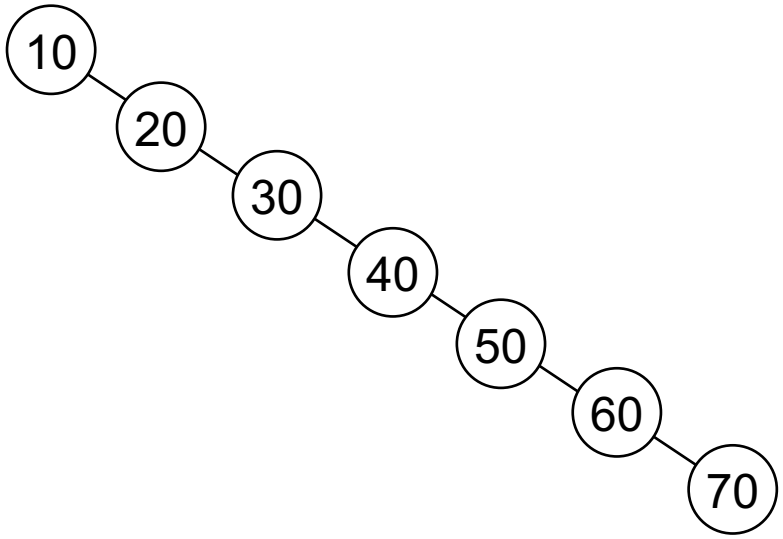
DS:
AVL Tree

Liwei

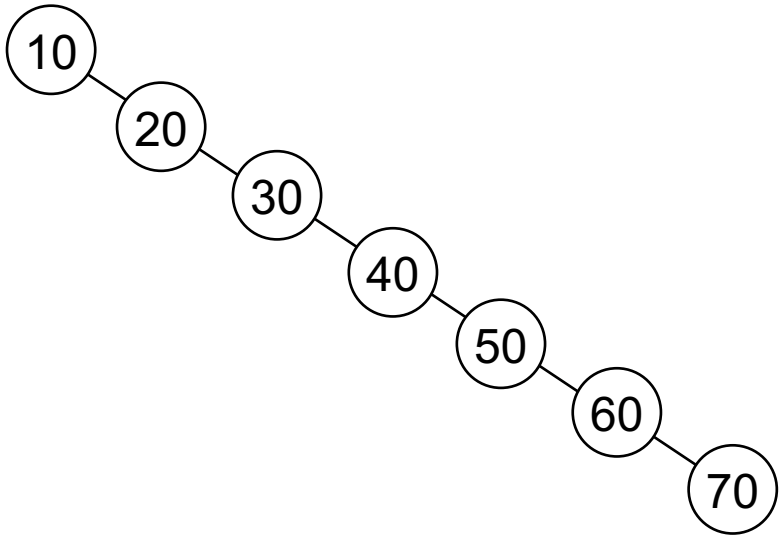
Why AVL tree?

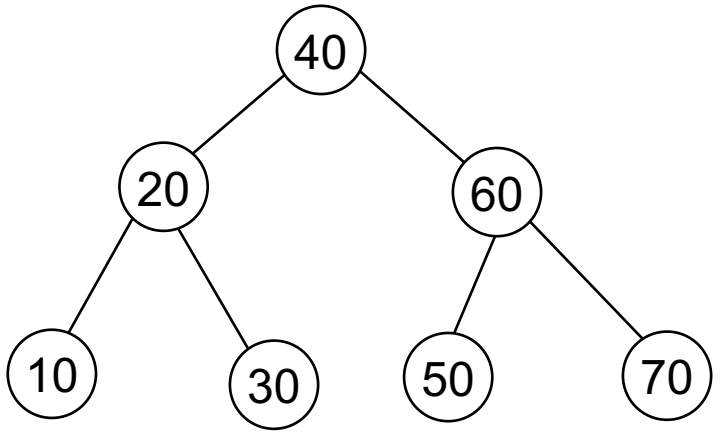
- Quesiton
  - Insert 10, 20, 30, 40, 50, 60, 70 in BST



Why AVL tree?

- Quesiton
  - Insert 10, 20, 30, 40, 50, 60, 70 in BST



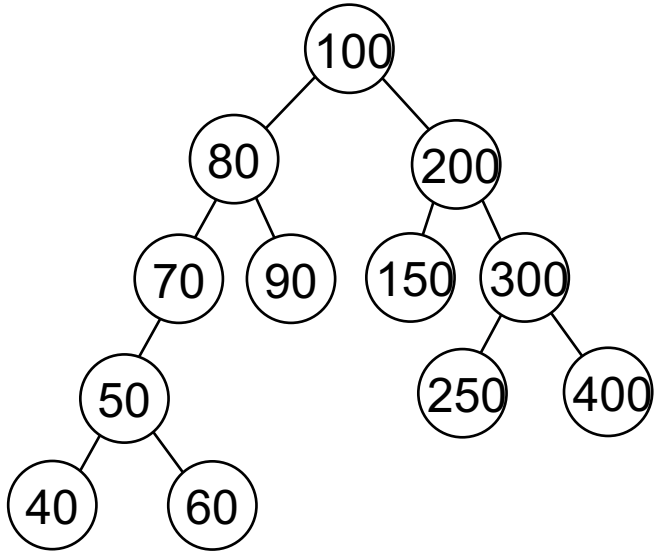


Why AVL tree?

- Depending on Incoming data, a Binary Search Tree can get skewed and hence its performance starts going down.
- Instead of O(logn) for insertion/search/deletion, it can go up to O(n)
- AVL tree attempts to solve this problem of “skewing” by introducing a concept called “Rotation”.

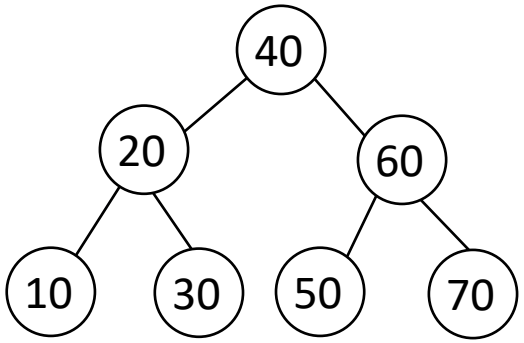
What is AVL tree?

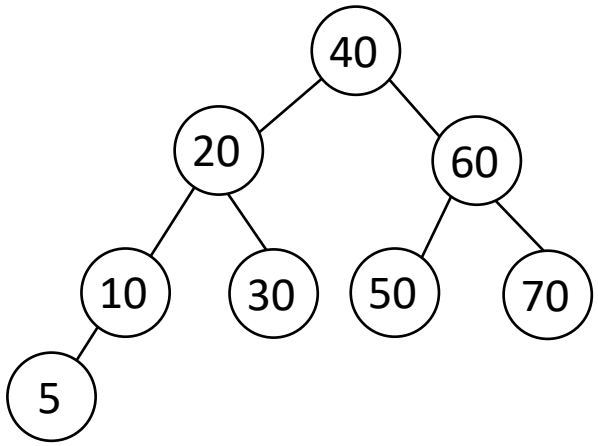
- An AVL tree is a balanced binary search tree where the height of immediate subtrees of any node differs at most one (also called balance factor)
- If at any time heights differ by more than one, rebalancing is done to restore this property (called rotation)
- Empty height is always considered -1

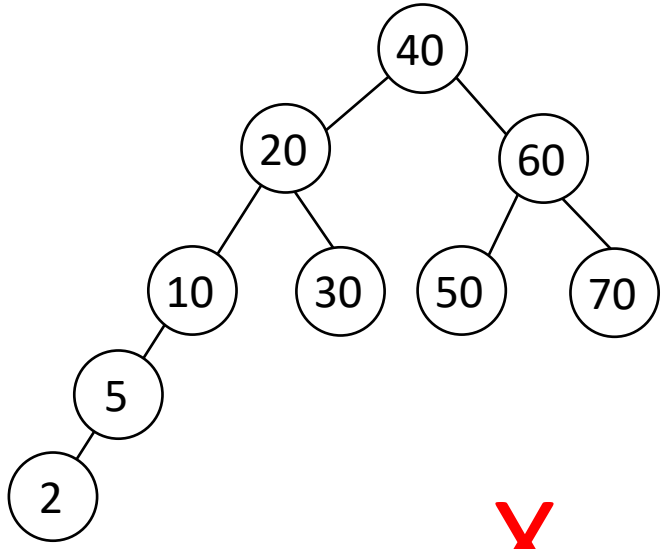


- Regular BST insert
- Track the height
- Determine balance factor (difference in height can’t be > 1)
- 4 cases

Examples of AVL tree







X

Common operations of AVL tree

- Create a AVL tree
- Search a node
- Traversal all nodes
- Insert a node
- Delete a node
- Delete the AVL tree

Insertion of node in AVL Tree

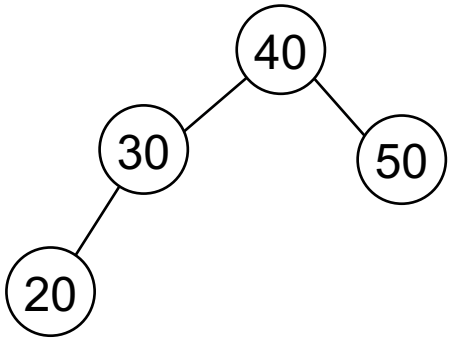
- Case 1: when rotation is not required
- Case 2: when rotation is requried (LL, LR, RR, RL)

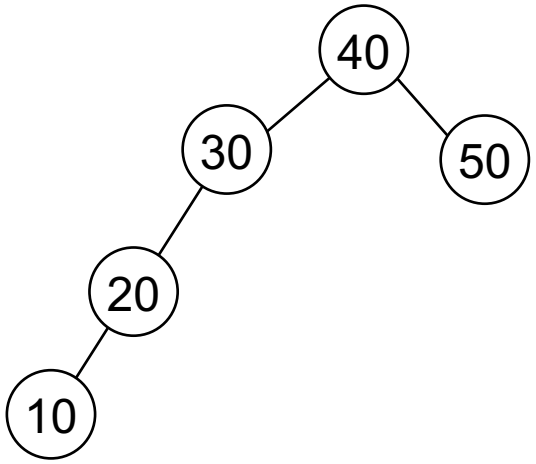
Rotation Conditions

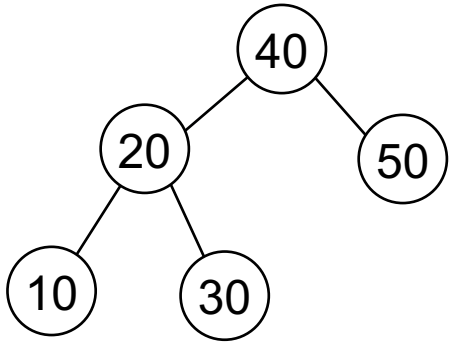
- Left Left Condition (LL)
- Left Right Condition (LR)
- Right Right Condition (RR)
- Right Left Condition (RL)

Left-Left Condition

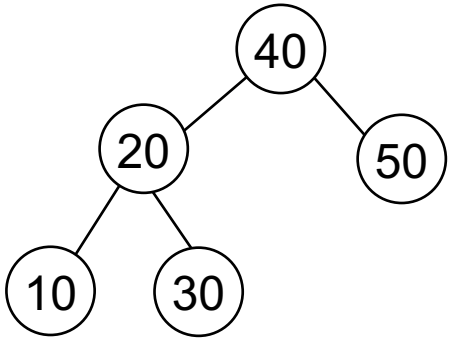
- What is Left-Left Condition?
  - Left-Left Node from currentNode is causing disbalance
  - In this case we do a Right Rotation

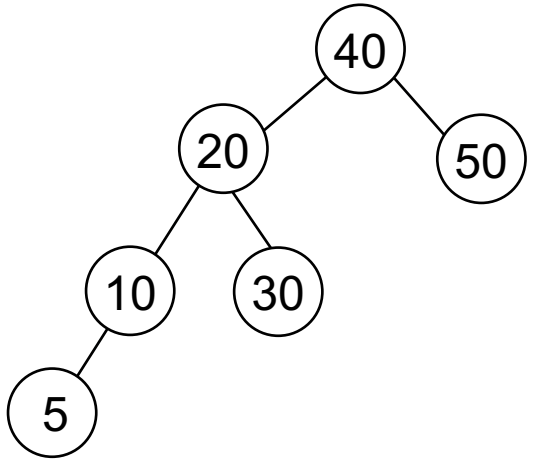


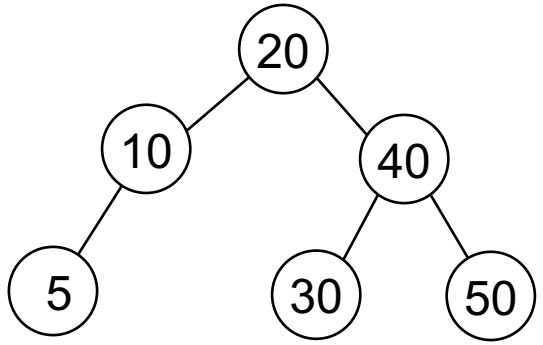




Left-Left Condition (Example 2)

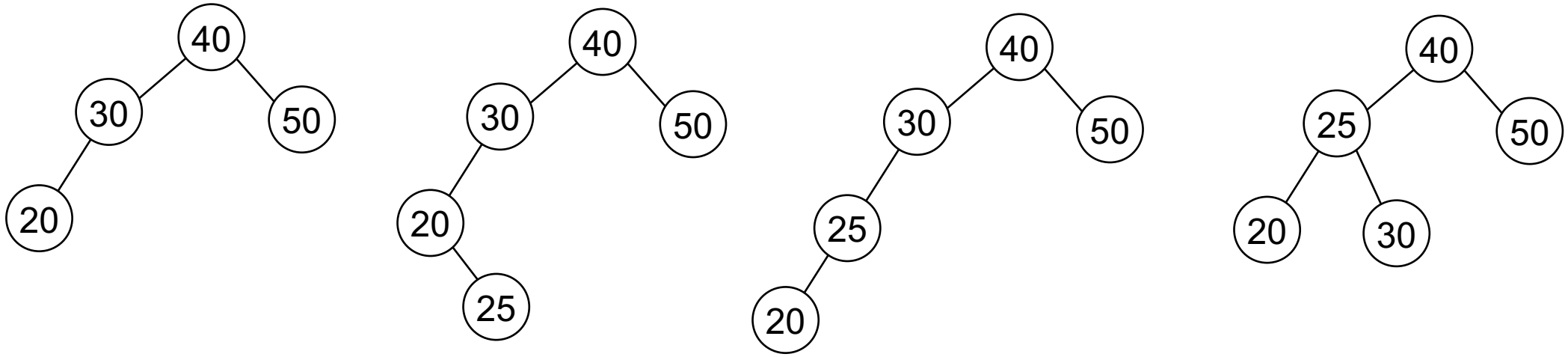






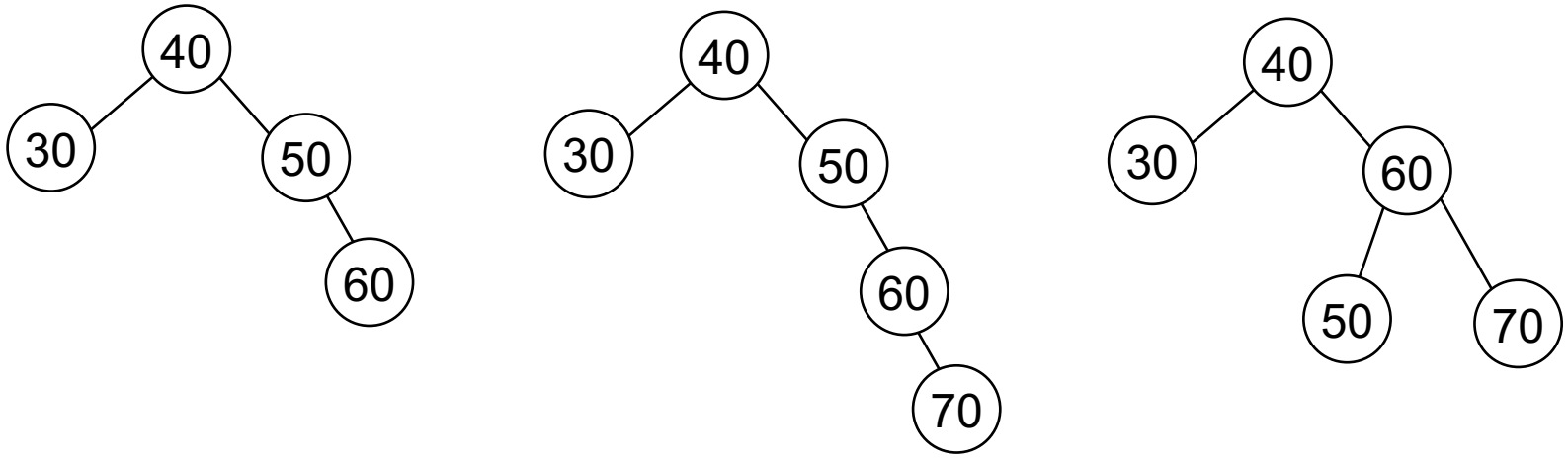
Left-Right Conditions

- What is Left Right Condition?
  - Left-Right Node from currentNode is causing disbalance
  - In this case we do a Left Rotation followed by Rigth Rotation

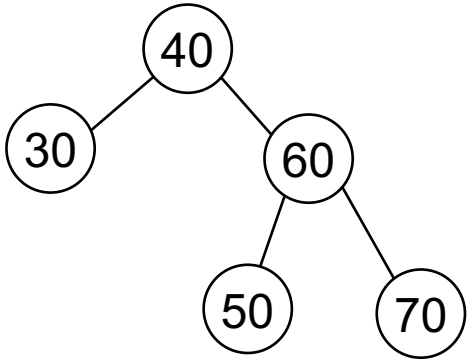


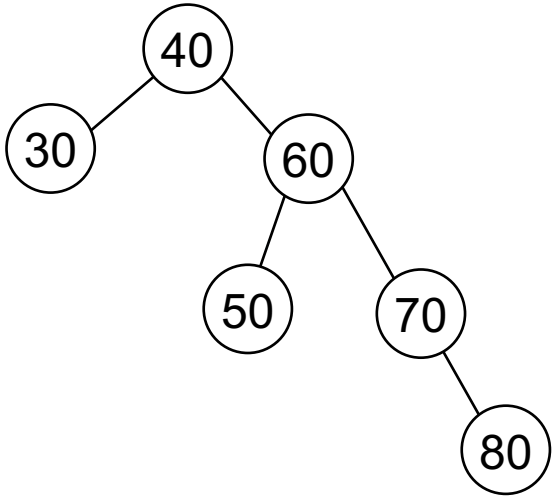
Right-Right Condition

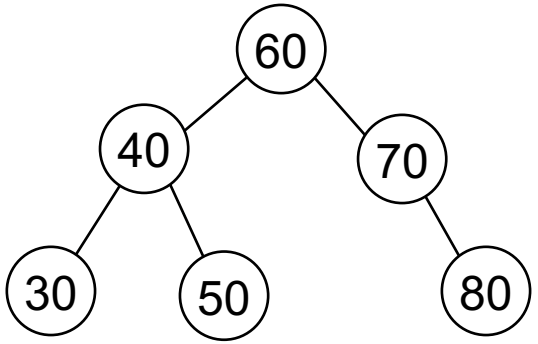
- What is Right-Right Condition?
  - Right-Right Node from currentNode is causing disbalance
  - In this case we do a Left Rotation



Right-Right Condition (Example 2)

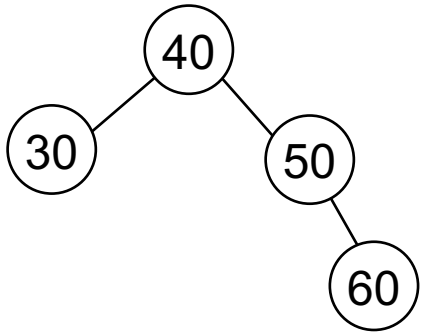


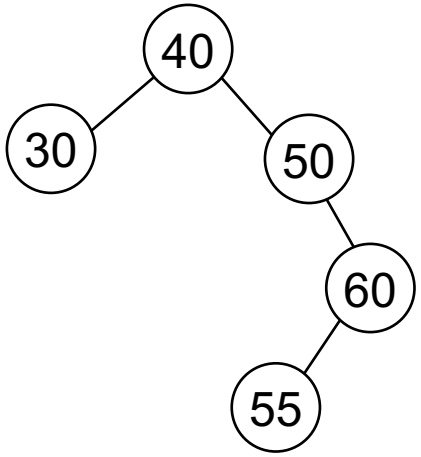


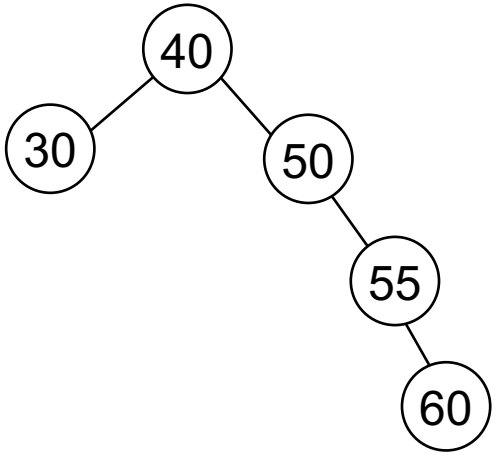


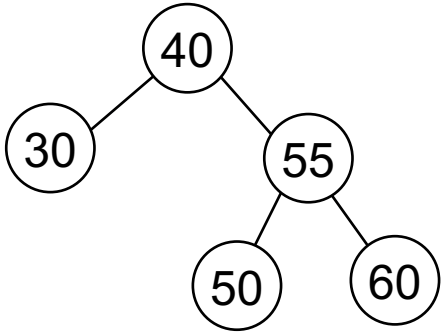
Right-Left Condition

- What is Right-Left Condition?
  - Right-Left Node from currentNode is causing disbalance
  - In this case we do a Right Rotation followed by Left Rotation









Insertion of node in AVL Tree

- Case 1: when rotation is not required
- Case 2: when rotation is requried (LL, LR, RR, RL)

Insertion in AVL Tree

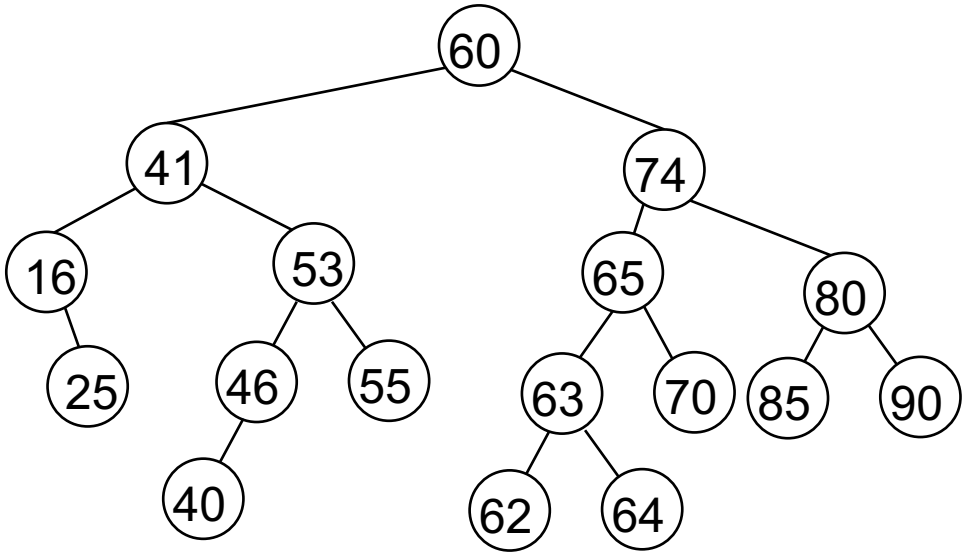
- Insert 30, 20, 40, 10, 5, 3, 4, 50, 60, 70, 65

Deletion of node from AVL Tree

- Case 1: when tree does not exist
- Case 2: when rotation is not requried (BST condition)
- Case 3: when rotation is required (LL, LR, RR , RL)

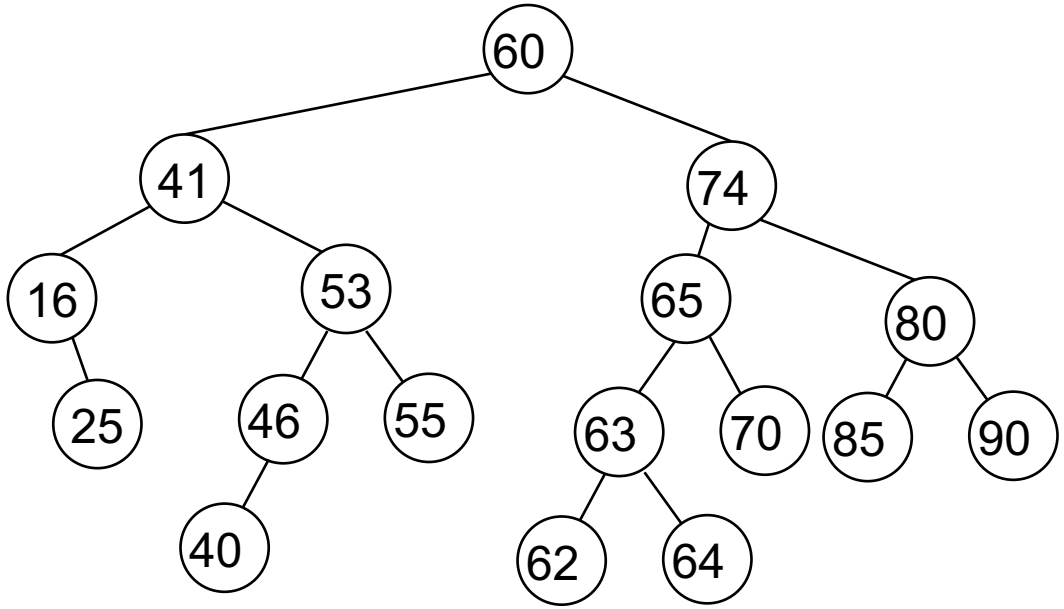
Deletion of node from AVL Tree: Case #2

- when rotation is not requried
  - Node to be deleted is leaf node; Just delete the node.
  - Node to be deleted has 1 child; Replace the node with the child.
  - Node to be deleted has 2 childred; Replace the node with the next successor.



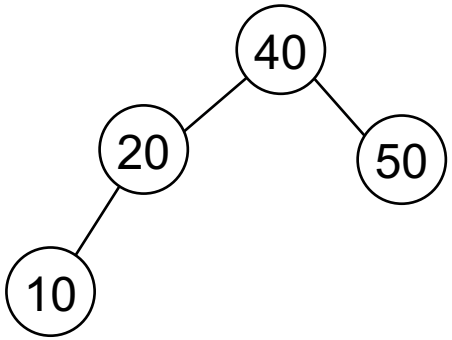
Deletion of node from AVL Tree: Case #3

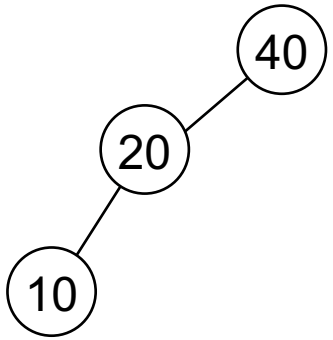
- when rotation is requried (LL, LR, RR, RL)

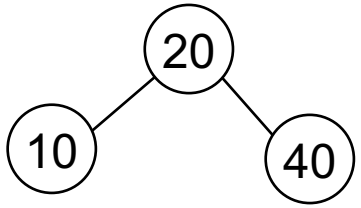


Left-Left Condition

- What is Left-Left Condition?
  - Left-Left Node from currentNode is causing disbalance
  - In this case we do a Right Rotation

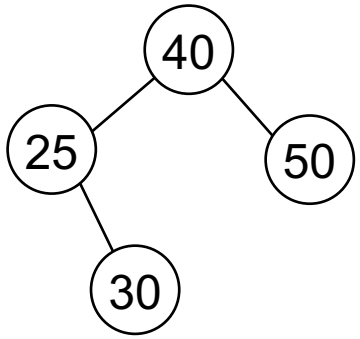


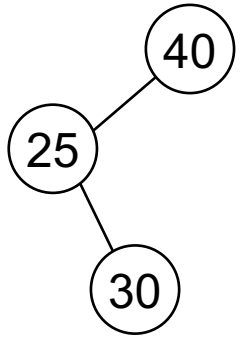


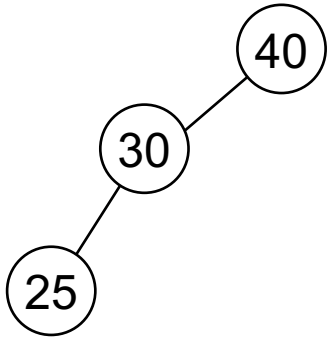


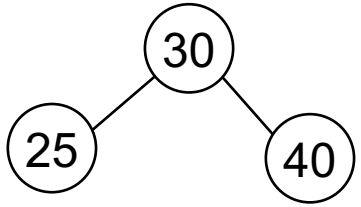
Left-Right Conditions

- What is Left Right Condition?
  - Left-Right Node from currentNode is causing disbalance
  - In this case we do a Left Rotation followed by Rigth Rotation



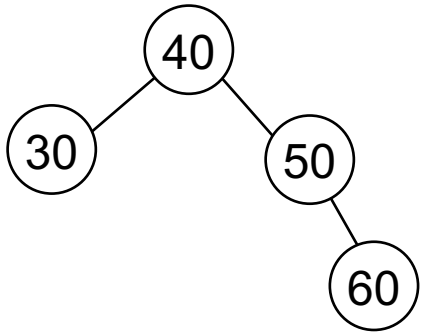


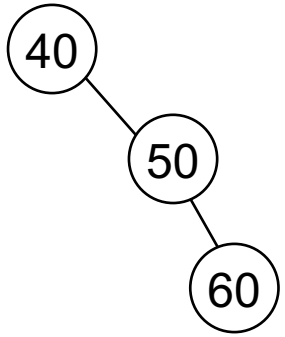


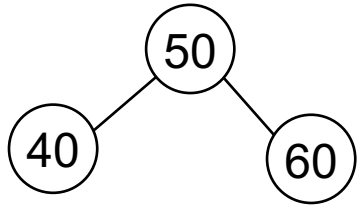


Right-Right Condition

- What is Right-Right Condition?
  - Right-Right Node from currentNode is causing disbalance
  - In this case we do a Left Rotation

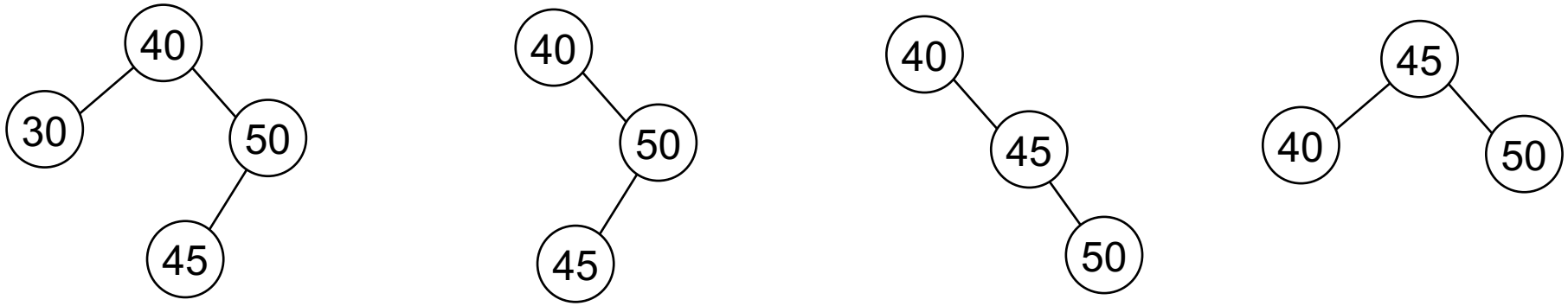




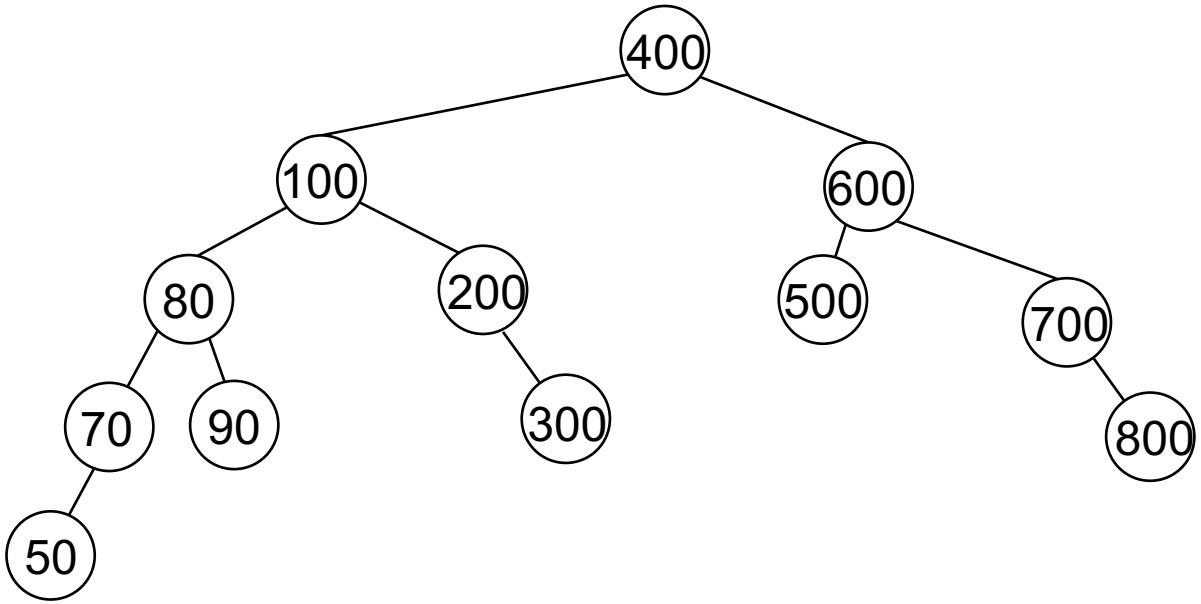


Right-Left Condition

- What is Right-Left Condition?
  - Right-Left Node from currentNode is causing disbalance
  - In this case we do a Right Rotation followed by Left Rotation



Delete Node from AVL Tree (example)



Delete: 90, 50 ,300, 200, 500
Insert 750
Delete 600

Time Complexity (AVL Tree)

	Time Complexity
Creation of AVL Tree	O(1)
Insertion of node	O(logn)
Deletion of node	O(logn)
Searching a value	O(logn)
Traversing entire AVL tree	O(n)
Deleting entire AVL tree	O(1)

Time Complexity (BST vs. AVL Tree)

	Worse Case Scenario	
	BST	AVL
Creation of Tree	O(1)	O(1)
Insertion of node	O(n)	O(logn)
Deletion of node	O(n)	O(logn)
Searching a value	O(n)	O(logn)
Traversing entire tree	O(n)	O(n)
Deleting entire L tree	O(1)	O(1)