Data Structures (15B11CI311)

Odd Semester 2020

3rd Semester , Computer Science and Engineering

Jaypee Institute Of Information Technology (JIIT), Noida

Lecture: 30-31

Topics to be covered:

- AVL Tree
- Operations on AVL Tree

AVL Tree

- Searching in a BST has O(n) worst-case runtime complexity, when n is the height of the tree
- If we can maintain the height of a binary tree equals to O(log n), then search operation can be performed in O(log n)
- The trees with a worst-case height of O(log n) are called **balanced trees**
- An example of a balanced tree is **AVL** (**A**delson-**V**elsky and **L**andis) tree

Definition of AVL Tree

- It is a Binary Search Tree
- If T_L and T_R are the left and right subtrees of a nonempty binary tree (T), then T will be an AVL tree if and only if
  - T_L and T_R are also AVL trees, and
  - $|h_L - h_R| \leq 1$ where h_L and h_R are the heights of T_L and T_R respectively

Properties of AVL Tree

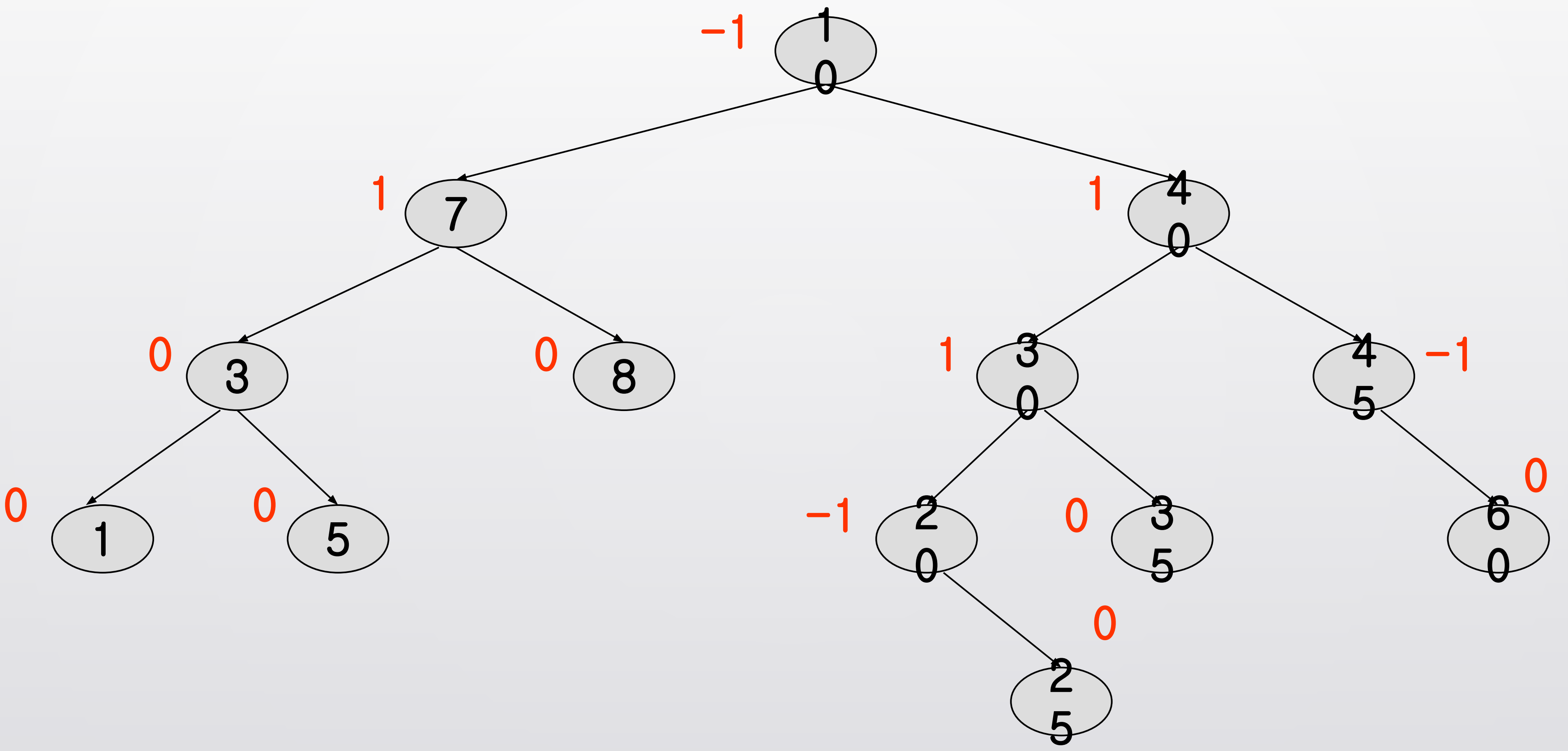
- Height of an AVL tree with n nodes: O(log n)
- Search time complexity in an n-node AVL tree = O(height) = O(log n)
- Insertion into an AVL tree can be done in O(log n) time and resultant tree is also AVL tree
- Deletion from an AVL tree can also be done in O(log n) time and resultant tree is also AVL tree

Representation Of AVL Tree

- Generally represented using the linked representation
- A balance factor (bf) is used with each node to perform the insertion and deletion
- The balance factor bf(x) of a node x is defined as
  - bf(x) = height(x->lchild) – height(x->rchild)
- Balance factor of each node in an AVL tree must be –1, 0, or 1

```
struct AVL
{
    int data;
    AVL *lchild, *rchild;
    int bf;
};
```

AVL Tree with Balance Factor



AVL Tree Operations

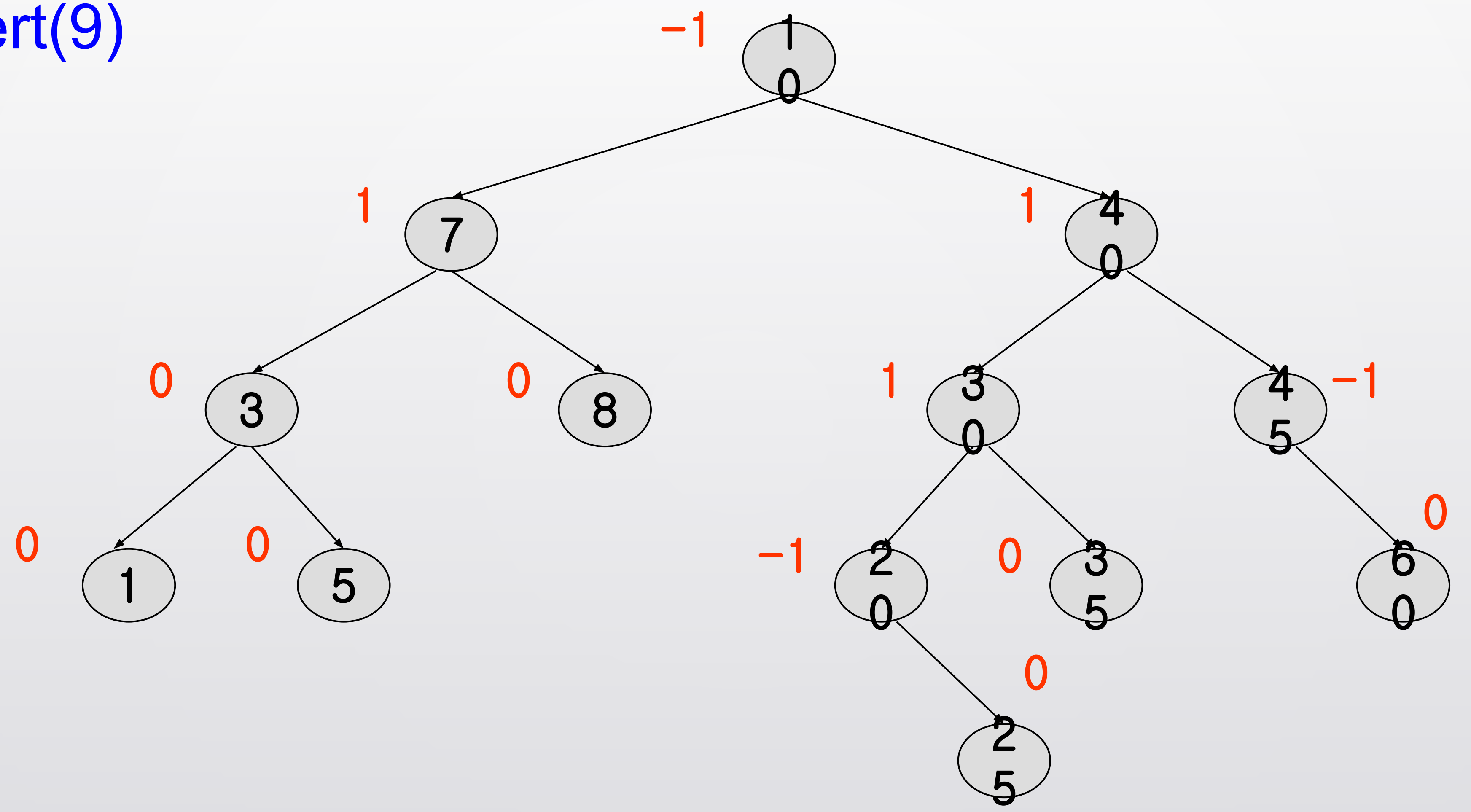
- Searching
  - Similar to Binary Search Tree in O(log n)
- Insertion
- Deletion

Insertion/Deletion in an AVL Tree

- Perform the Insertion or deletion in AVL tree like Binary Search Tree
- After performing the operation, check the balance factor of each node
  - **Case-01:**
    - The balance factor of each node is either 0 or 1 or -1
    - The operation is concluded
  - **Case-02:**
    - The balance factor of at least one node is not 0 or 1 or -1
    - That is, the AVL tree is not balanced
    - Perform suitable Rotations to balance the tree

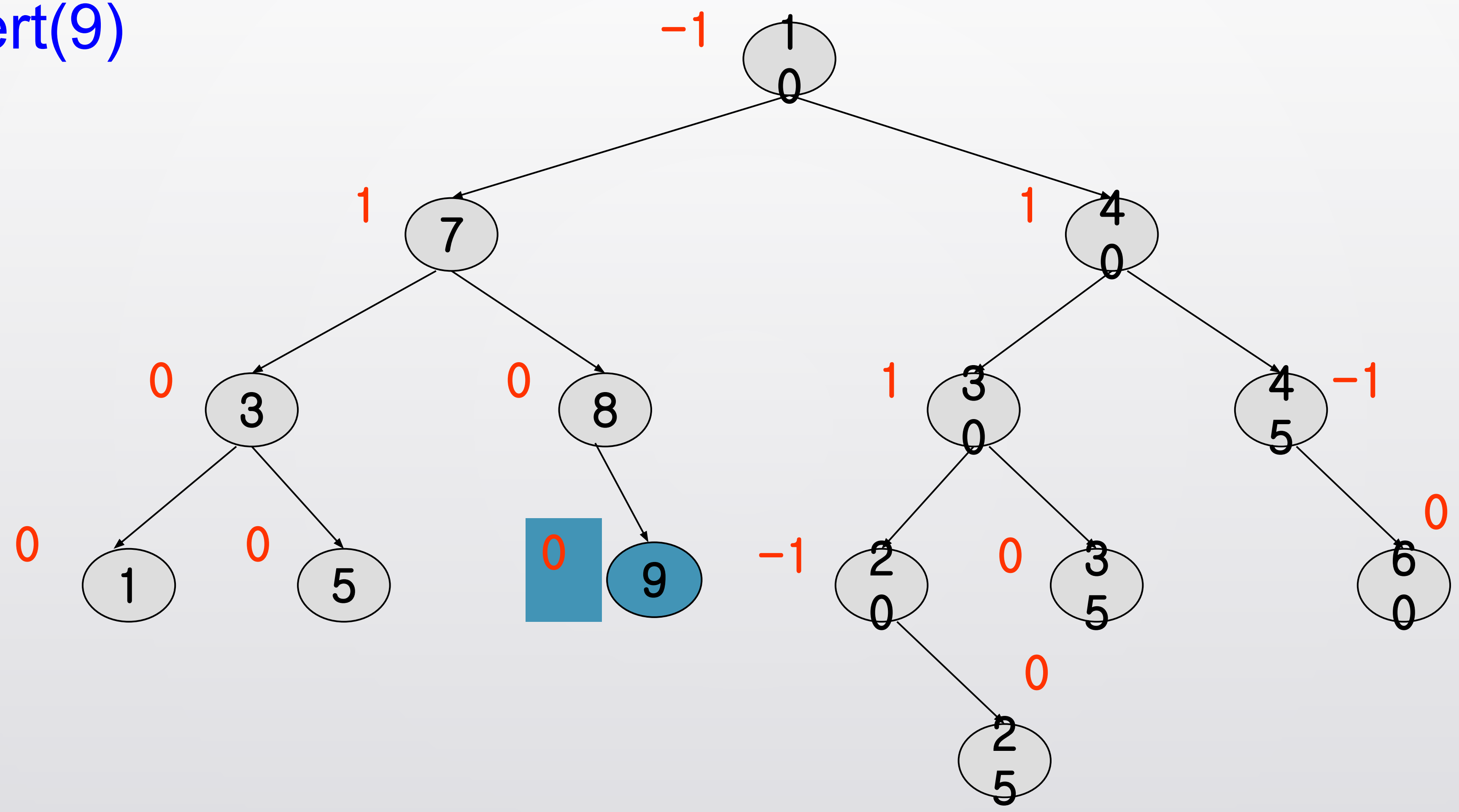
Insertion/Deletion in an AVL Tree (<u>Case-01</u>)

Insert(9)



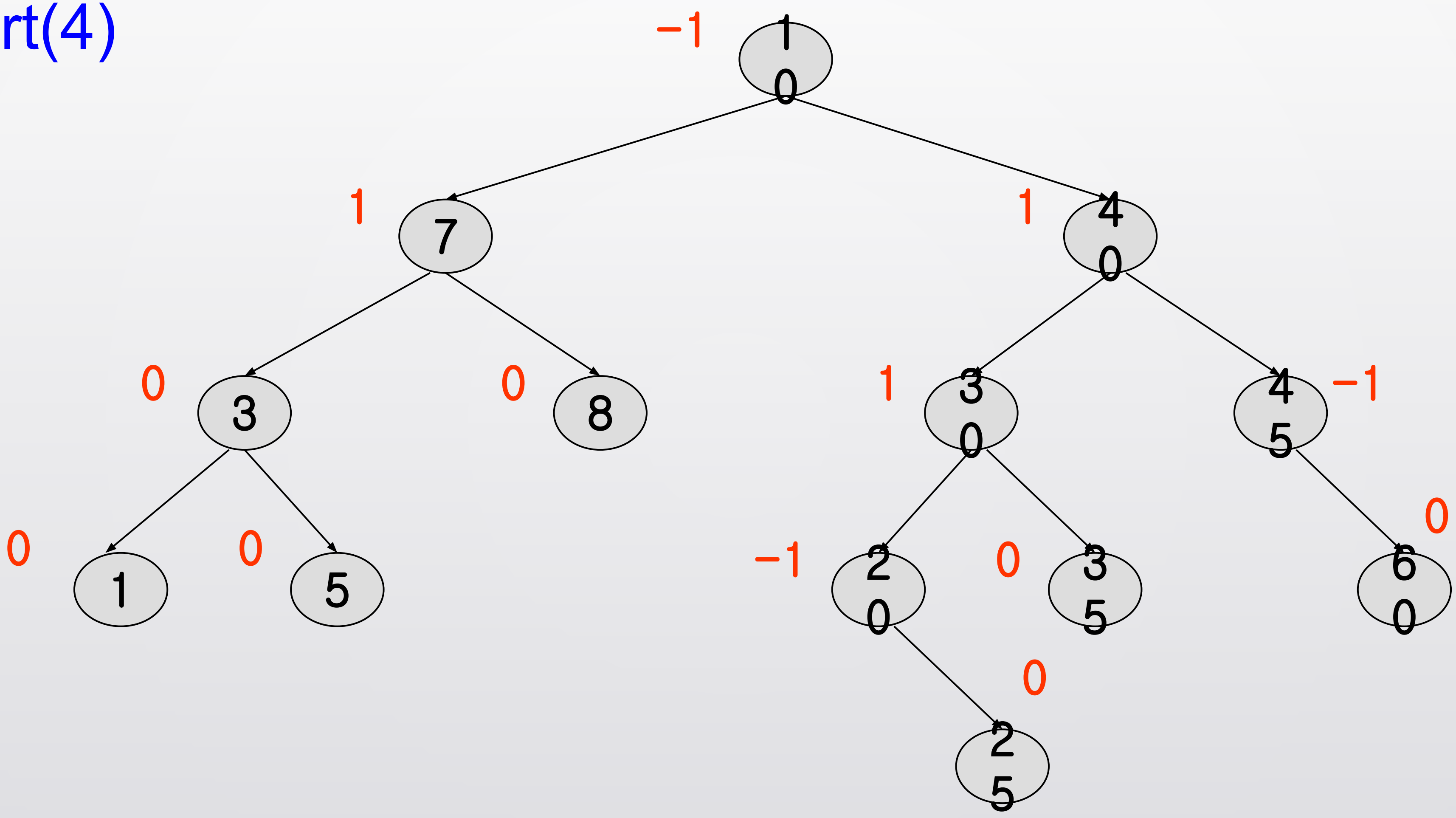
Insertion/Deletion in an AVL Tree Case-01

Insert(9)



Insertion/Deletion in an AVL Tree Case-02

Insert(4)

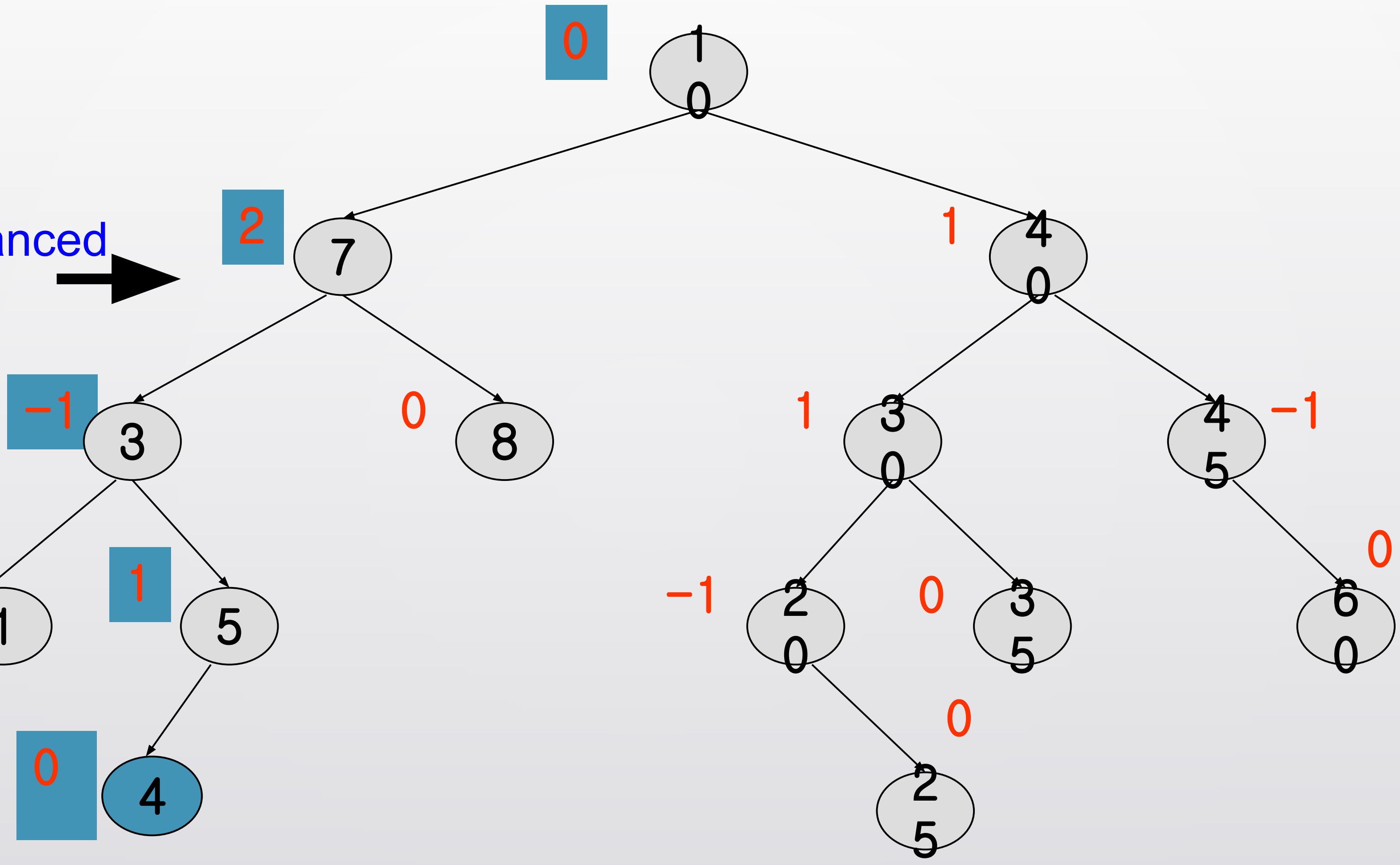


Insertion/Deletion in an AVL Tree Case-02

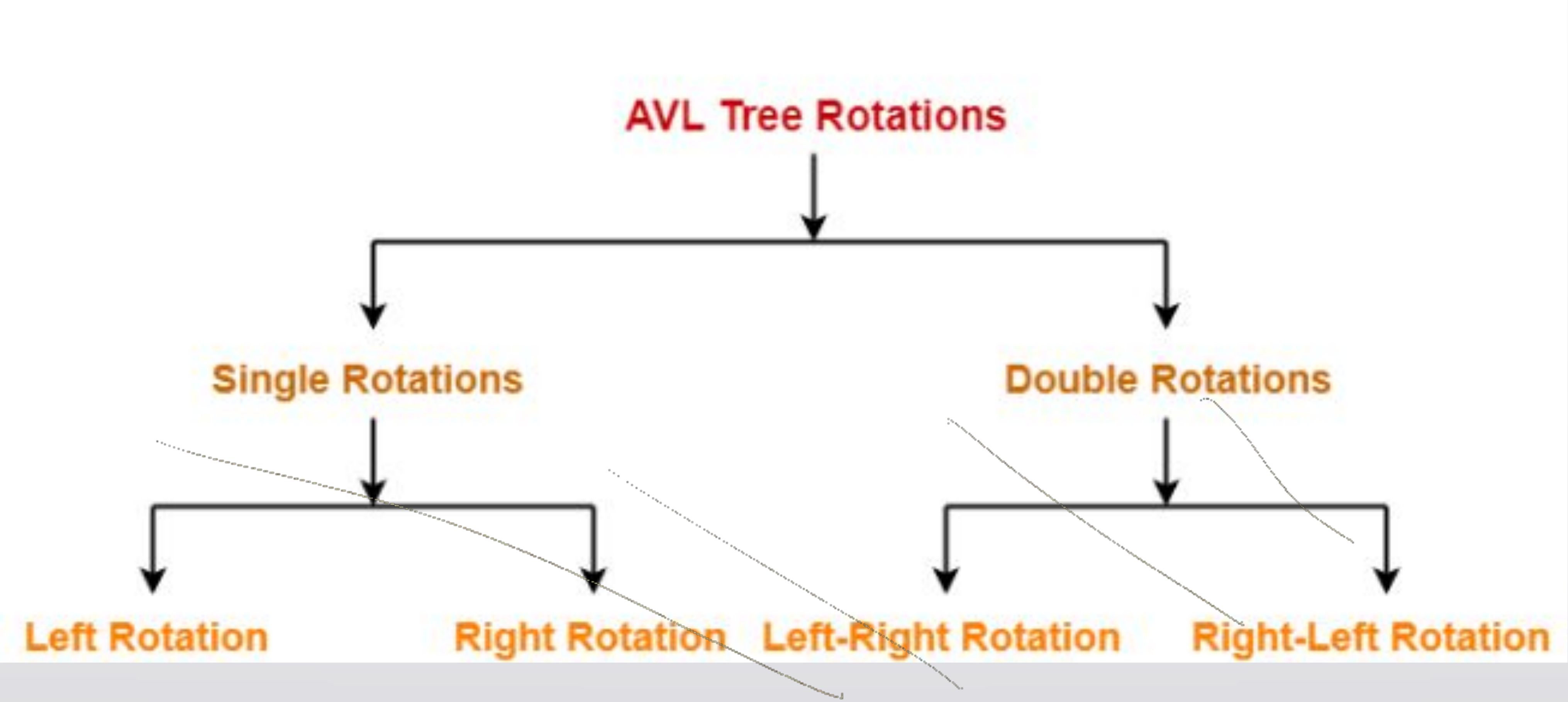
Insert(4)

Tree becomes unbalanced

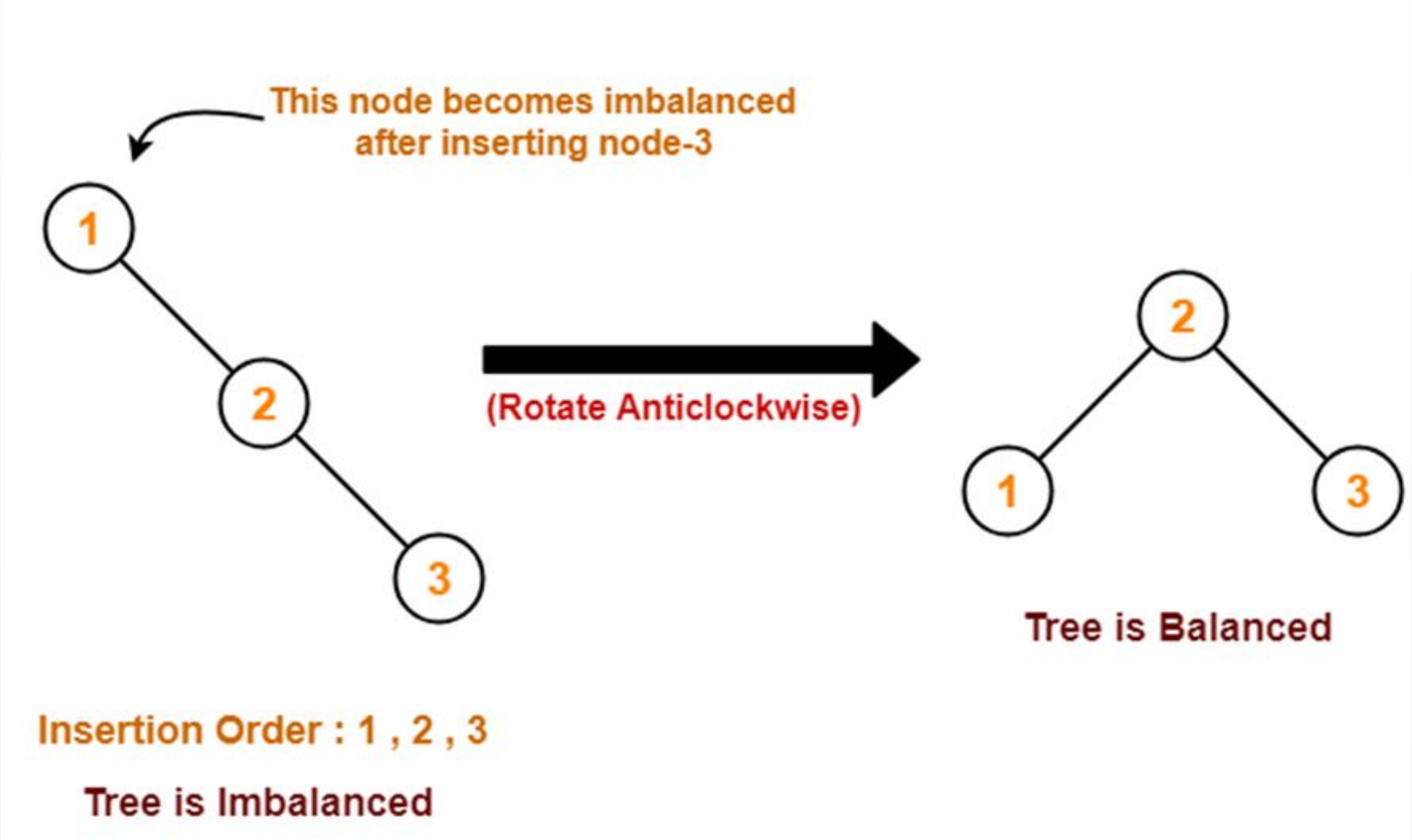
Perform Rotation



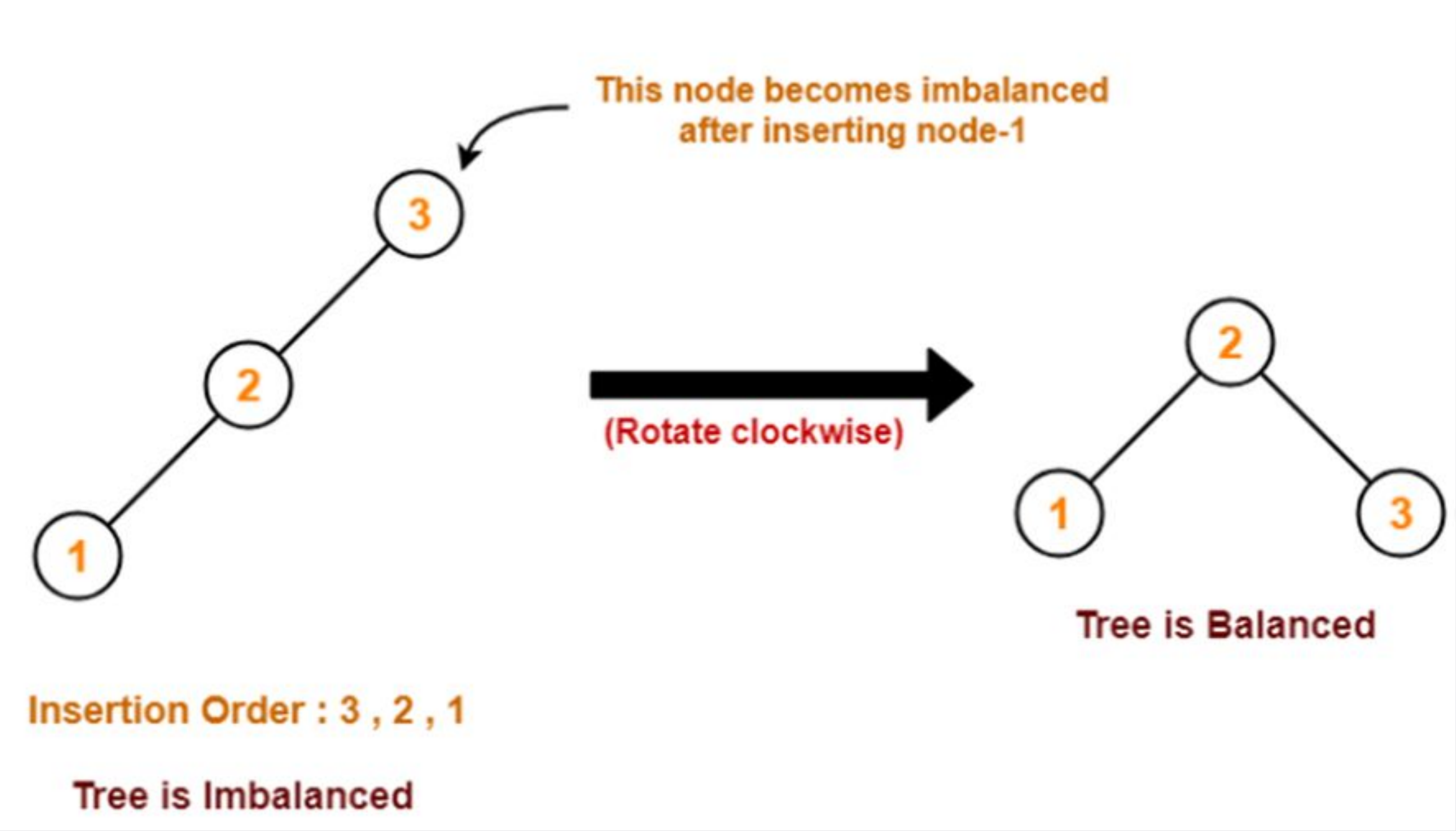
AVL Tree Rotations



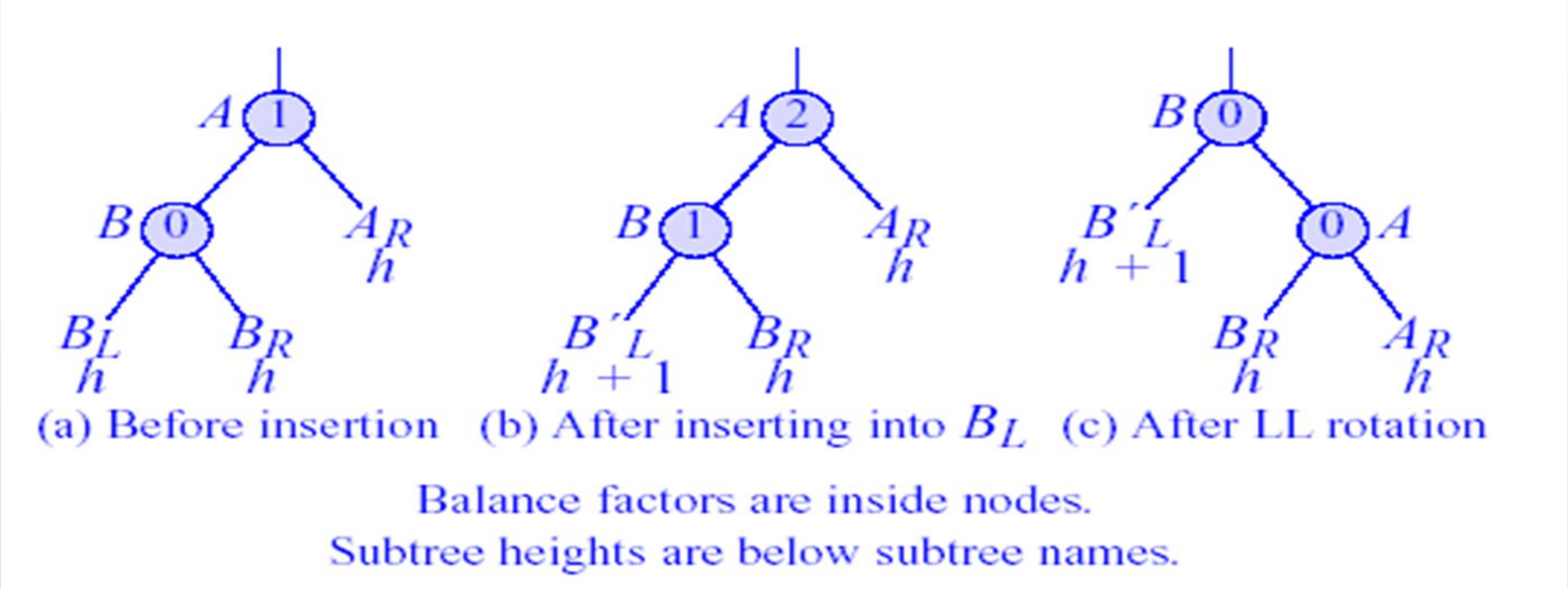
Case-01: Left Rotation



Case-02: Right Rotation

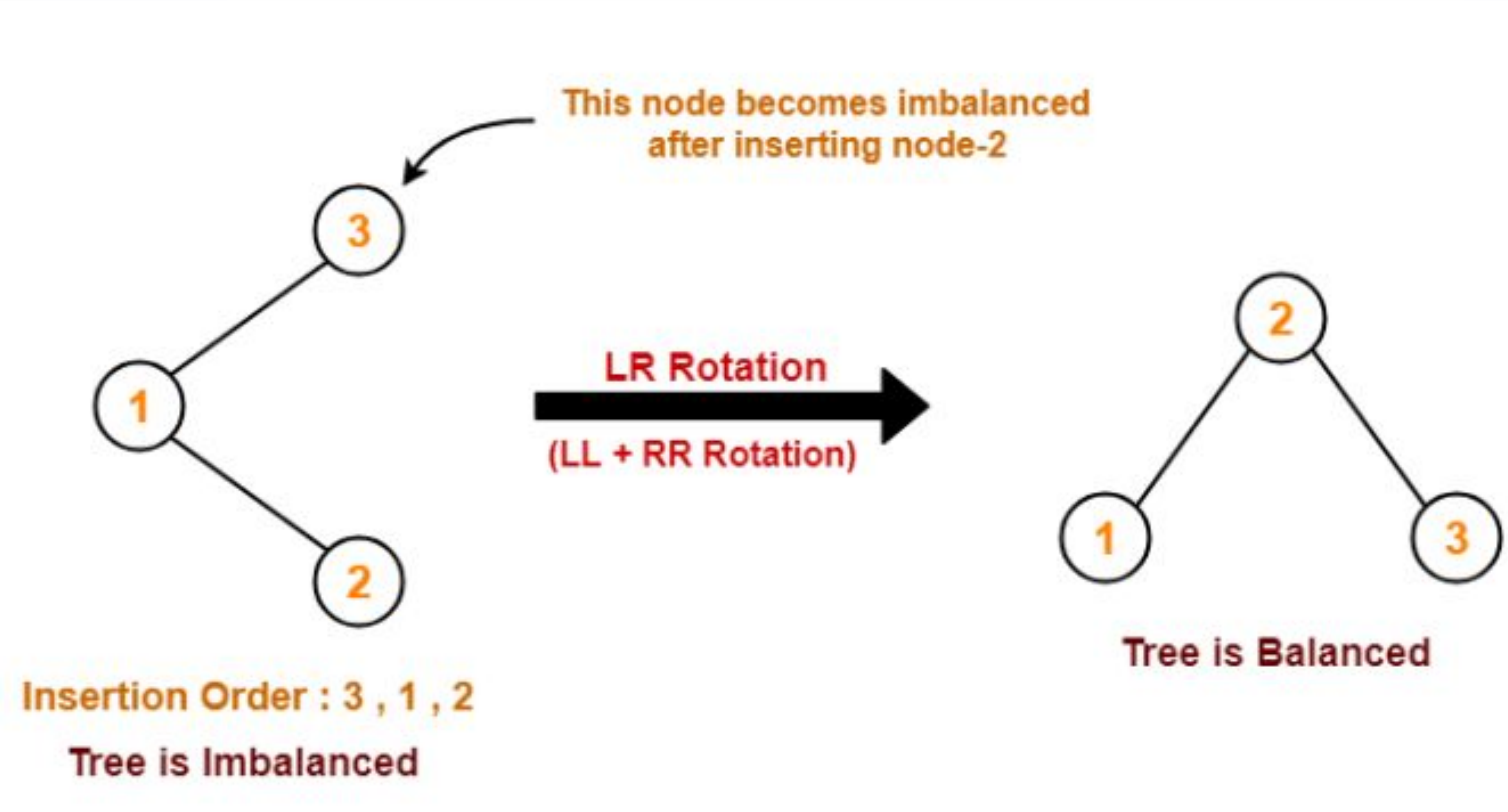


Case-01: Right Rotation

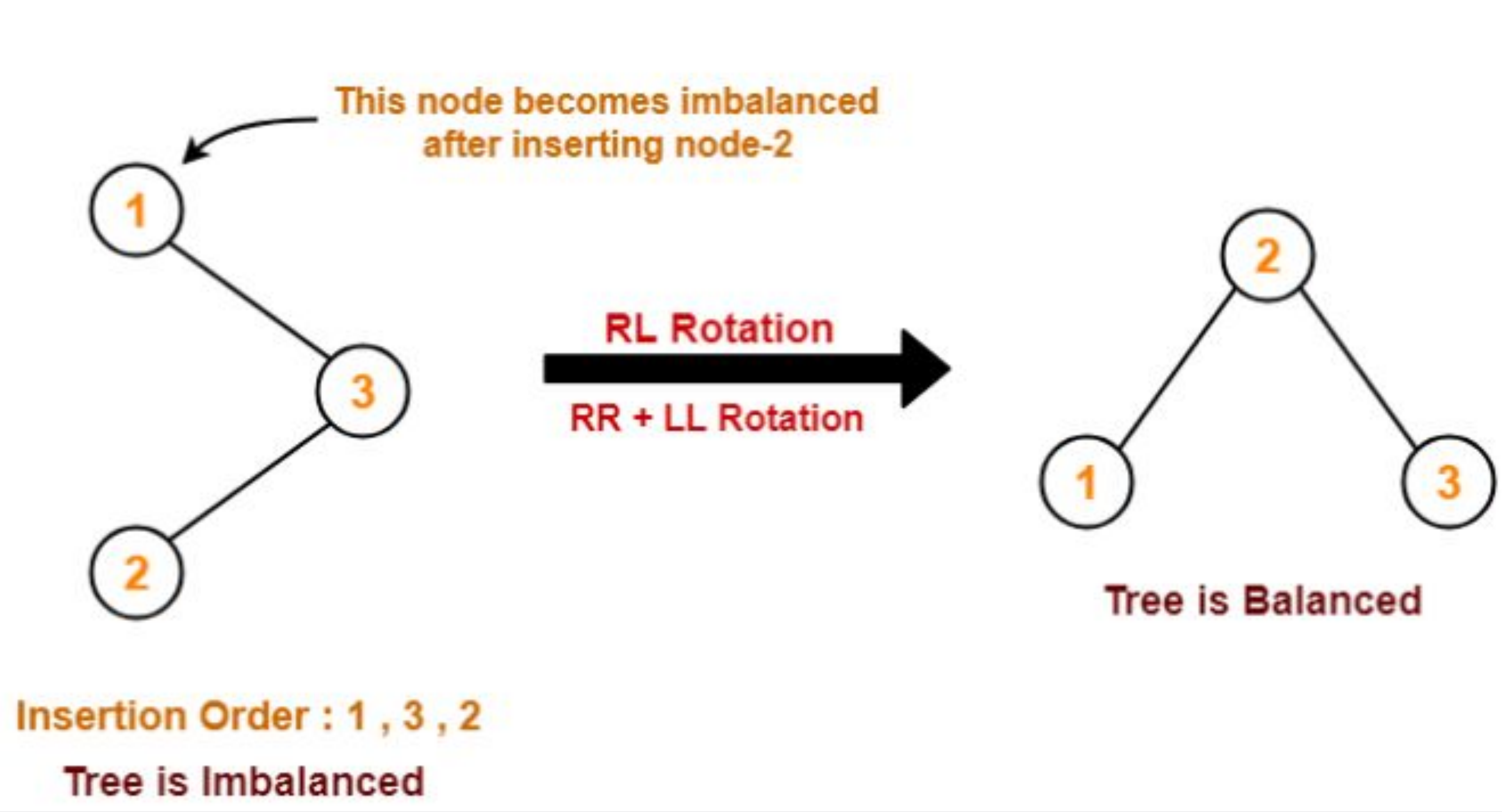


- Similarly, we can do the rotation for Left rotation

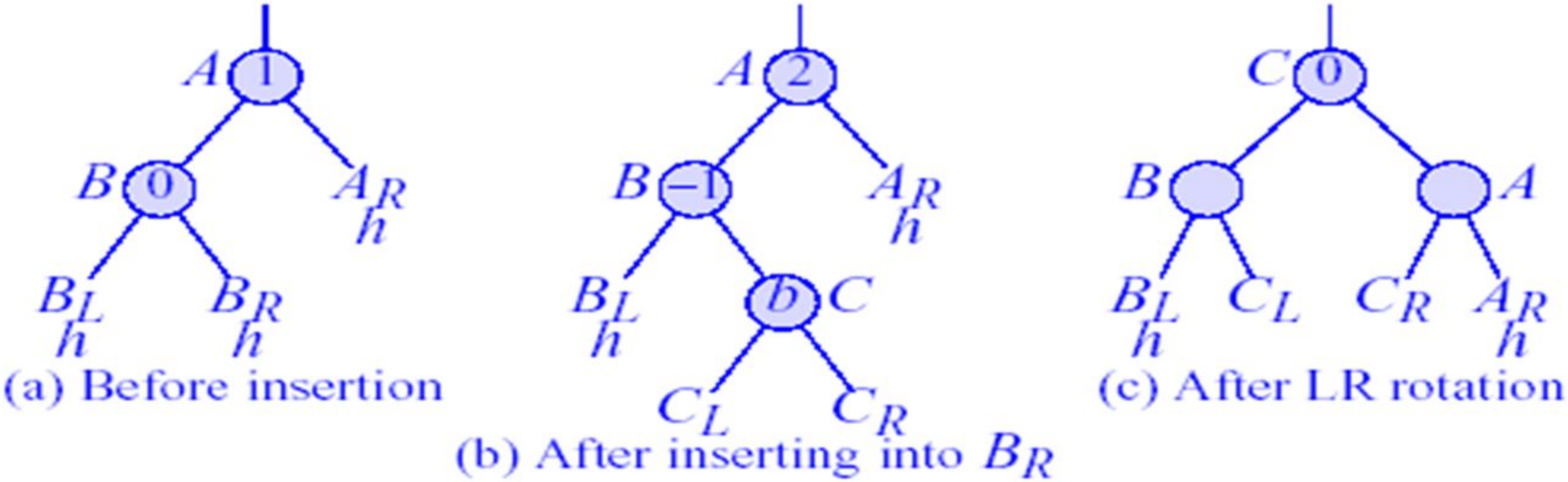
Case-03: LR Rotation



Case-04: RL Rotation



Case-01: LR Rotation



$b = 0 \Rightarrow bf(B) = bf(A) = 0$ after rotation

$b = 1 \Rightarrow bf(B) = 0$ and $bf(A) = -1$ after rotation

$b = -1 \Rightarrow bf(B) = 1$ and $bf(A) = 0$ after rotation

An LR Rotation

- Similarly, we can do the rotation for RL rotation

Insertion in an AVL Tree: Example

- Insert the following elements in an AVL tree and do the required rotations to balance it.

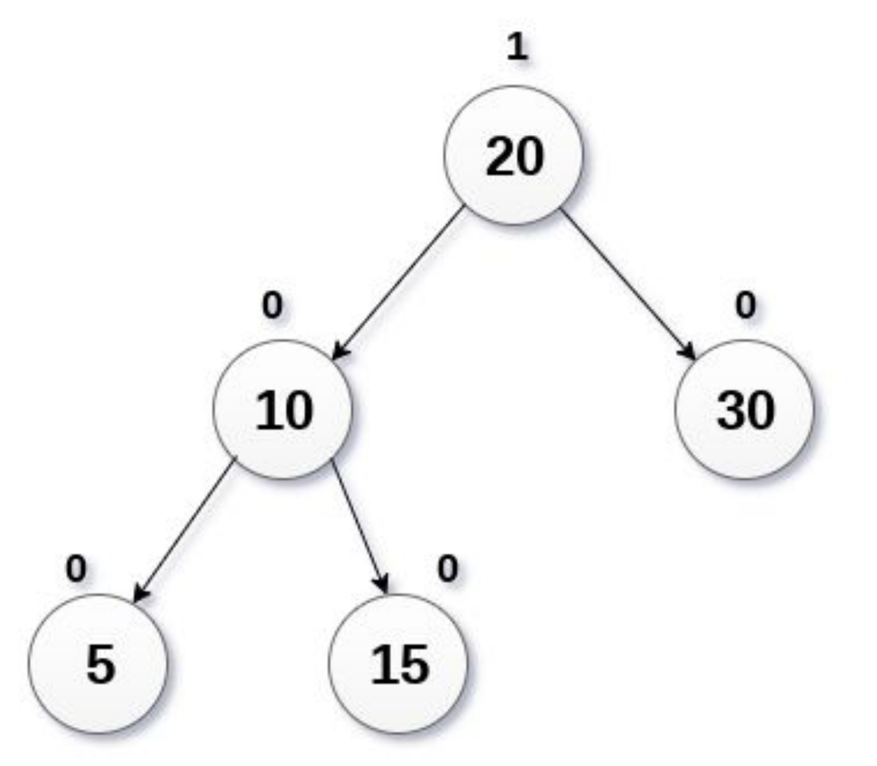
- 10, 7, 27, 9, 40, 3, 8, 30, 45, 1, 5, 20, 35, 60, 25, 4, 29,

Deletion in an AVL Tree

- Perform the deletion in AVL tree like Binary Search Tree
- After performing the operation, check the balance factor of each node
  - **<u>Case-01:</u>**
    - The balance factor of each node is either 0 or 1 or -1
    - The operation is concluded
  - **<u>Case-02:</u>**
    - The balance factor of at least one node is not 0 or 1 or -1
    - That is, the AVL tree is not balanced
    - Perform suitable Rotations, as described previously, to balance the tree

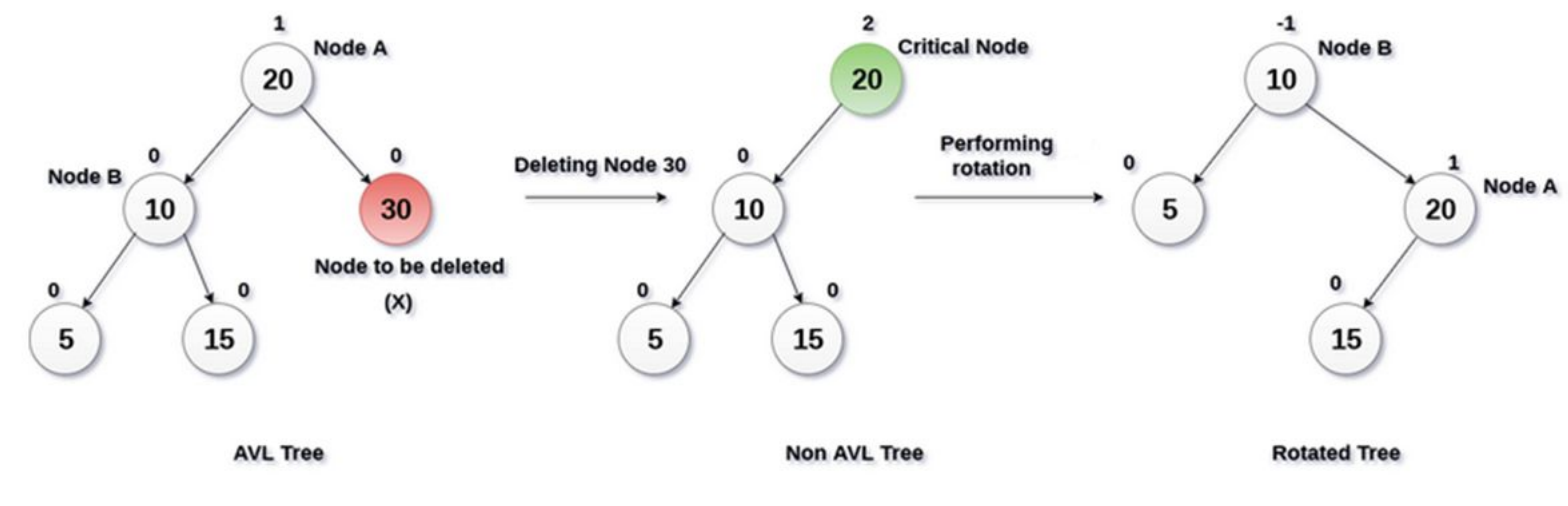
Deletion in an AVL Tree

- Delete 30 from the below AVL tree



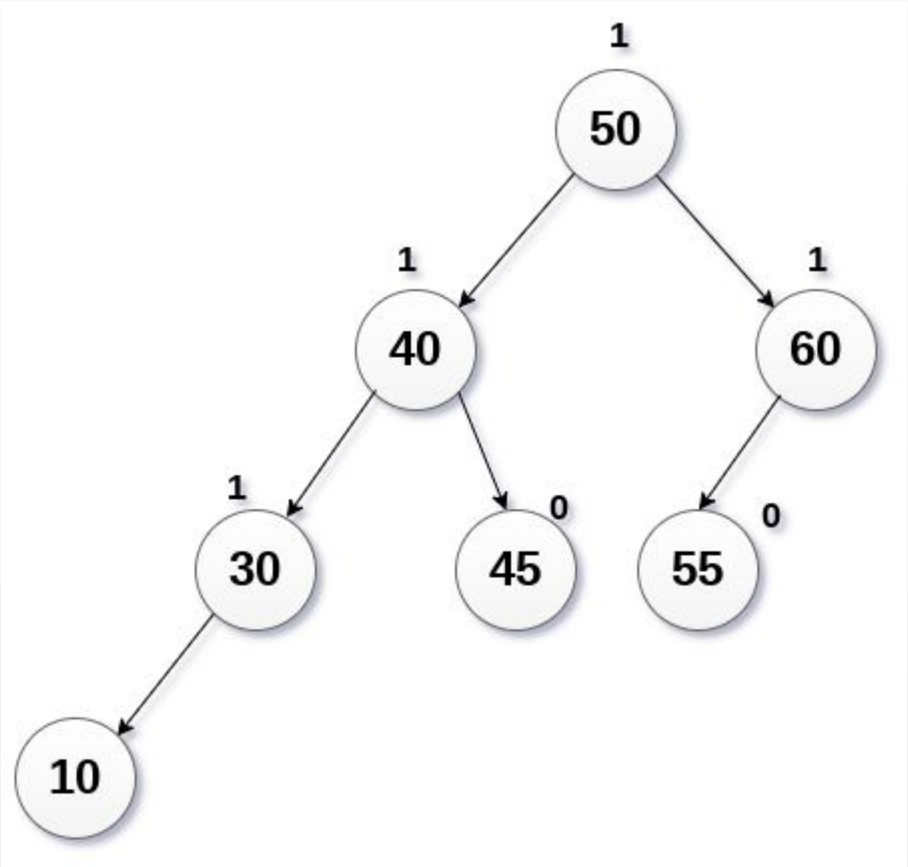
Deletion in an AVL Tree

- Delete 30 from the below AVL tree



Deletion in an AVL Tree

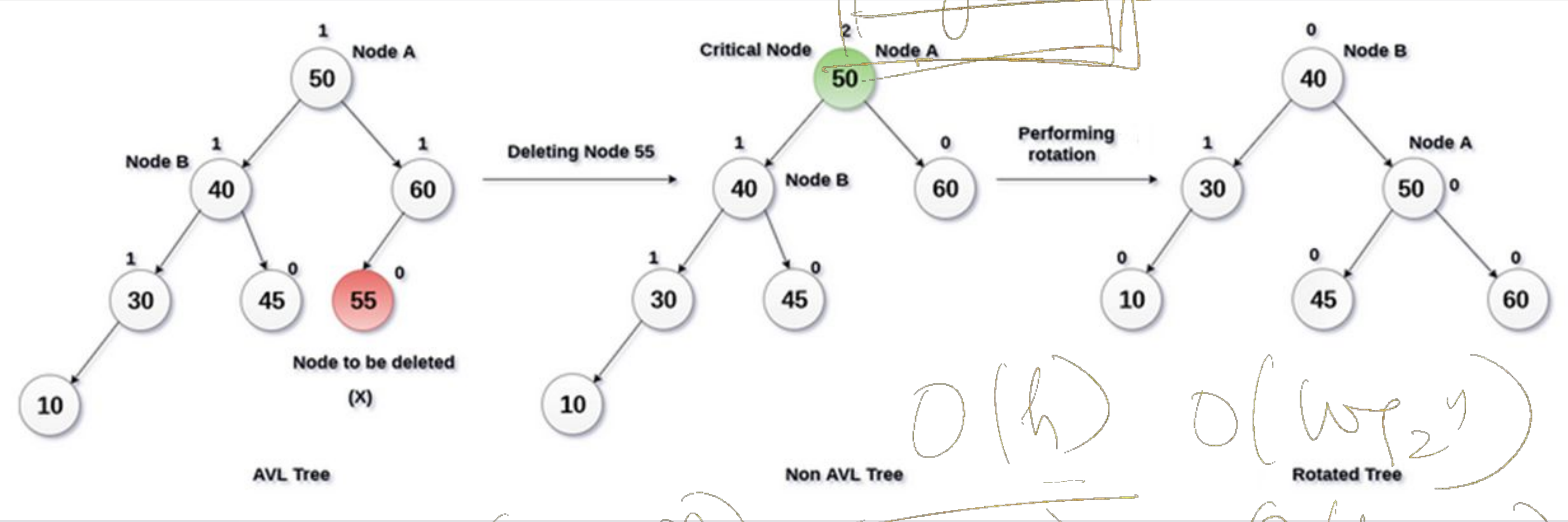
- Delete 55 from the below AVL tree



Deletion in an AVL Tree

- Delete 55 from the below AVL tree

$\log_2(n)$



$O(h)$ $O(\log_2 n)$

$O(\log_2 n)$ $O(h) \rightarrow O(\log_2 n)$

Reference

- https://nptel.ac.in/courses/106/103/106103069/
- https://www.cs.cmu.edu/~wlovas/15122-r11/lectures/18-avl.pdf
- https://www.gatevidyalay.com/avl-tree-avl-tree-example-avl-tree-rotation/
- https://www.javatpoint.com/deletion-in-avl-tree
- https://www.javatpoint.com/avl-tree
- http://dpnm.postech.ac.kr/cs233/