Algorithms

Lecture 2
Tree Data Structure (1)

A. S. M. Sanwar Hosen

Email: sanwar@wsu.ac.kr

Date: 16 March, 2023

Tree Data Structure (1)

- **Definition:** A tree is a hierarchical data structure with a set of connected nodes. Each node in the tree can be connected to many children but must be connected to exactly one parent except for the root node which has no parent.

✓ A tree can be defined as either:

(a) Empty (null), or

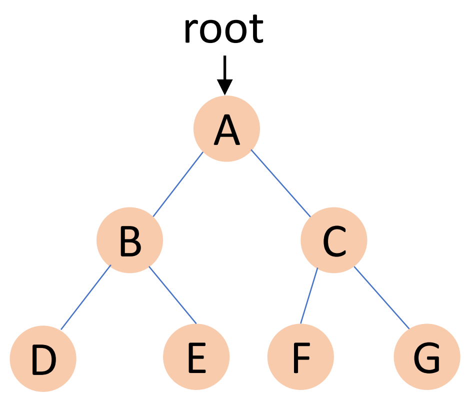
(b) A root node that contains:

1) data

2) a left subtree

3) a right subtree

(the left and right subtree could be empty)

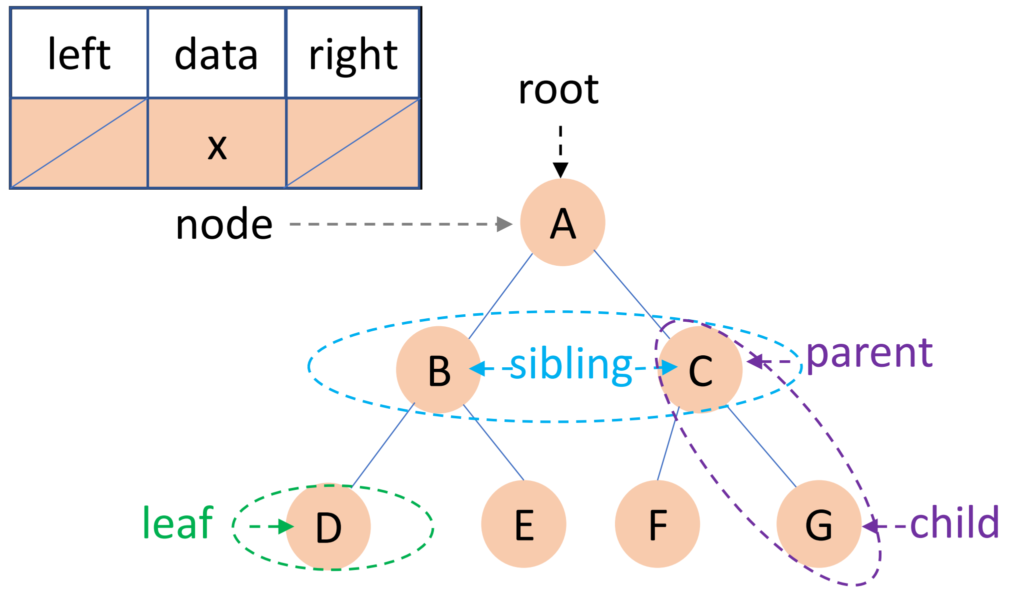


Tree Data Structure

Tree Data Structure (2)

❑ Properties (1)

- ✓ Node: an object containing a data value and left/right children
- ✓ Root: the topmost node of a tree
- ✓ Leaf: a node that has no children
- ✓ Branch: any internal node; neither the root nor a leaf
- ✓ Parent: the node which is predecessor of any node
- ✓ Child: the node which is descendent of any node
- ✓ Sibling: a node with a common parent

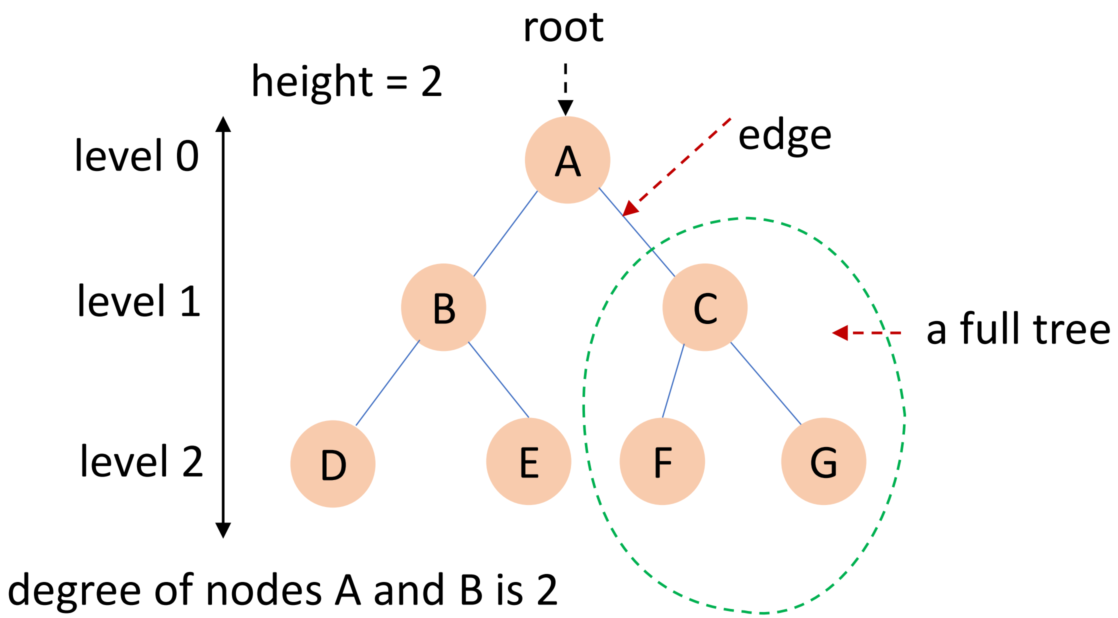


Tree Data Structure

Tree Data Structure (3)

❑ Properties (2)

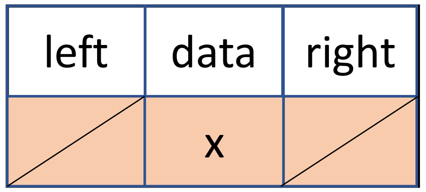
- ✓ Edge: the connecting link of any two nodes
- ✓ Degree: the total number of children of a node
- ✓ Subtree: the tree of nodes linked to the left/right from the current node
- ✓ Height: length of the longest path from the root to any node
- ✓ Level or Depth: length of the path from a root to a given node
- ✓ Full Tree: one where every branch has 2 children



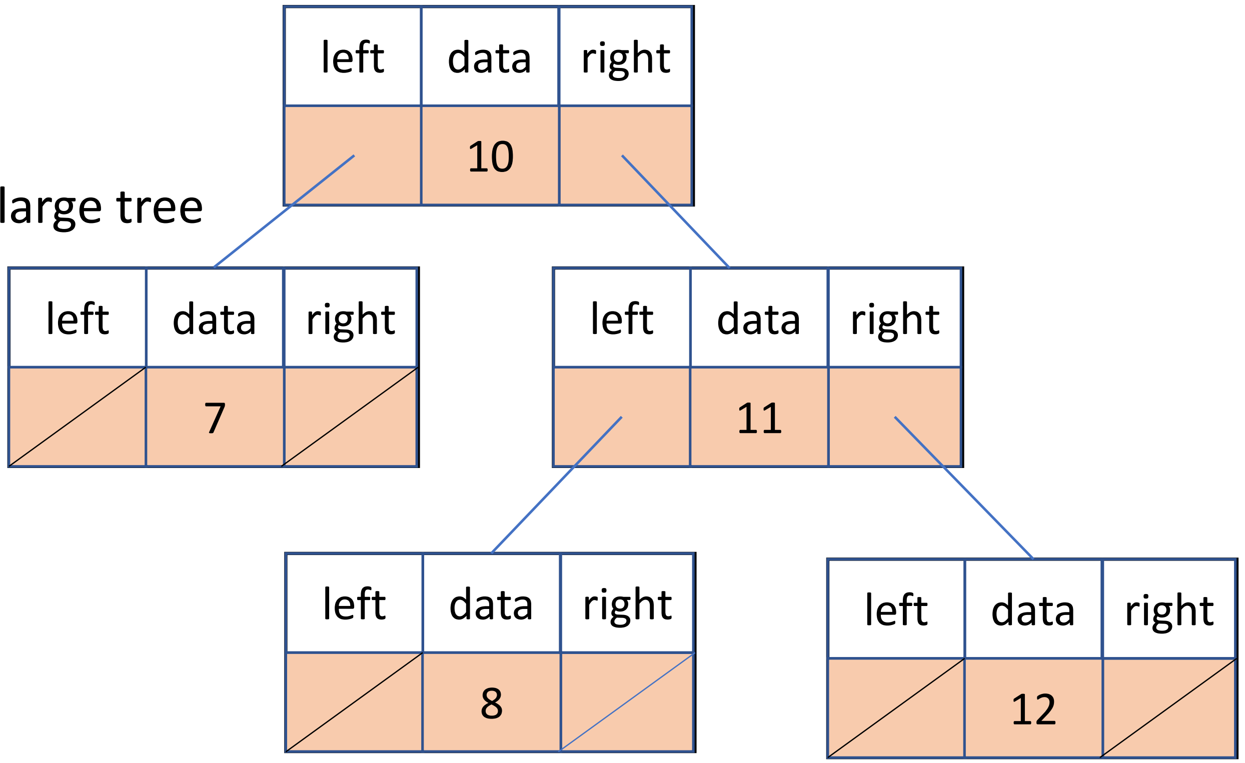
Tree Data Structure

Tree Data Structure (4)

- A basic tree node object stores data and refers to left/right



- Multiple nodes can be linked together into a large tree



Tree Data Structure (5)

- ❑ **Basic Functions**
- ✓ Create a node
- ✓ Insert values
- ✓ Display the tree
- ✓ Search a node/value
- ✓ Delete an element
- ✓ Finding height

Tree Data Structure (6)

❑ **Types of Trees**

✓ General Tree

✓ Binary Tree

✓ Binary Search Tree

✓ B-Tree

✓ AVL (Adelson-Velskii and Landis) Tree

Binary Tree (1)

- **Definition:** In binary tree, every parent node has at most 2 children. Each node can have either 0, 1 or 2 children. Typically, the 2 children are called left child and the right child.
- **Properties of a Binary Tree**

✓ A binary tree is either an empty tree or consists of a node called the root node, a left subtree, a right subtree, the subtree will also act as a binary tree once

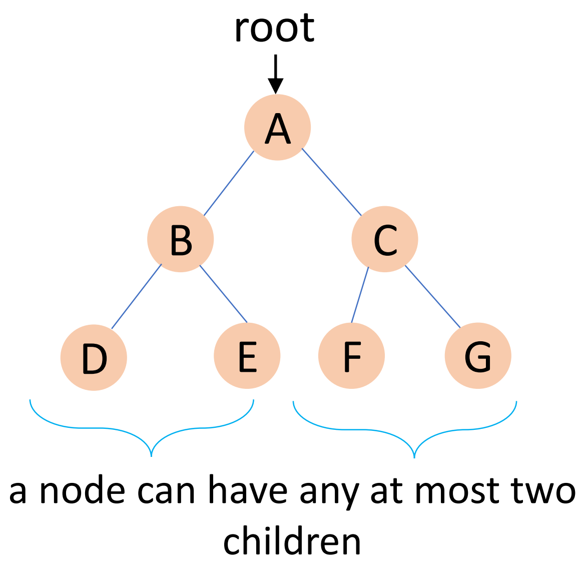
✓ The topmost node is called the root

✓ A node without children is called a leaf node or terminal node

✓ The maximum number of nodes at each level of i is 2^i

✓ Height of the tree is the longest path from the root node to the leaf node

✓ Depth of a node is the length of the path to its root



Binary Tree

Binary Tree (2)

- **Types of Binary Trees**

- ✓ Perfect Binary Tree: every internal node has two child nodes. All the leaf nodes are at the same level
- ✓ Full Binary Tree: every parent node or an internal node has either exactly two children or no child nodes
- ✓ Complete Binary Tree: all levels except the last one are full of nodes
- ✓ Degenerate Binary Tree: all the internal nodes have only one child
- ✓ Balanced Binary Tree: the left and right trees differ by either 0 or 1

Binary Search Tree

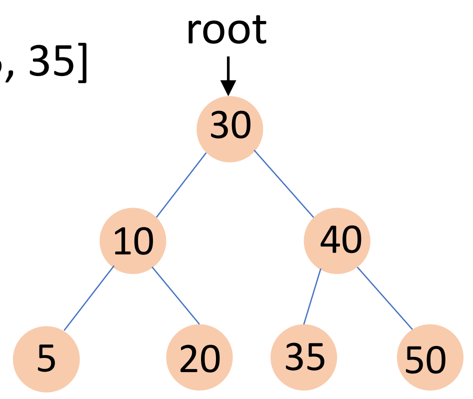
- ❑ **Definition:** A binary search tree (BST) is a type of tree that is sorted of a binary tree data structure.
- ❑ **Properties of a BST**

✓ The value of the left child should be less than to the parent node value

✓ The value of right child should be greater than to the parent node value

Example:

Keys/values: [30, 40, 10, 50, 20, 5, 35]



left node value < root node value < right node value

Binary Search Tree (BST)

B-Tree (1)

- **Definition:** B-Tree is a self–balancing search tree in which each node can contain more than one key and can have more than two children.
- **Properties of a B-Tree**

✓ Every node can have more than one value and child nodes

✓ Height balanced m-way tree

-- every node has m children

-- minimum children for

leaf node = 0,

root node = 2,

internal node = $\left\lceil \frac{m}{2} \right\rceil$

✓ All the leaf nodes must be at the same level

✓ Every node has maximum (m-1) keys

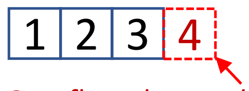
✓ Minimum keys: root node = 1 and all other nodes $\left\lceil \frac{m}{2} \right\rceil$ -1

✓ The left subtree of the node will have lesser values that the right side of the subtree

B-Tree (2)

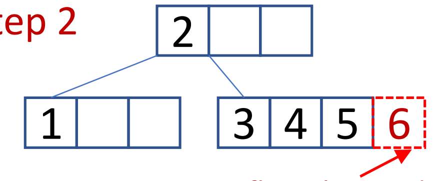
- **Example:** Construct a B-Tree of the following keys [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] of order $m = 4$.

Step 1



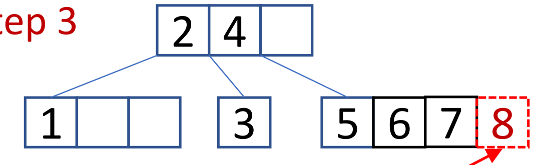
Overflow the condition,
max keys $m - 1 = 3$, choose a new root using median of the elements 1, 2, 3, 4 of left order is 2

Step 2



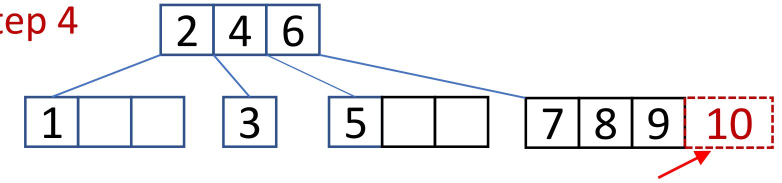
Overflow the condition,
max keys $m - 1 = 3$, shift the median of the elements 3, 4, 5, 6, of left order is 4, to the empty block of the upper node (if satisfy)

Step 3



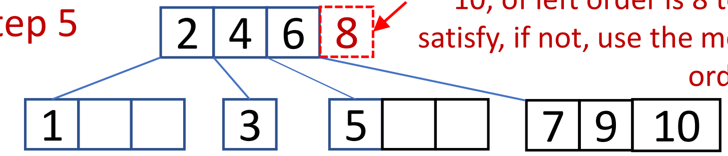
Overflow the condition,
max keys $m - 1 = 3$, shift the median of the elements 5, 6, 7, 8, of left order is 6 to the empty block of the upper node (if satisfy)

Step 4



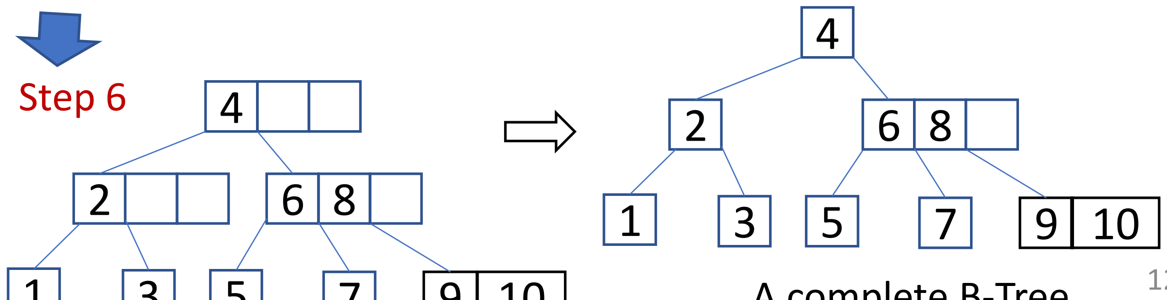
Overflow the condition,
max keys $m - 1 = 3$, shift the median of the elements 7, 8, 9, 10, of left order is 8 to the empty block of the upper node (if satisfy)

Step 5



Overflow the condition,
max keys = $m - 1 = 3$, shift the median of the elements 7, 8, 9, 10, of left order is 8 to the empty block of the upper node (if satisfy, if not, use the median of the elements 2, 4, 6, 8, of the left order is 4 as a new root)

Step 6



A complete B-Tree

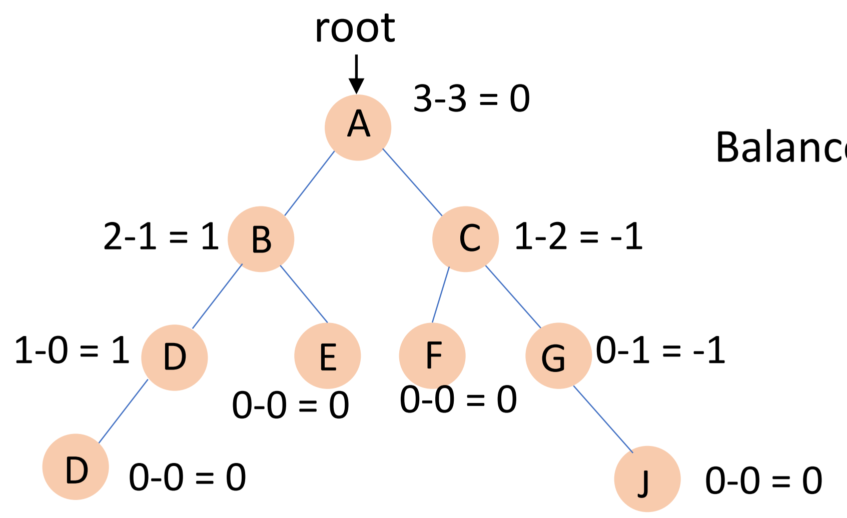
AVL Tree (1)

- **Definition:** AVL (Adelson-Velskii and Landis) tree is self-balancing binary search tree.
- **Properties of AVL Tree**

✓ Self-balancing tree

✓ Each node stores a value called a balanced factor, which is difference in the height of the left sub-tree and right sub-tree

✓ All nodes in the AVL tree must have a balance factor of –1, 0, and 1



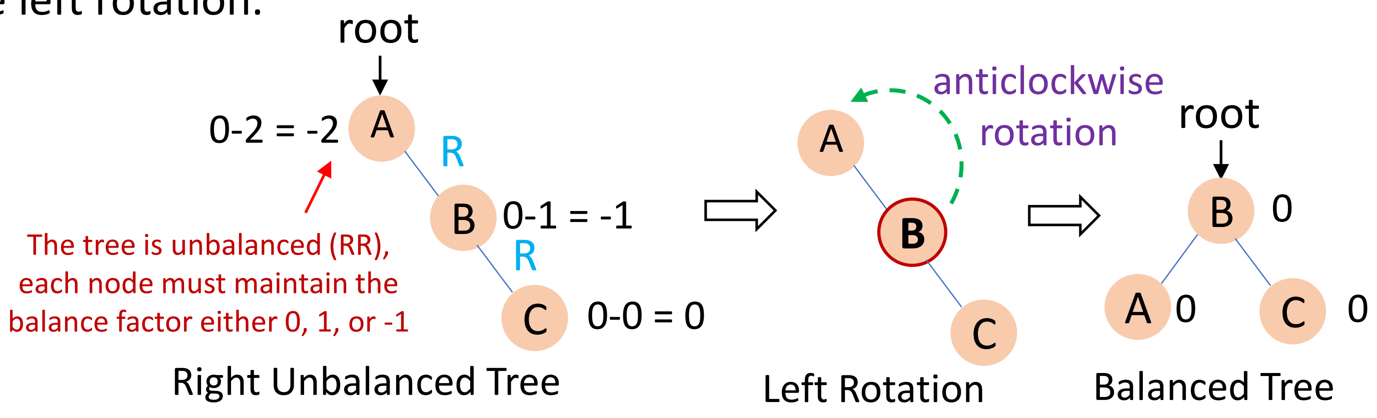
AVL Tree

Balance factor = |*left height – right height*|

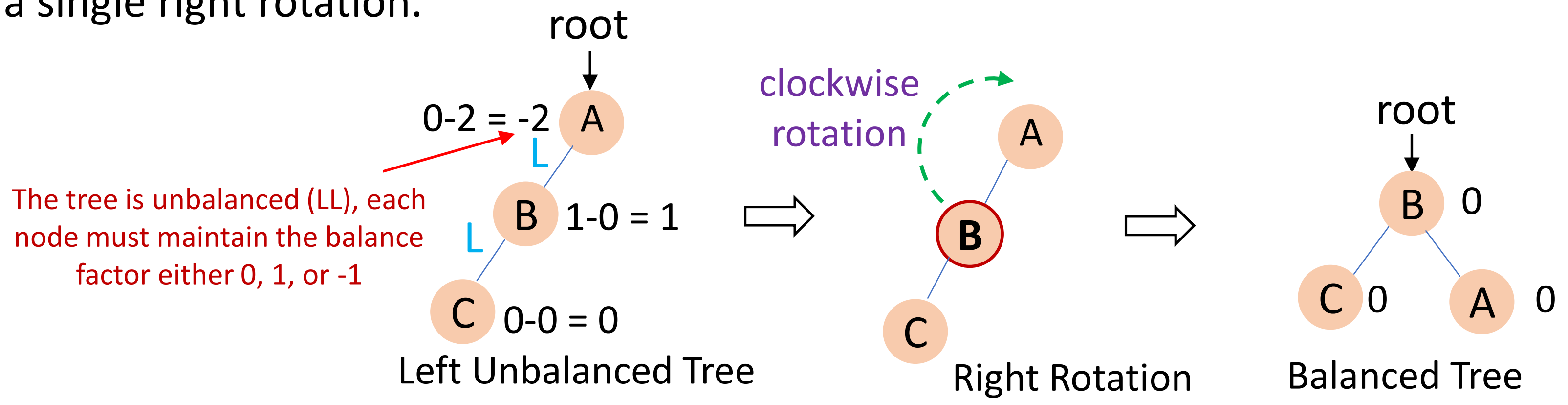
AVL Tree (2)

- **AVL Rotation:** To balance a tree itself, an AVL tree may performs the following rotations:

✓ **Left Rotation:** if a tree becomes unbalanced, when a node is inserted into the right subtree, then it requires perform a single left rotation.

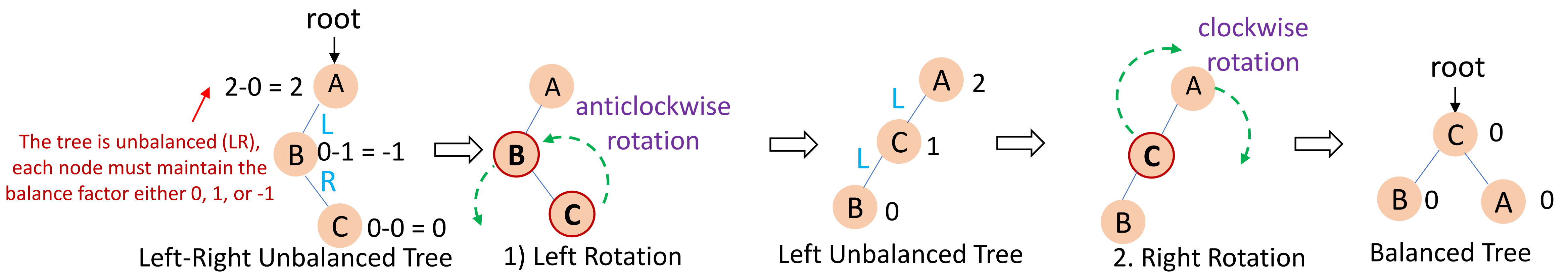


✓ **Right Rotation:** if a tree becomes unbalanced, when a node is inserted into the left subtree, then it requires perform a single right rotation.

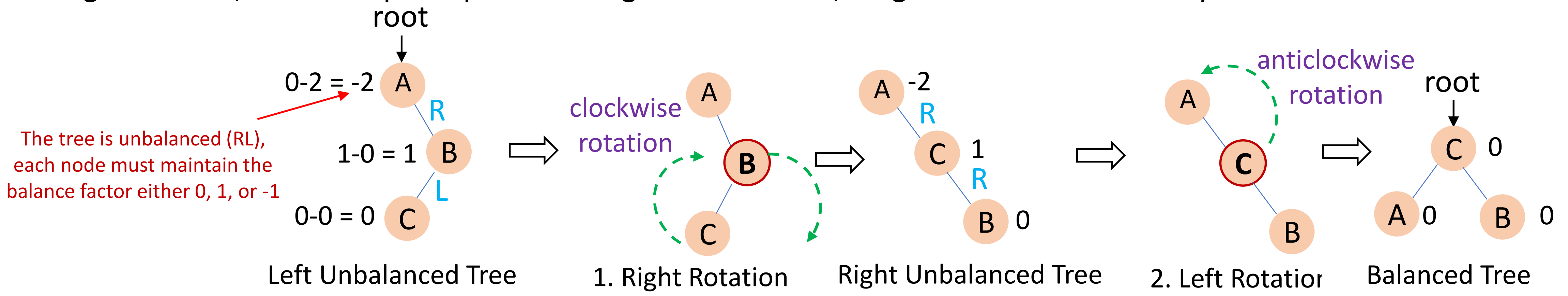


AVL Tree (3)

- ✓ **Left-Right Rotation:** if a tree becomes unbalanced, when a node is inserted into the right subtree of the left subtree, then it requires perform a left-right rotation, a left rotation followed by right rotation.

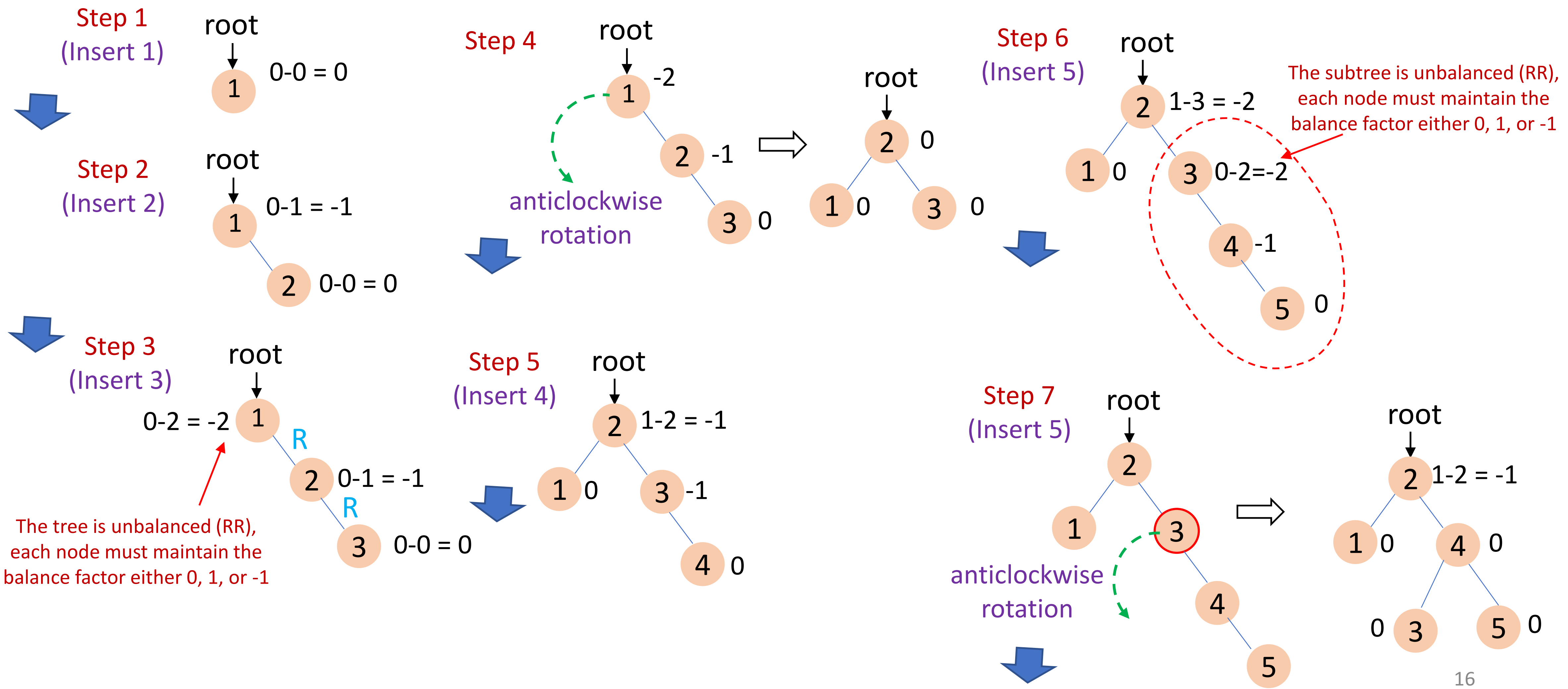


- ✓ **Right-Left Rotation:** if a tree becomes unbalanced, when a node is inserted into the left subtree of the right subtree, then it requires perform a right-left rotation, a right rotation followed by left rotation.



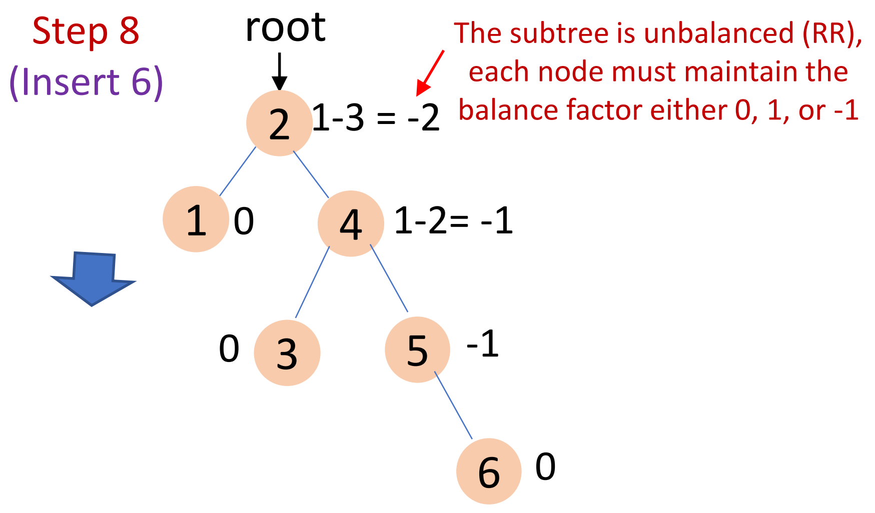
AVL Tree (4)

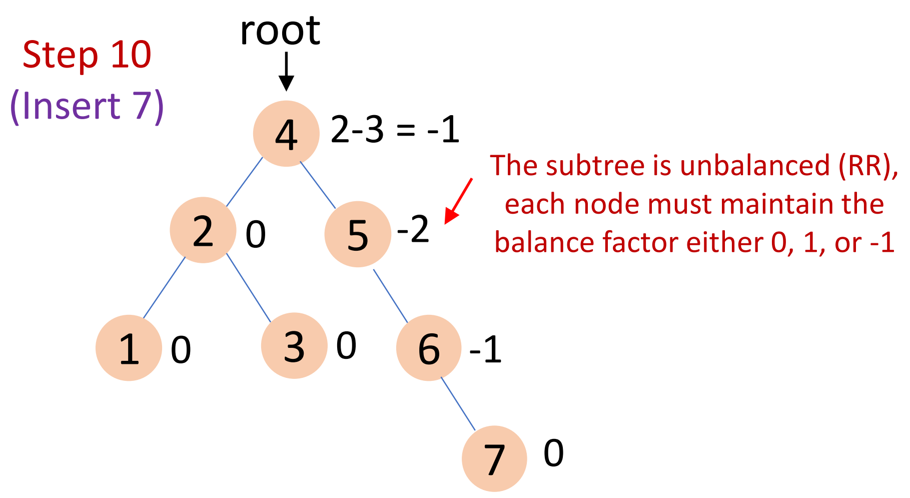
- **Example:** Construct an AVL tree for the following elements [1, 2, 3, 4, 5, 6, 7, 8]

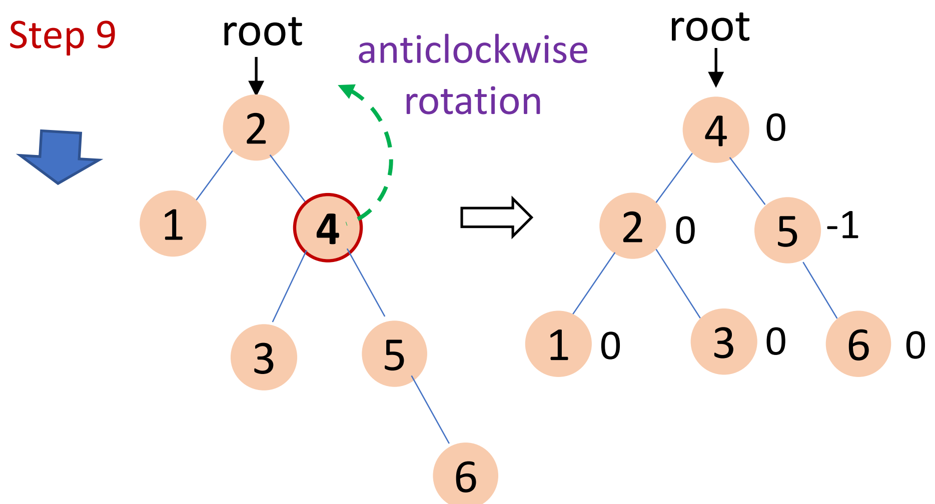


AVL Tree (5)

- **Example:** Construct an AVL tree for the following elements [1, 2, 3, 4, 5, 6, 7, 8] (continue)

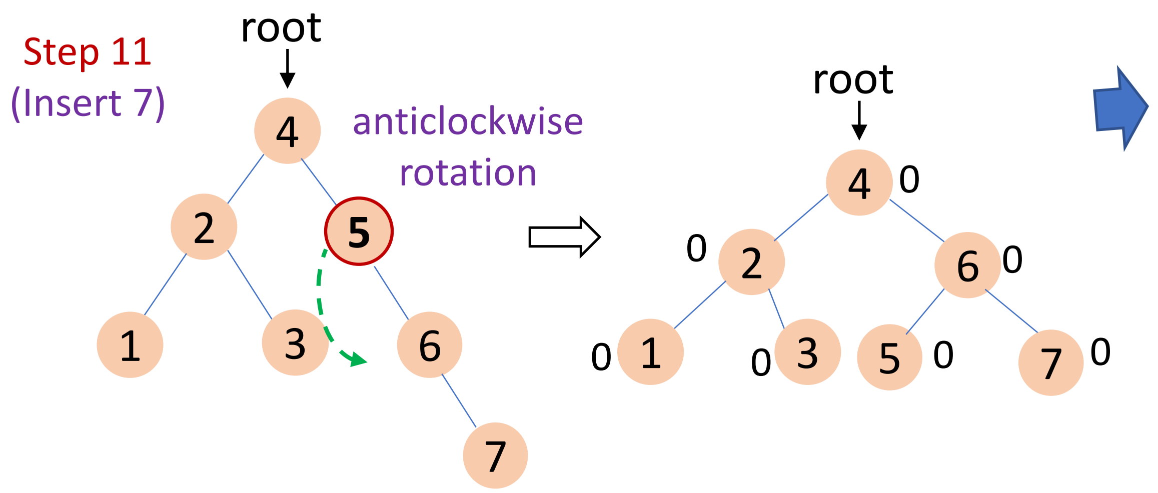

Step 8
(Insert 6)
root
The subtree is unbalanced (RR), each node must maintain the balance factor either 0, 1, or -1
2 1-3 = -2
1 0
4 1-2= -1
0 3
5 -1
6 0

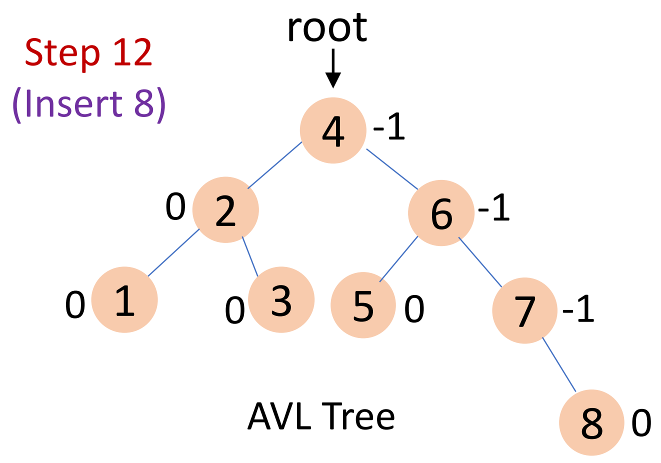

Step 10
(Insert 7)
root
4 2-3 = -1
The subtree is unbalanced (RR), each node must maintain the balance factor either 0, 1, or -1
2 0
5 -2
1 0
3 0
6 -1
7 0



AVL Tree (6)

❑ **Example:** Construct an AVL tree for the following elements [1, 2, 3, 4, 5, 6, 7, 8] (continue)





Lecture Review

- This Lecture Discussed About:
- ✓ Tree Data Structure
  - Properties of Tree
- ✓ Types of Tress in Data Structure
  - Binary Tree,
  - Binary Search Tree
  - B-Tree
  - AVL Tree