TECHNO INDIA UNIVERSITY
WESTBENGAL

Data Structures & Algorithms
(PCC-CS 301)

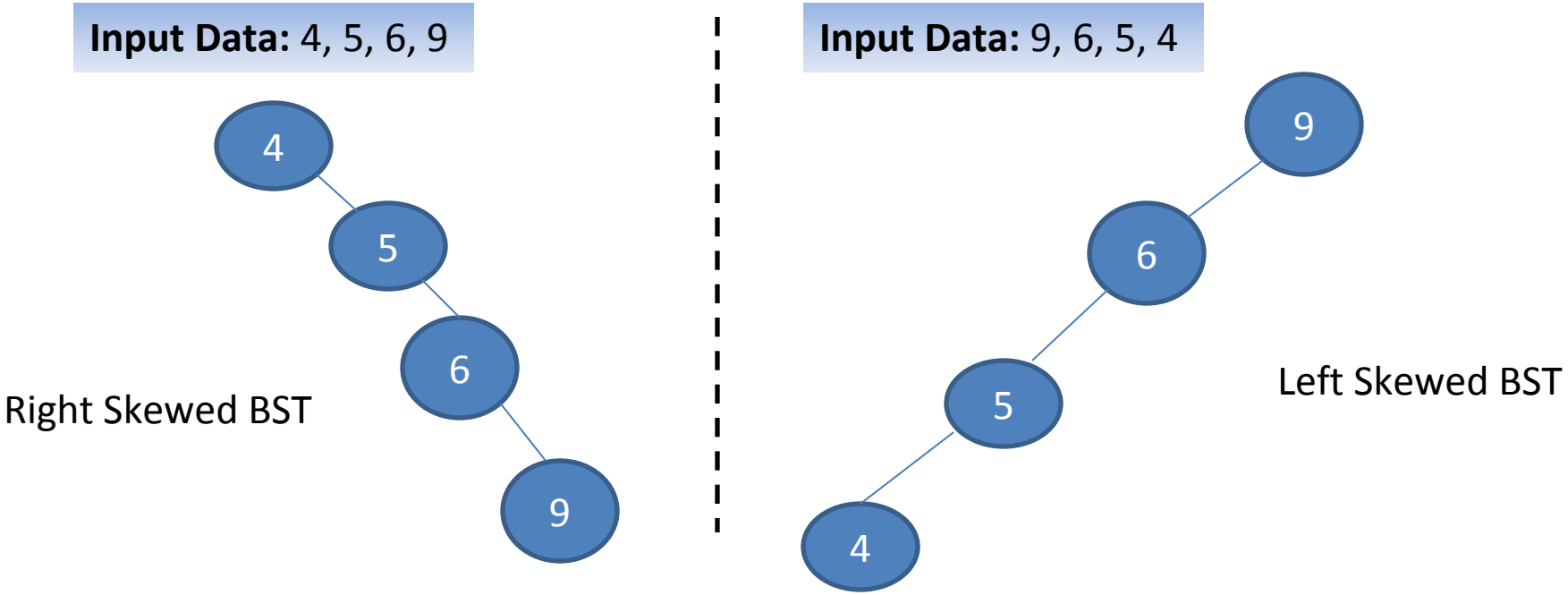
Dr. Debashis Das
Associate Professor
Department of CSE
Techno India University, Kolkata

Topics Covered

1. Disadvantage of BST
2. Introduction to Balanced Tree
3. AVL tree: data insertion

Binary Search Tree

- Disadvantage of BST
  - Higher searching time on few BST



Searching time of leaf requires **O(n)**

Balanced Tree

- Introduction to height balanced tree

  - Can we reduce the searching time in worst case?
    - Yes, by maintaining the height of the BST
    - Height balanced BST can search an arbitrary element in $O(\log_2 n)$ time [in **worst-case**]

  - Example of height-balanced BST
    - AVL tree (invented in 1962, by **A**delson-**V**elskii and **L**andis)
    - Red-Black tree (invented in 1972, by Rudolf Bayer)

AVL Tree

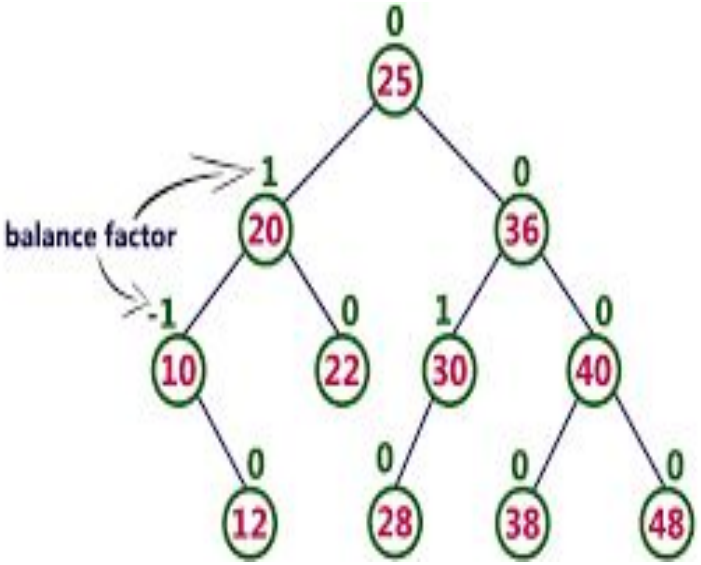
- Introduction
  - Definition

A BST is called as AVL tree iff for each node it maintains:

Balance Factor(BF)= $|h(TL) - h(TR)| \leq 1$ i.e. BF={-1,0,1}

h(TR) : height of right sub-tree

h(TL) : height of left sub-tree



- Property

Maintain a balanced height of the tree that is able to produce a search result in $\log_2(n)$ time

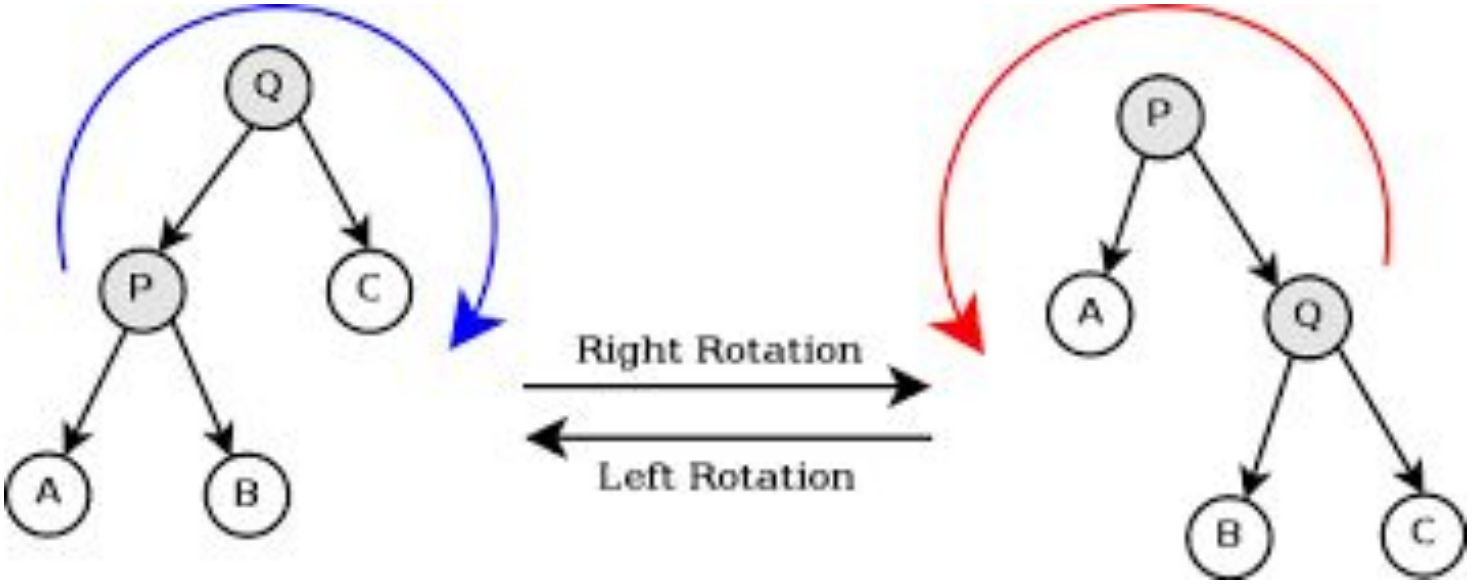
AVL Tree

- Data insertion on AVL tree
  - Data insertion in AVL tree is similar as BST
    - new data is compared with each node starting from the root and if found smaller , move to left sub-tree otherwise move to right sub-tree until reached to its proper place
  - After insertion, check the Balance Factor of each node
  - If any node holds BF out of range i.e BF < -1 or BF > 1
  - Trace the unbalanced node scanning from the bottom of the tree
  - Apply any of the following rotations on the unbalanced node, based on the node position for which it becomes unbalanced
    - LL rotation
    - RR rotation
    - LR rotation
    - RL rotation

AVL Tree

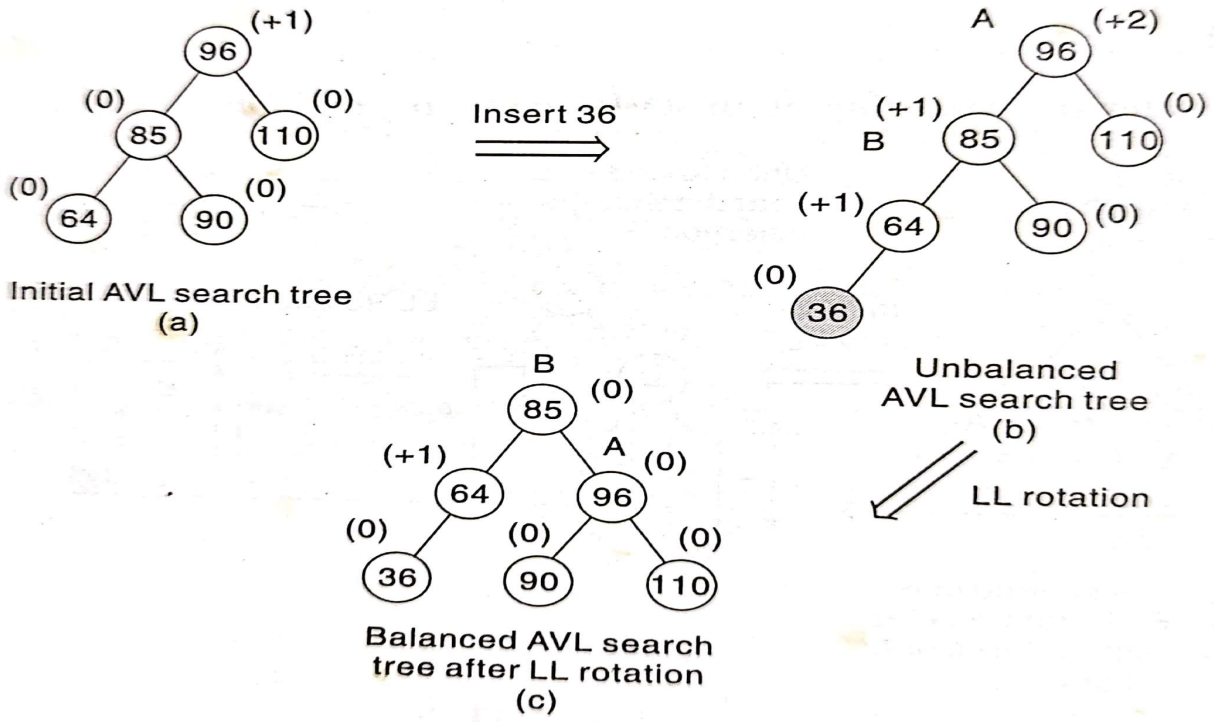
- Data insertion on AVL tree

  - Basic rotation concept
    - How the nodes will be changing in rotation



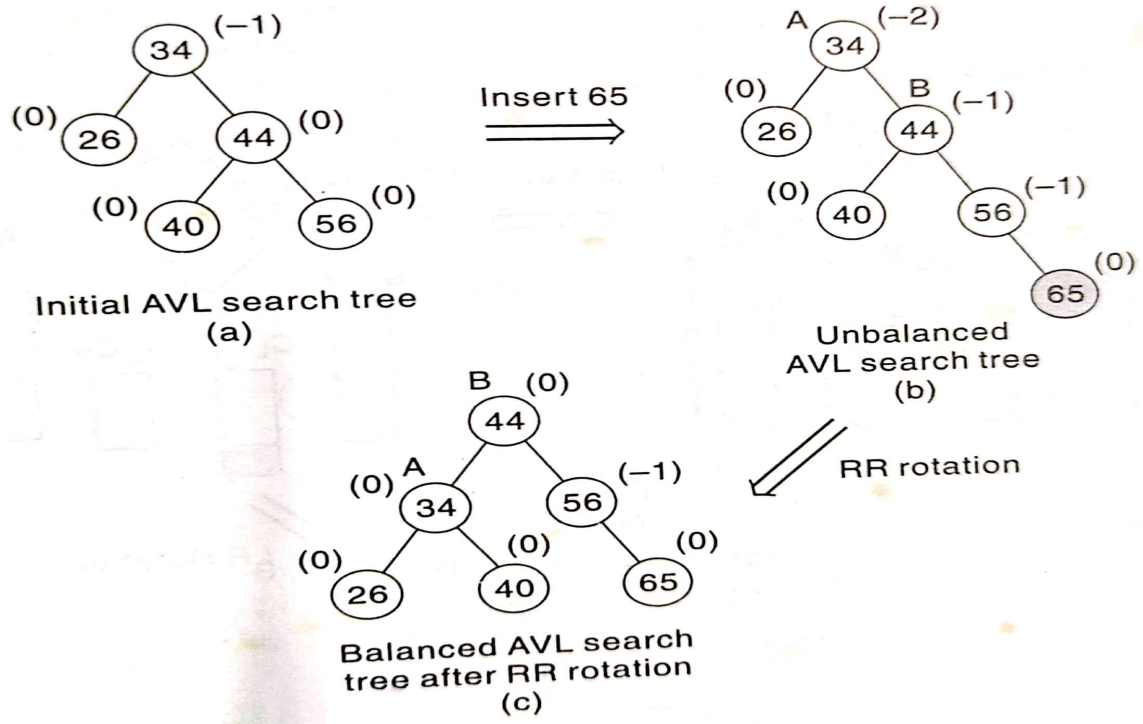
AVL Tree

- Data insertion
  - **LL rotation:** if inserted node is in the left to left sub-tree of unbalanced node (it is a **clock-wise rotation**)



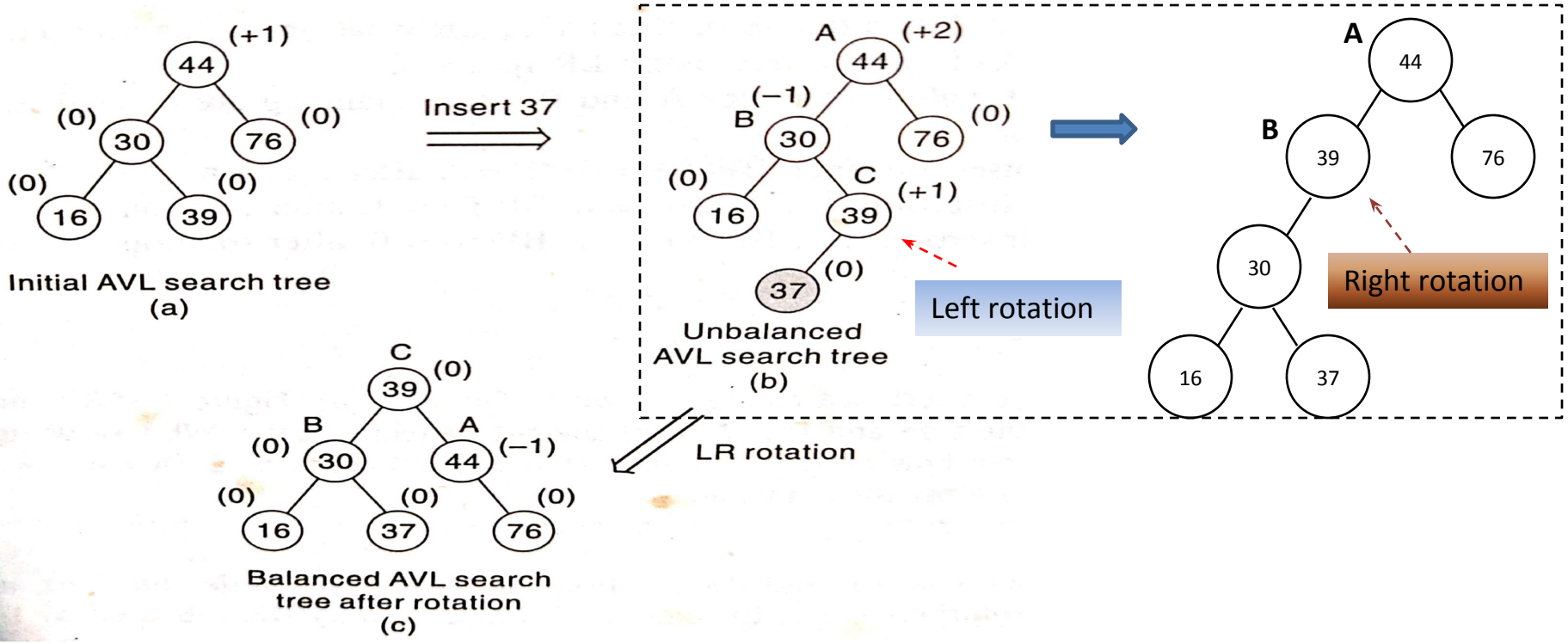
AVL Tree

- Data insertion
  - **RR rotation:** if inserted node is in the right to right sub-tree of unbalanced node (**anti-clock-wise rotation**)



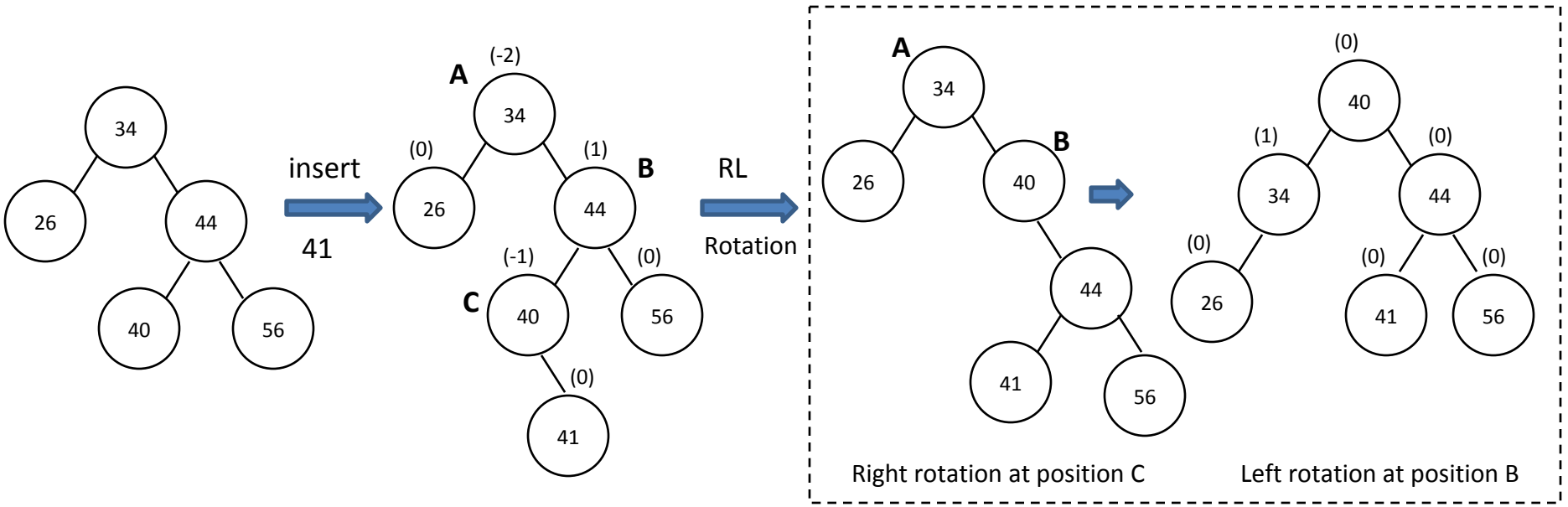
AVL Tree

- Data insertion
  - **LR rotation:** if inserted node is in the right to left sub-tree of unbalanced node (**double rotation: left rotation -> right rotation**)



AVL Tree

- Data insertion
  - **RL rotation:** if inserted node is in the left to right sub-tree of unbalanced node (**double rotation: right rotation -> left rotation**)



Queries?

Problem

Q1. Consider the following data that need to be inserted in an AVL tree

Data set: { 25, 40, 15, 10, 6, 28, 32, 35, 50, 5, 12, 8, 10, 38, 3, 45 }