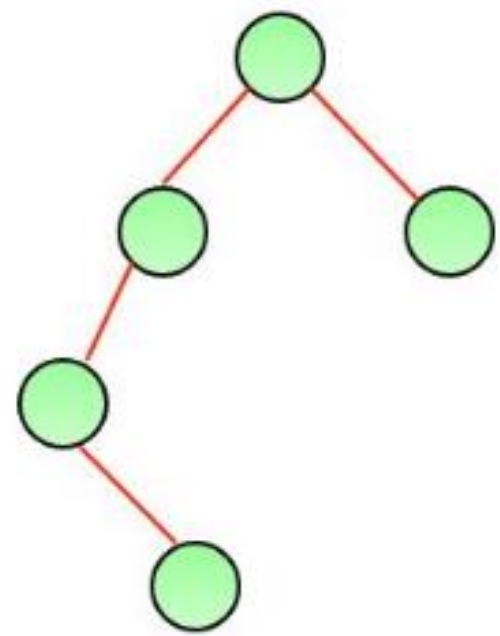
- An **AVL** *(Adelson-Velskii and Landis)* **Tree** is one with the following properties:

  1) *It is a Binary Search Tree*
  2) *For any given node X, the height of left subtree of X and right subtree of X differs by* ***at most 1***

Example



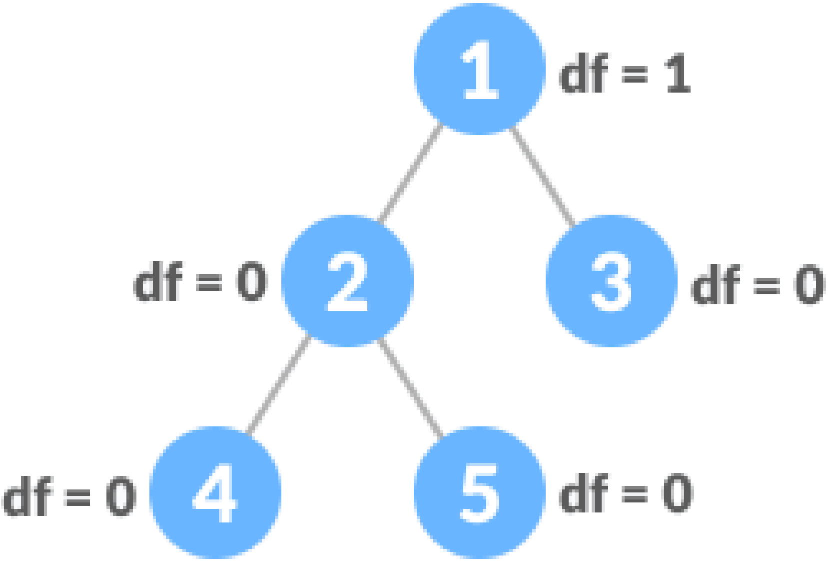
A height balanced tree



Not a height balanced tree

Example

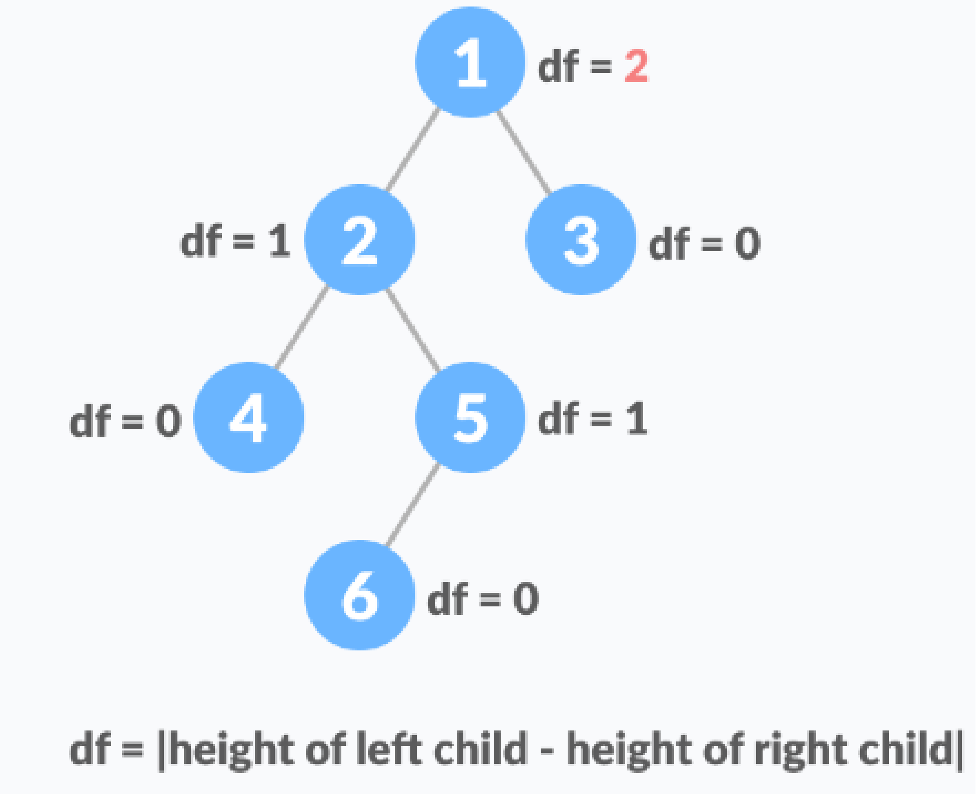
- If HB(Node 1) :



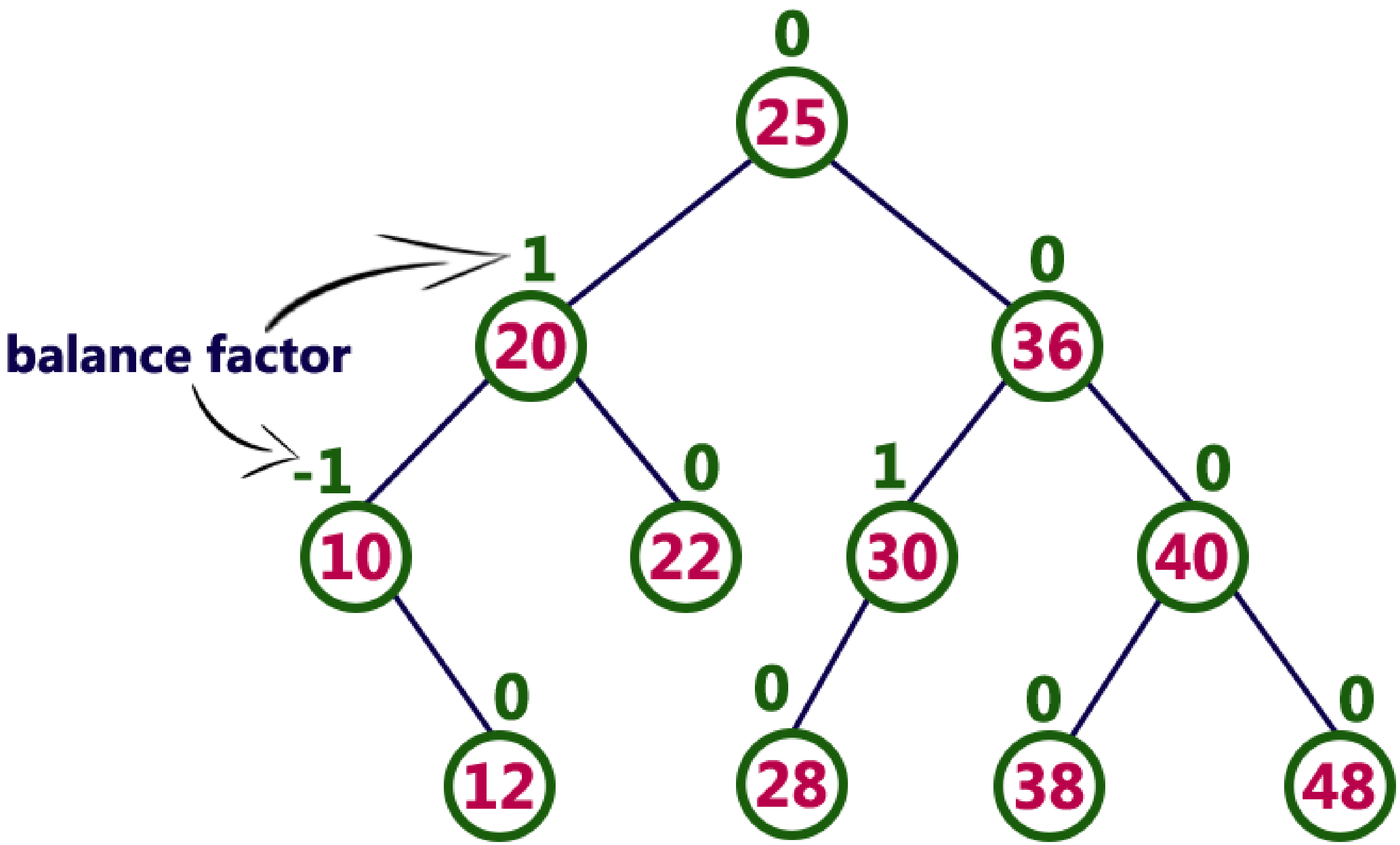
- There are no nodes with a difference in height of left and right subtrees of more than 1

Example

- If HB(Node 1) :



Example



Rotations

- When the tree structure changes (e.g., with insertion or deletion), we need to modify the tree to restore the AVL tree property. This can be done using ***single rotations*** or ***double rotations***

- If we balance the AVL tree every time, then at any point for a given node X, the difference in heights of left(*X) and right(X) differ* ***by exactly 2***

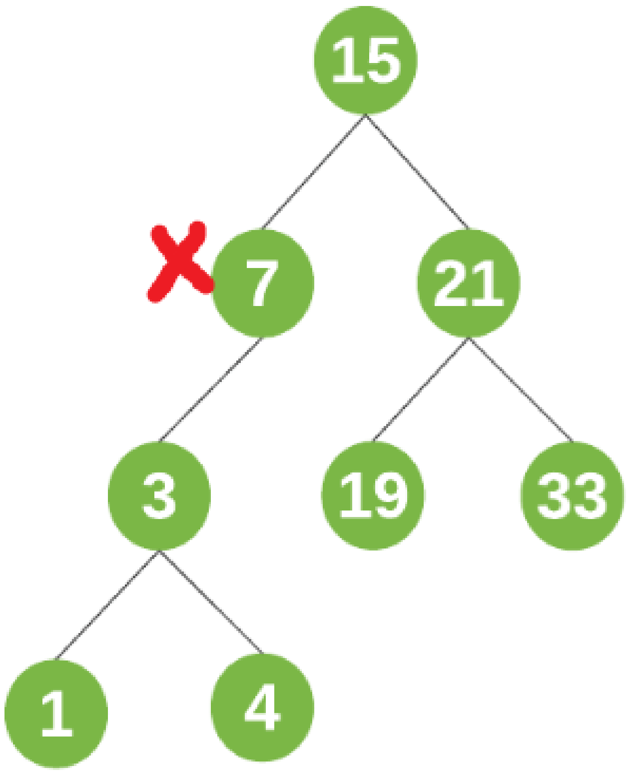
Important Observation

- Only nodes that are on the path from the insertion point to the root might have their balances altered

- To restore the AVL tree property, we start at the insertion point and keep going to the root of the tree.

- While moving to the root, we need to consider the first node that is not satisfying the AVL property. From that node onwards, every node on the path to the root will have the issue.

Observation



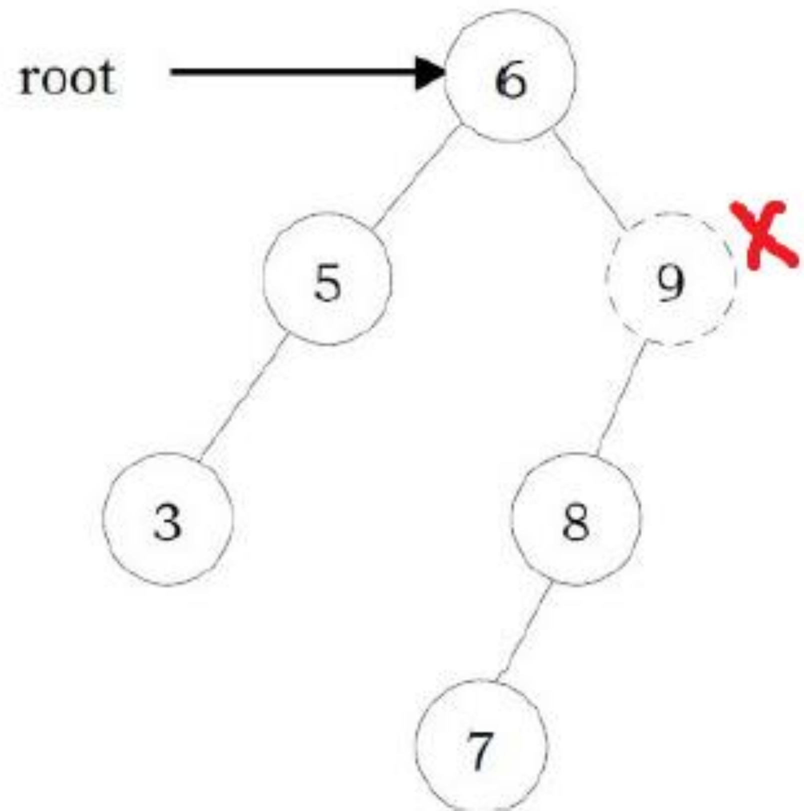
- Difference is at most 2 (in case of violation) at any given time

Types of Violations

- The violation may occur in four cases:

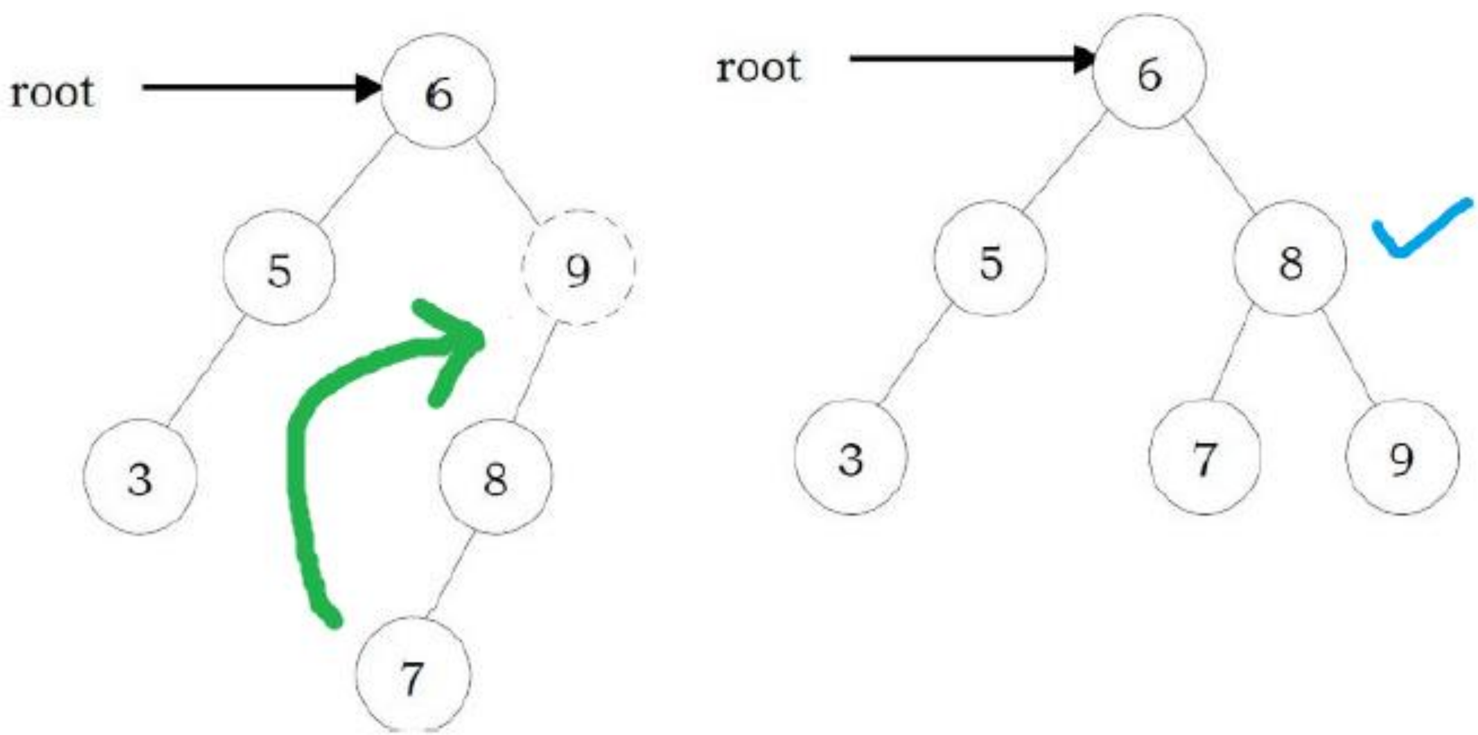
1) ***An insertion into the left subtree of the left child of node X*** *(L-L case)*
2) ***An insertion into the right subtree of the left child of node X*** *(L-R case)*
3) ***An insertion into the left subtree of the right child of node X*** *(R-L case)*
4) ***An insertion into the right subtree of the right child of node X*** *(R-R case)*

Case 1: L-L case

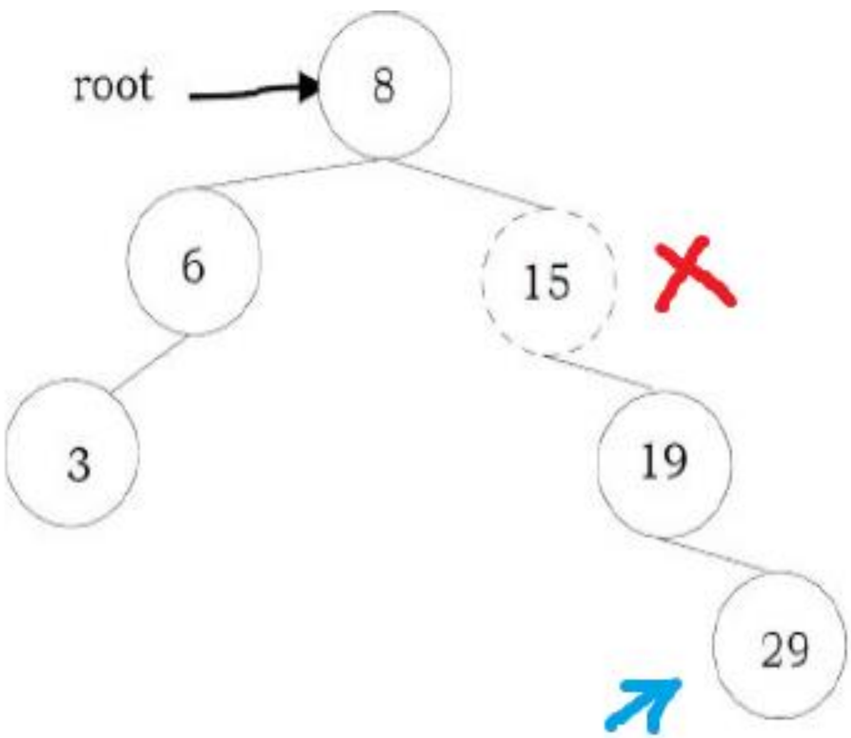


Solution: Single Right Rotation

Performing single right rotation:

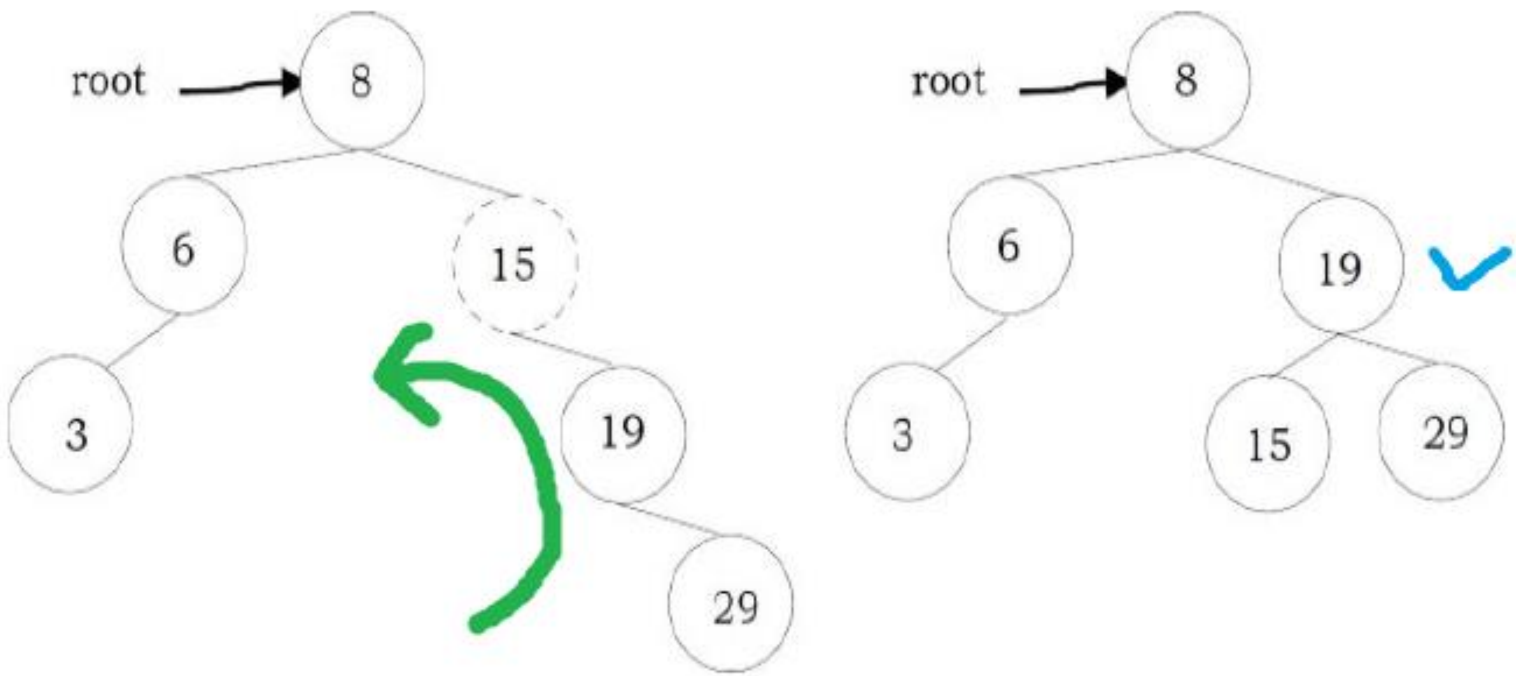


Case 4: R-R case

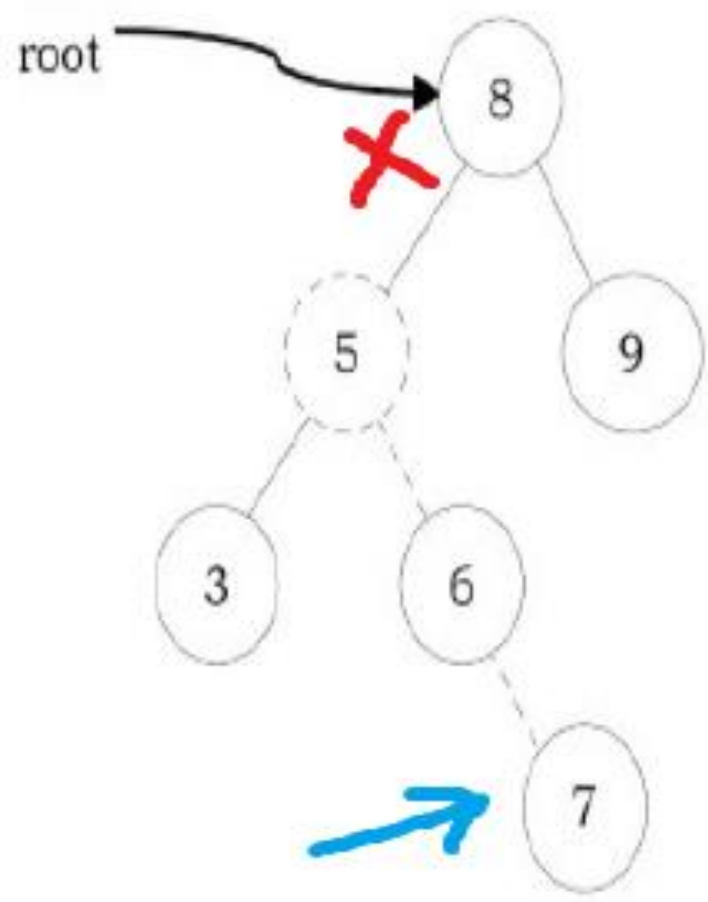


Solution: Single Left Rotation

Performing single left rotation:

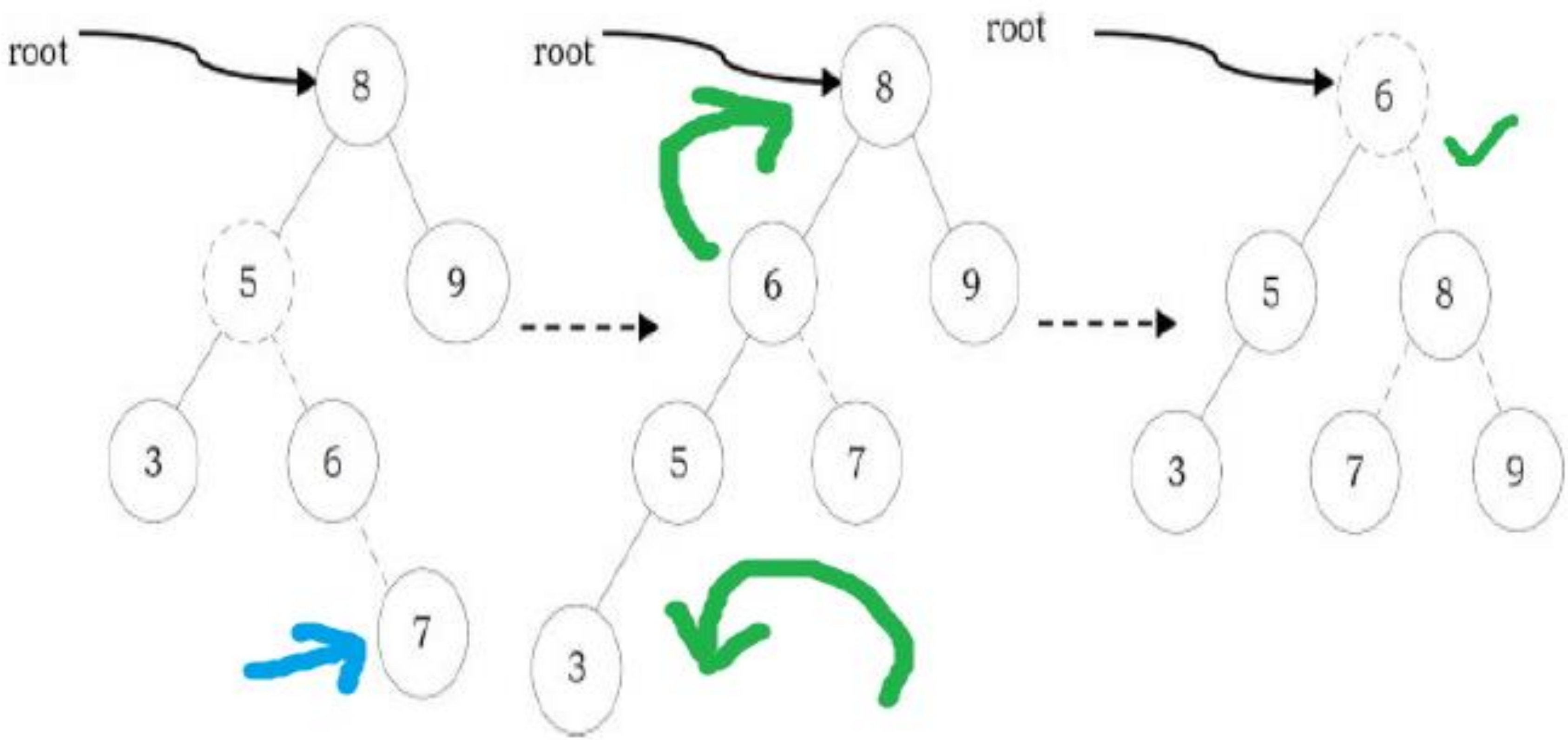


Case 2: L-R case

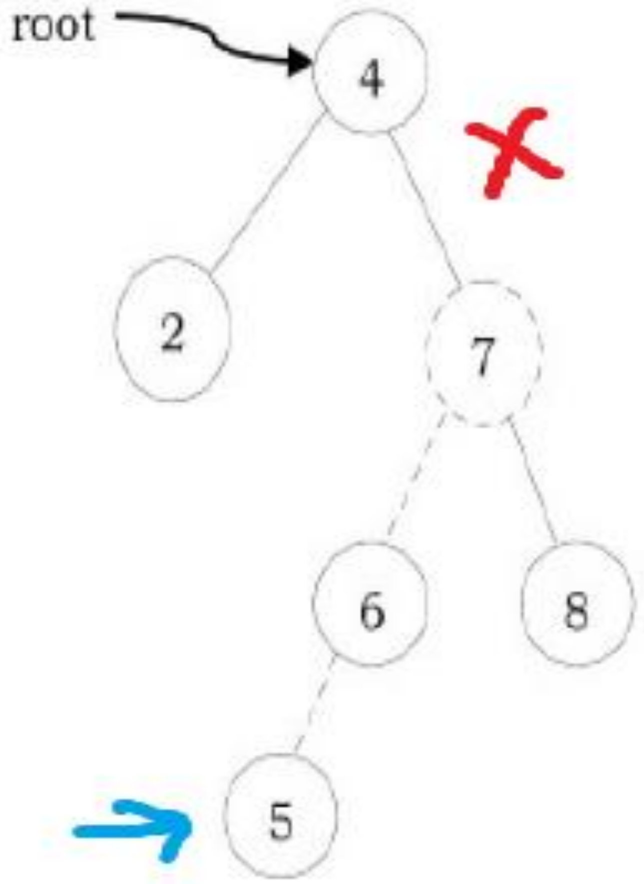


Solution: Double Rotation

Performing left rotation then right rotation

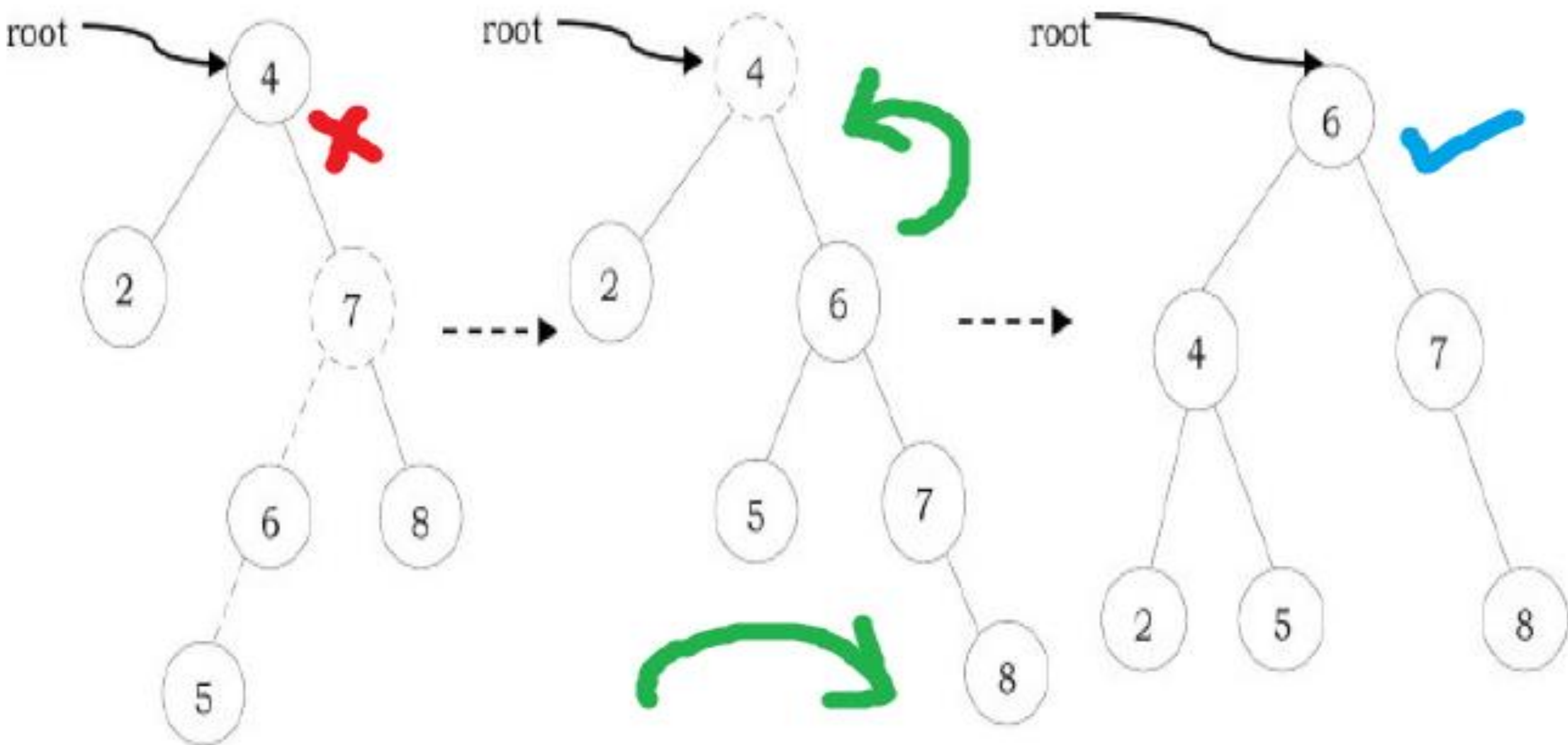


Case 3: R-L case



Solution: Double Rotation

- Performing right rotation then left rotation

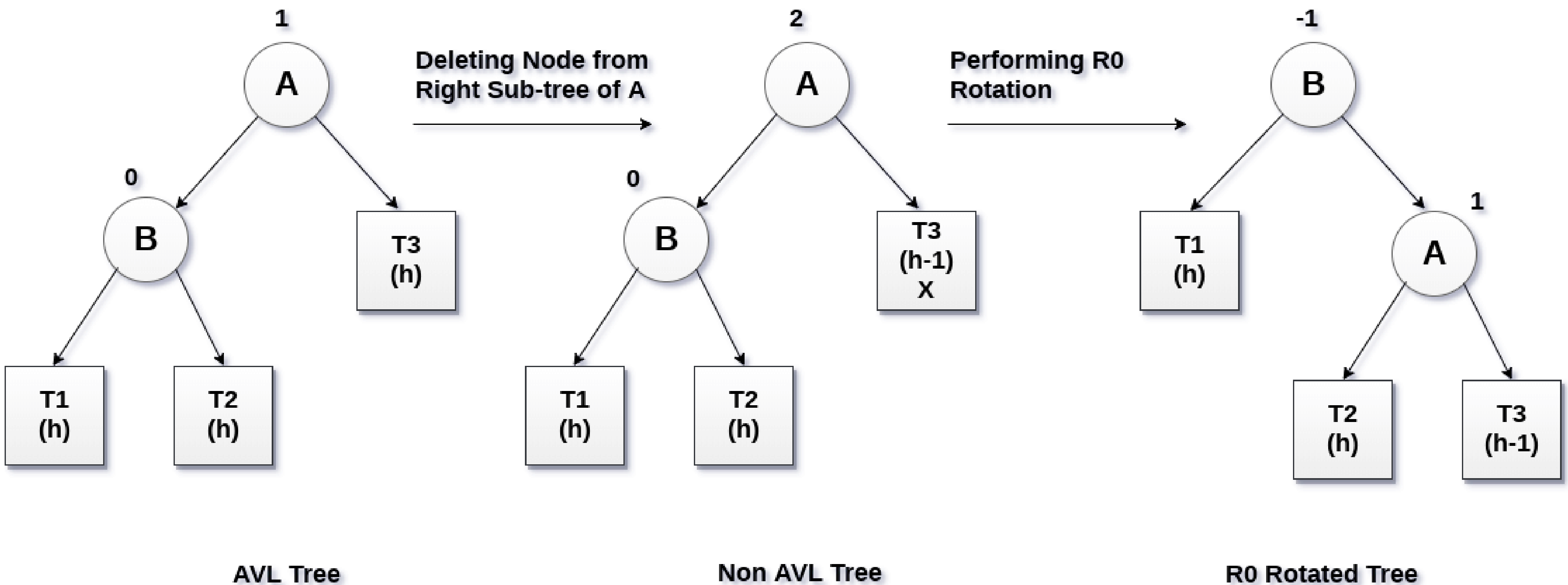


Deletion in AVL

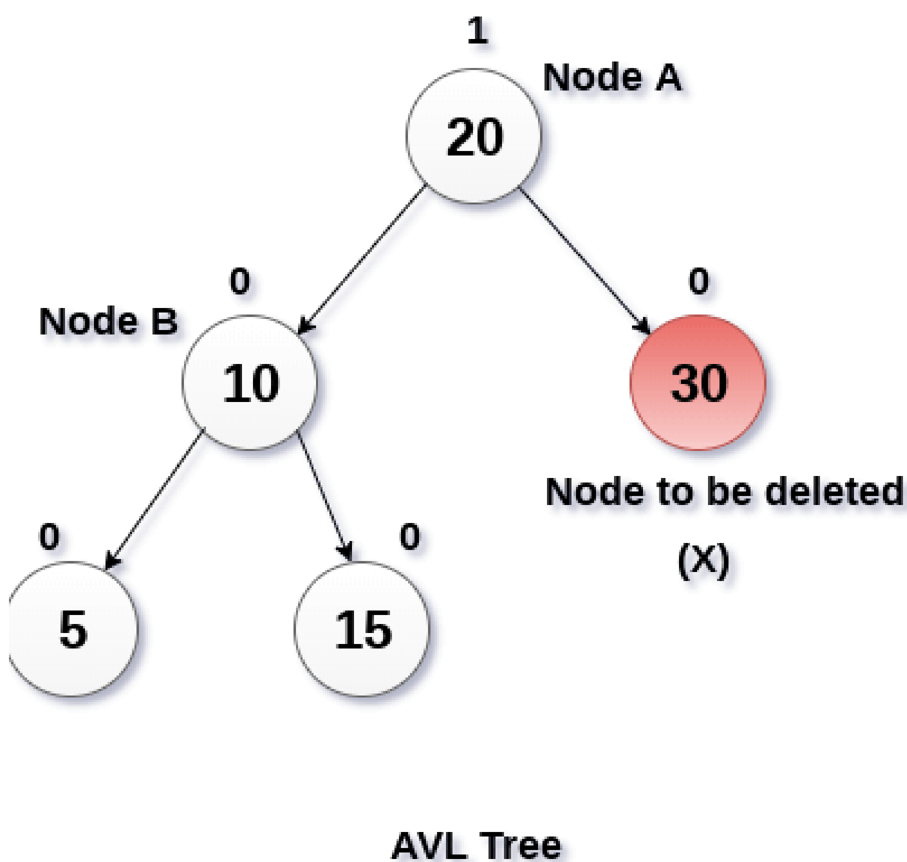
- Deletion is similar to any BST
- After deletion, we need to restore the balance of tree
- For that, we perform rotations

Case 1

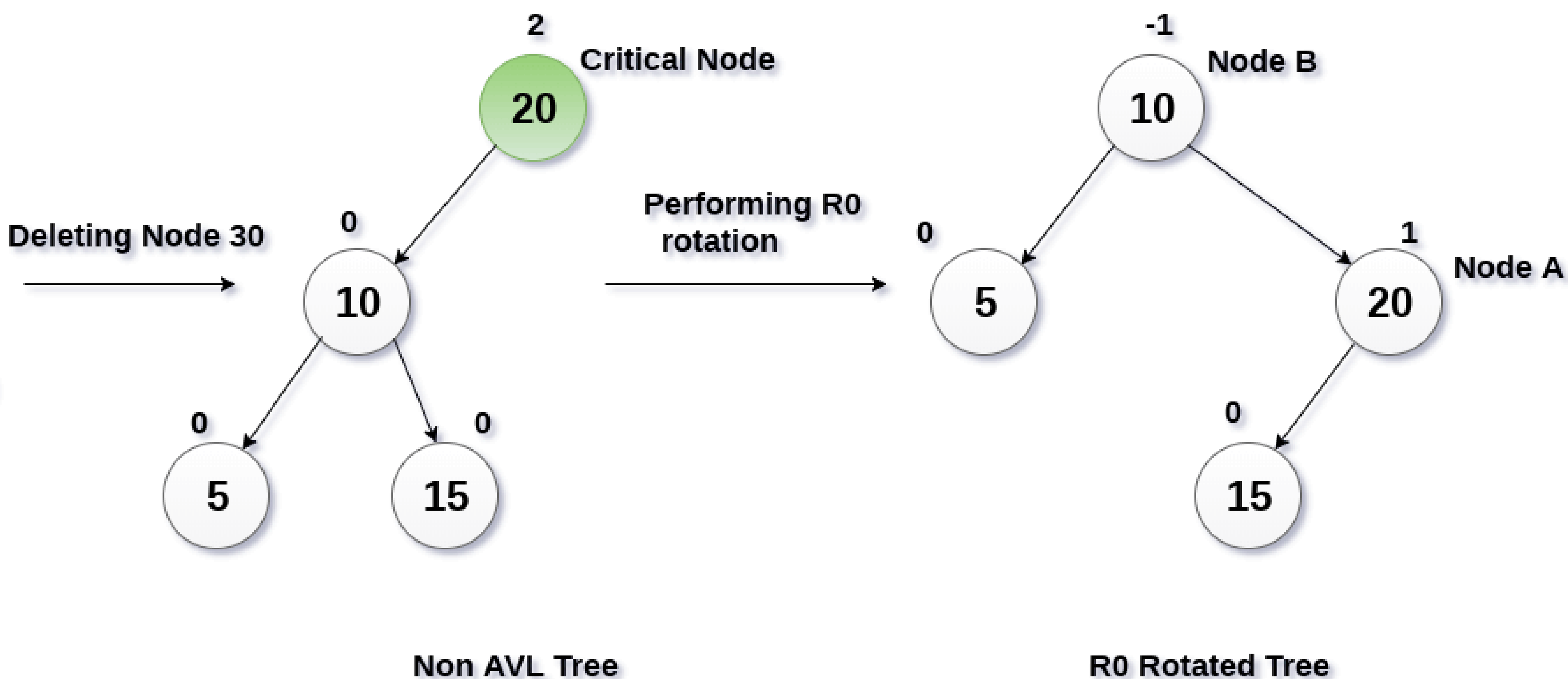
- When Node B has balance factor 0



Case 1: Example

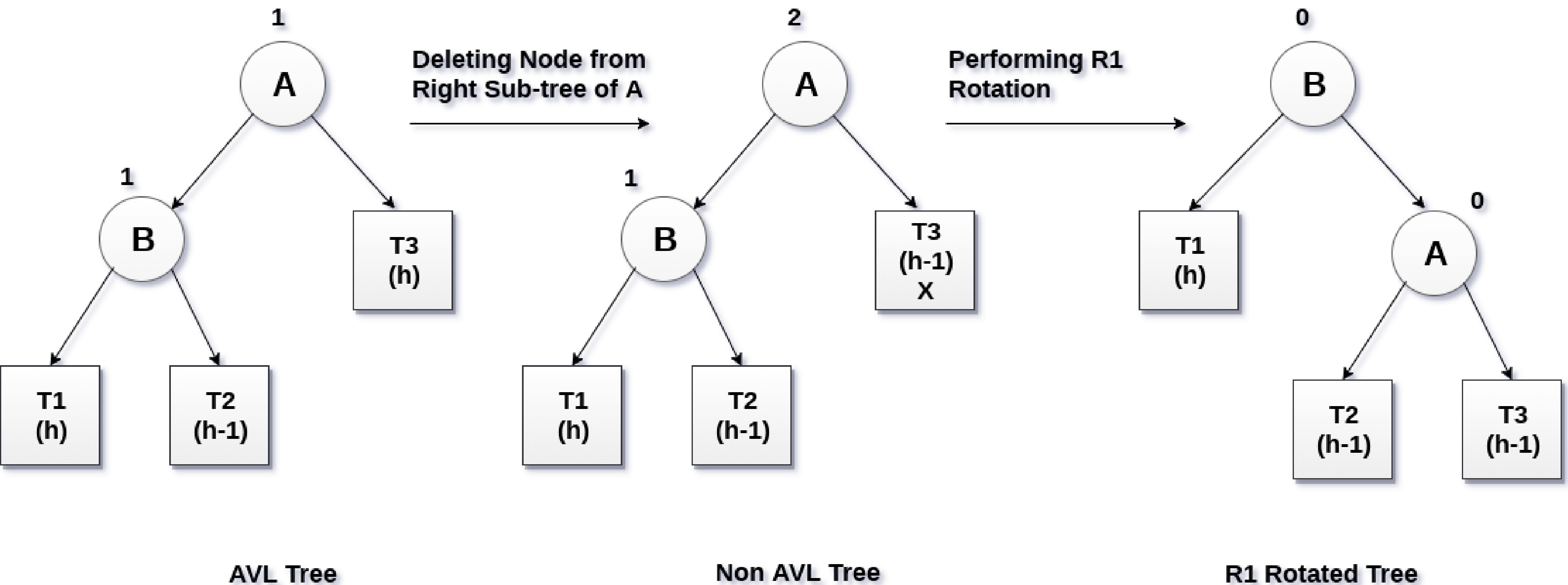


Case 1: Example

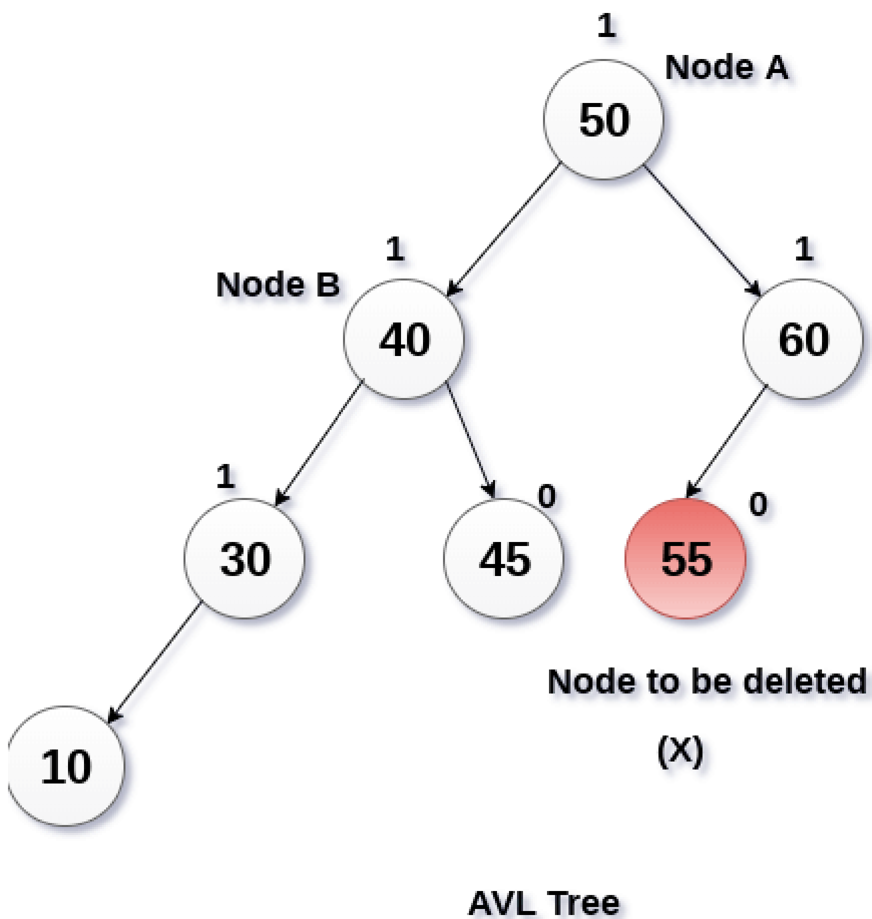


Case 2

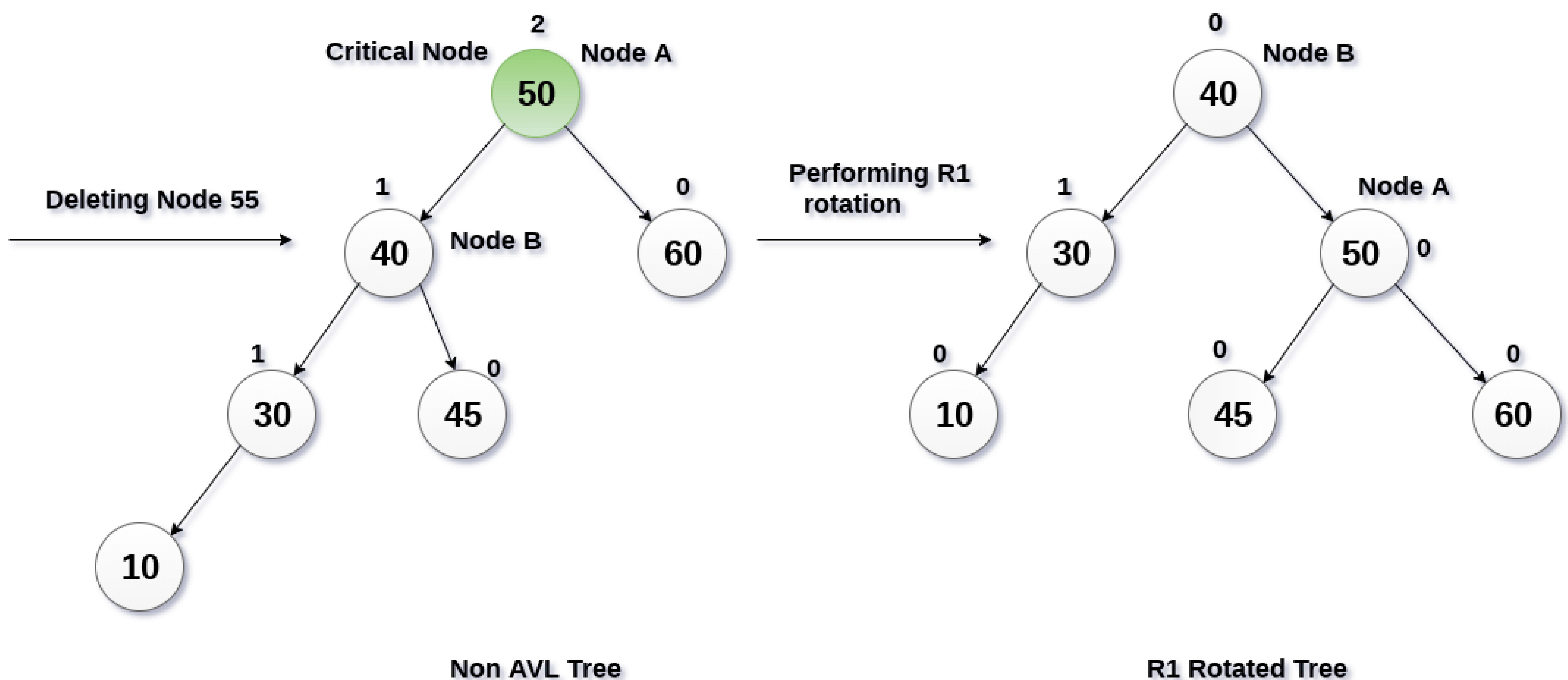
- Node B has balance factor 1



Case 2: Example

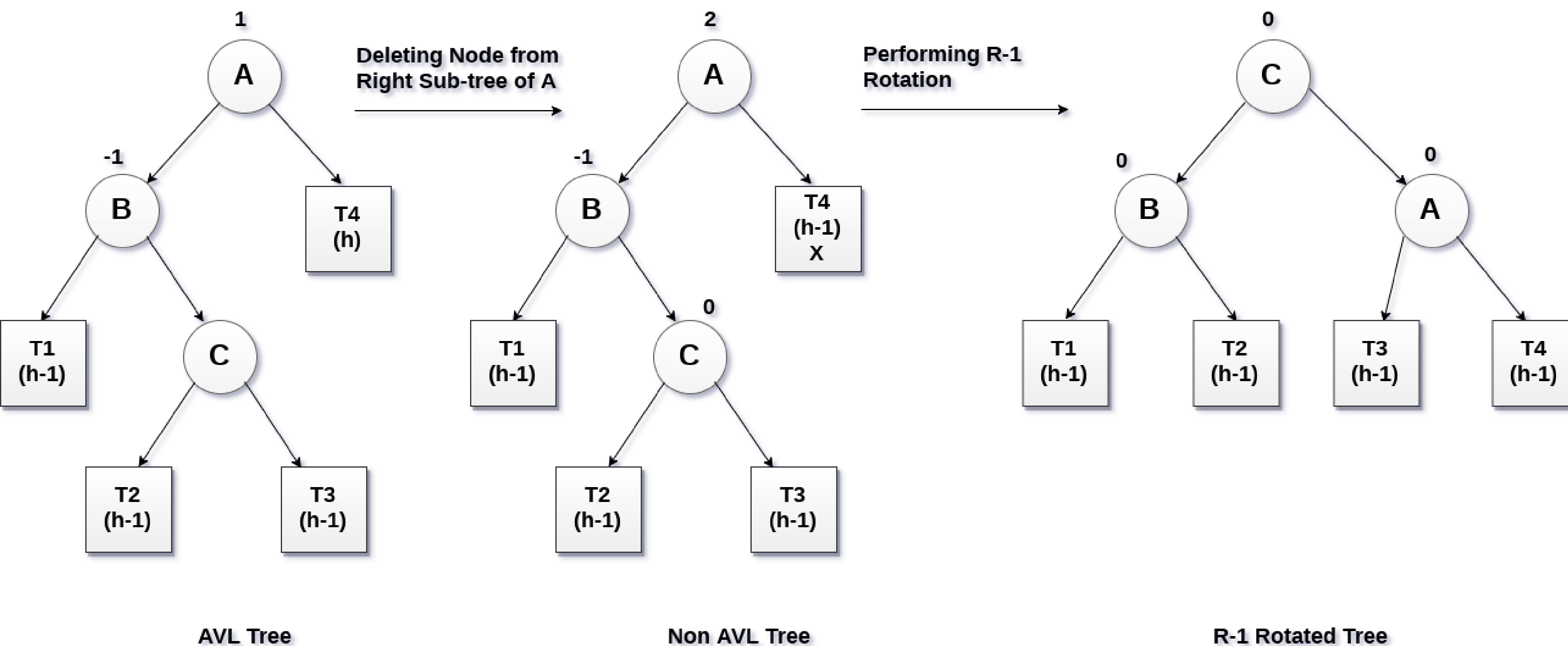


Case 2: Example

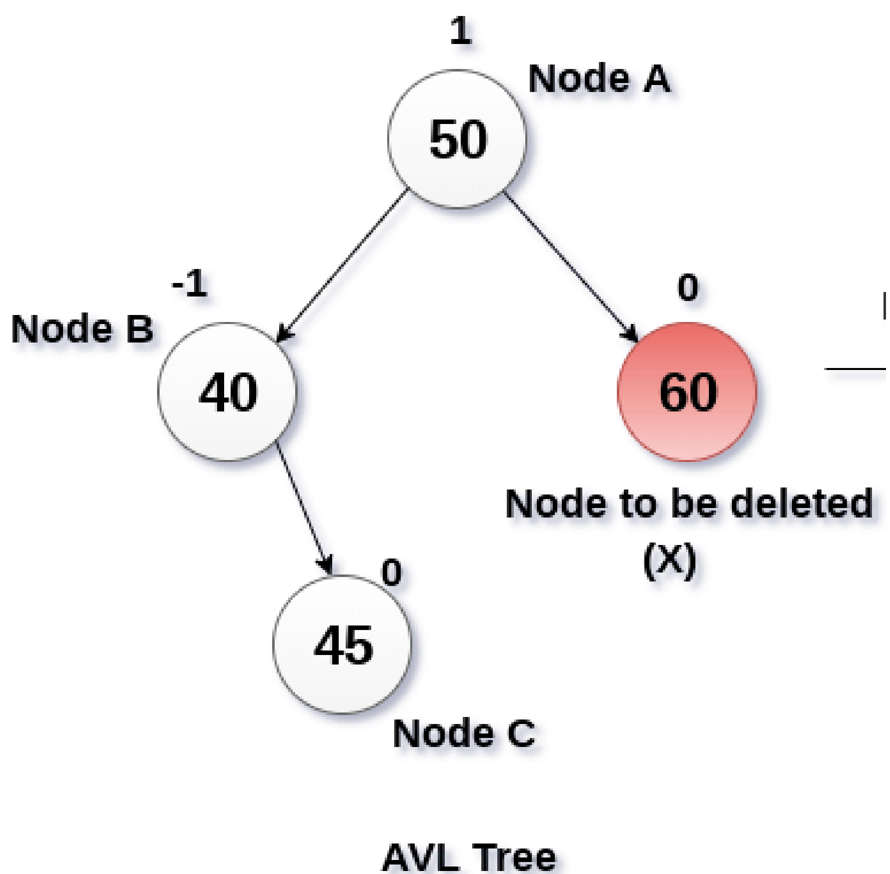


Case 3

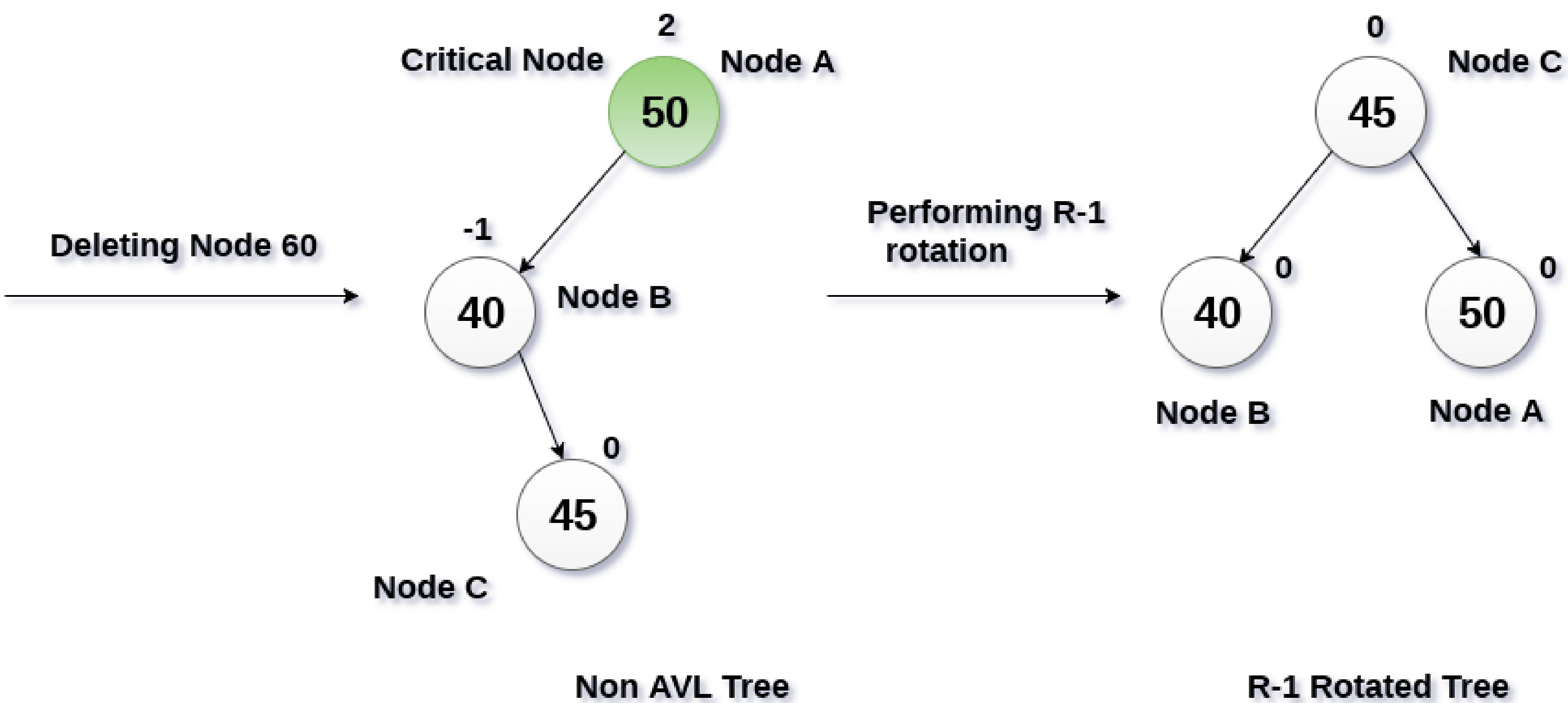
- When Node B has balance factor -1



Case 3: Example



Case 3: Example



Implementation: C++

- A C++ implementation can be found in the folder titled “Code” on course Google Drive