BML MUNJAL UNIVERSITY™

Data Structures & Algorithms

Tree

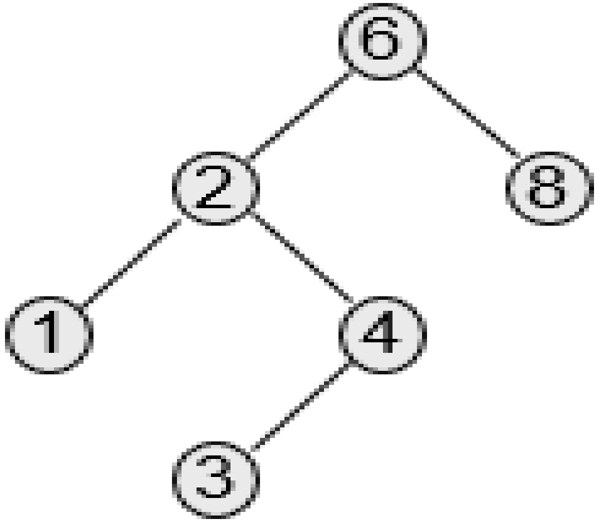
AVL Tree

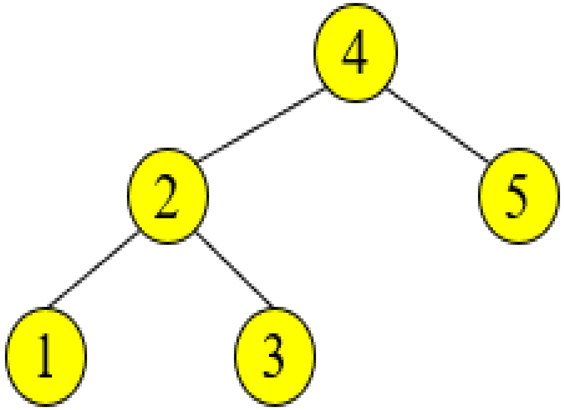
AVL Tree

AVL is named for its inventors: Adel’son, Vel’skii and Landis.

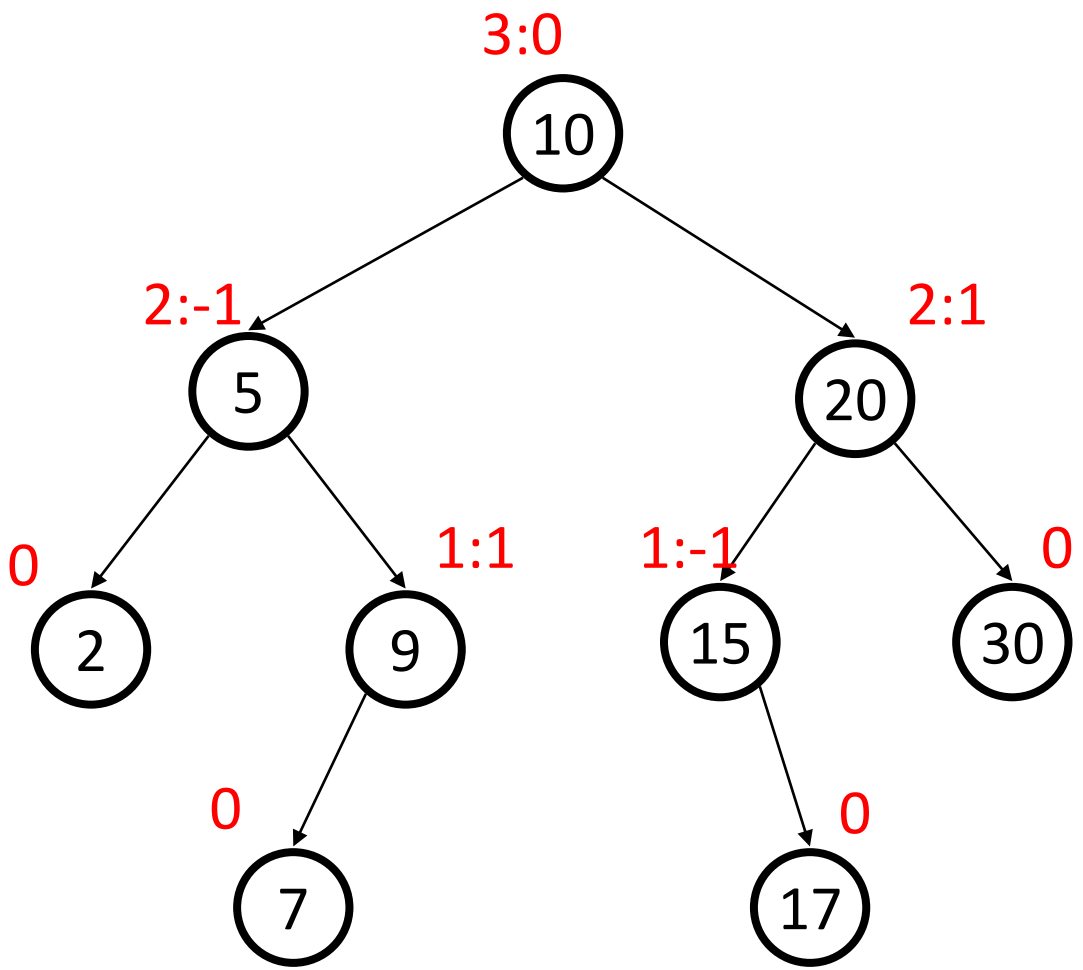
An AVL tree is a binary search tree such that for any node in the tree, the height of the left and right subtrees can differ by at most 1.

With each node of the AVL tree is associated a balance factor which is defined as height(left subtree) - height(right subtree)





AVL Trees



AVL Tree

Structure of AVL tree nodes

```
typedef struct AVLTreeNode
{
    int bf;  // balance factor of node
    struct BinaryTreeNode *left;   // Left child
    int data; // The data in the node
    struct BinaryTreeNode *right;    // Right child
} AVLnode;
```

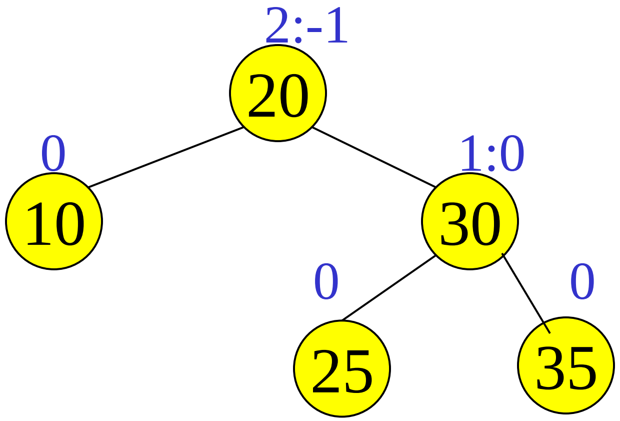
AVL Tree

An update (insert or delete) in an AVL tree could destroy the balance. It must then be rebalanced by transform the tree to restore the AVL tree property before the operation can be considered complete.

Balance is restored by tree rotations.

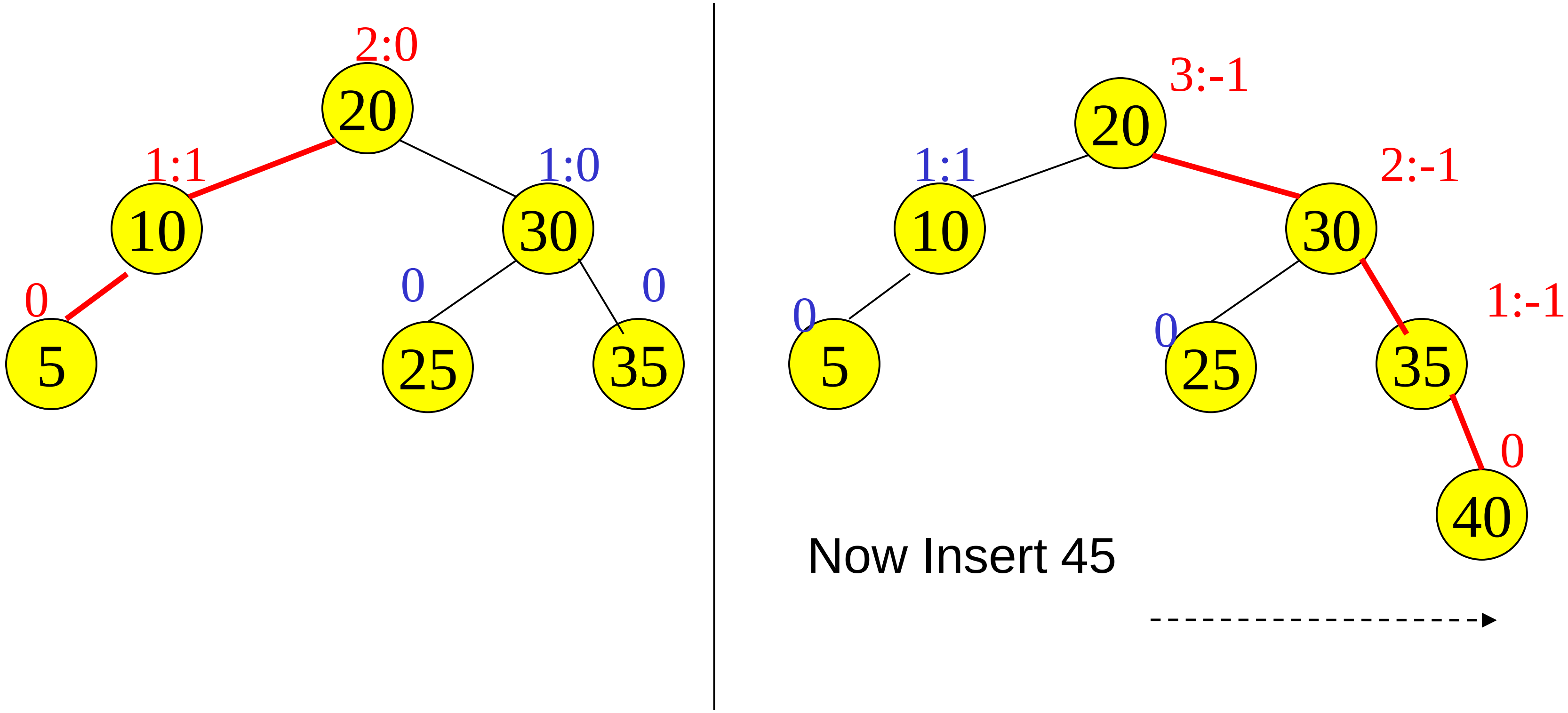
This is done using single rotations or double rotations.

Example of Insertions in an AVL Tree



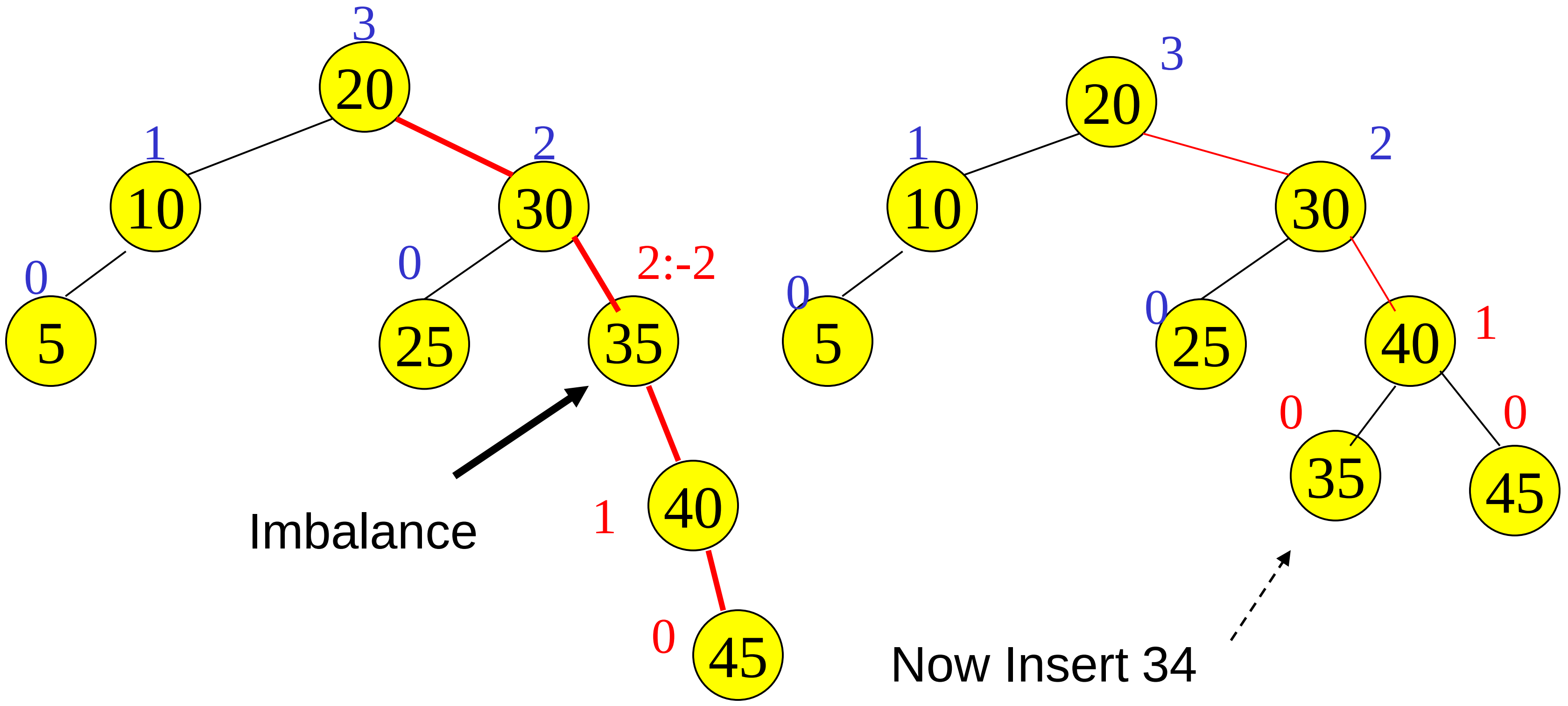
Insert 5, 40

Example of Insertions in an AVL Tree

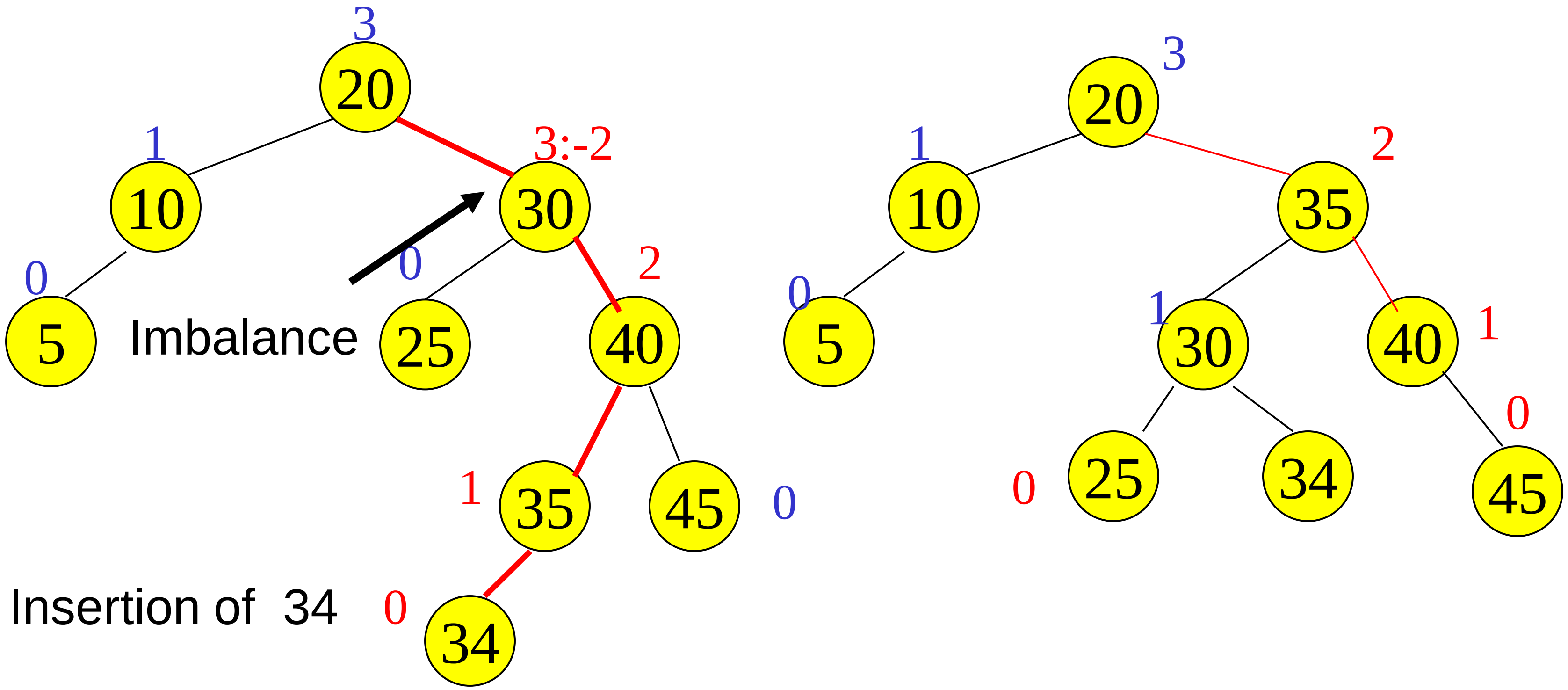


Now Insert 45

Single rotation



Double rotation



Deletion of a Node from AVL Tree

Delete a node as in binary search tree.
Deletion of a node from an AVL tree requires the same basic ideas of single and double rotations, that are used for insertion.

Operations (searching, insertion, deletion) on an AVL tree takes O(log n) time.