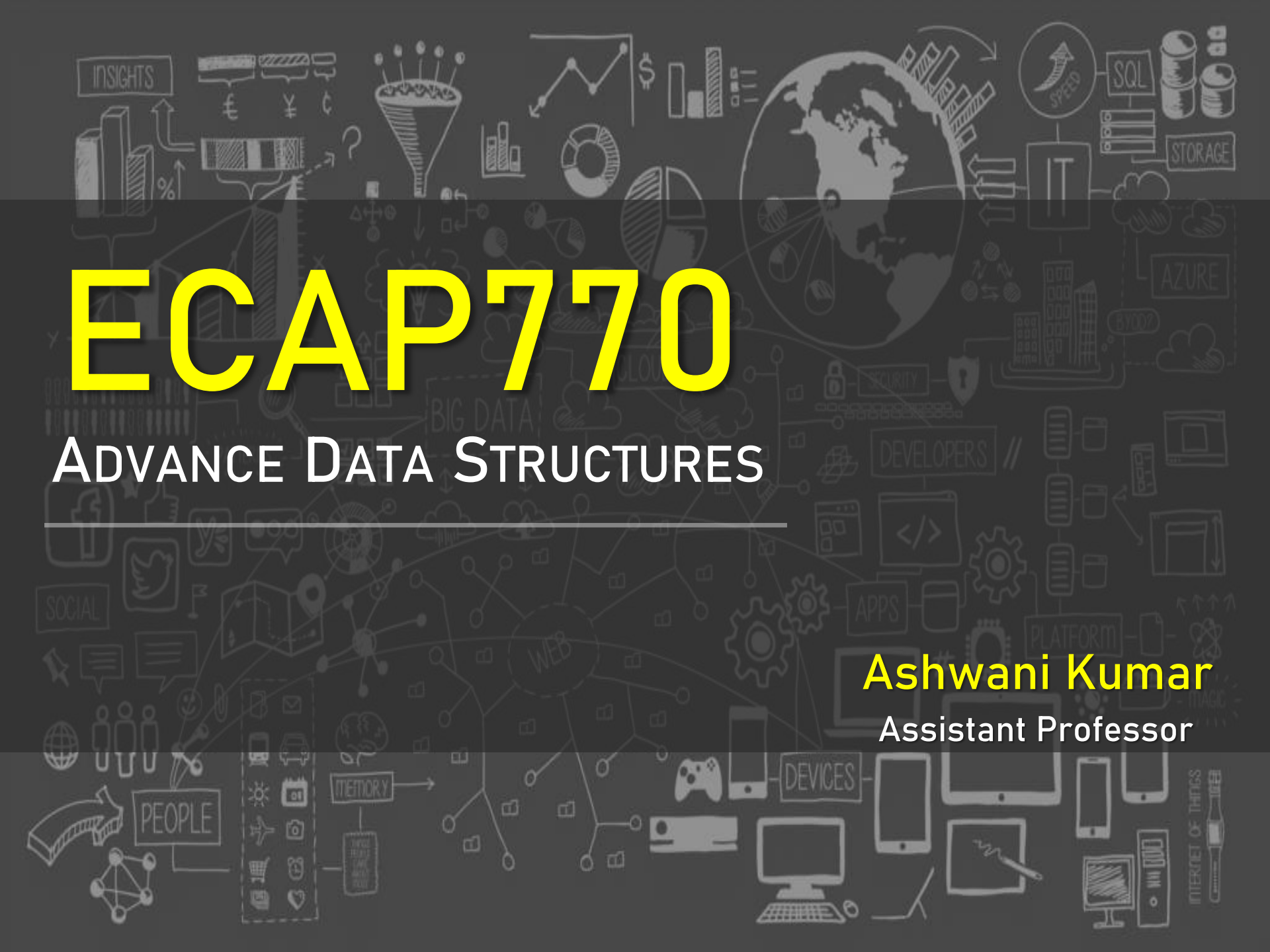


ECAP770

Advance Data Structures

Ashwani Kumar

Assistant Professor

Learning Outcomes

After this lecture, you will be able to

- Understand AVL tree

AVL tree

- AVL tree is a self balancing Binary Search Tree.
- AVL Tree is invented in 1962 by

  Adelson

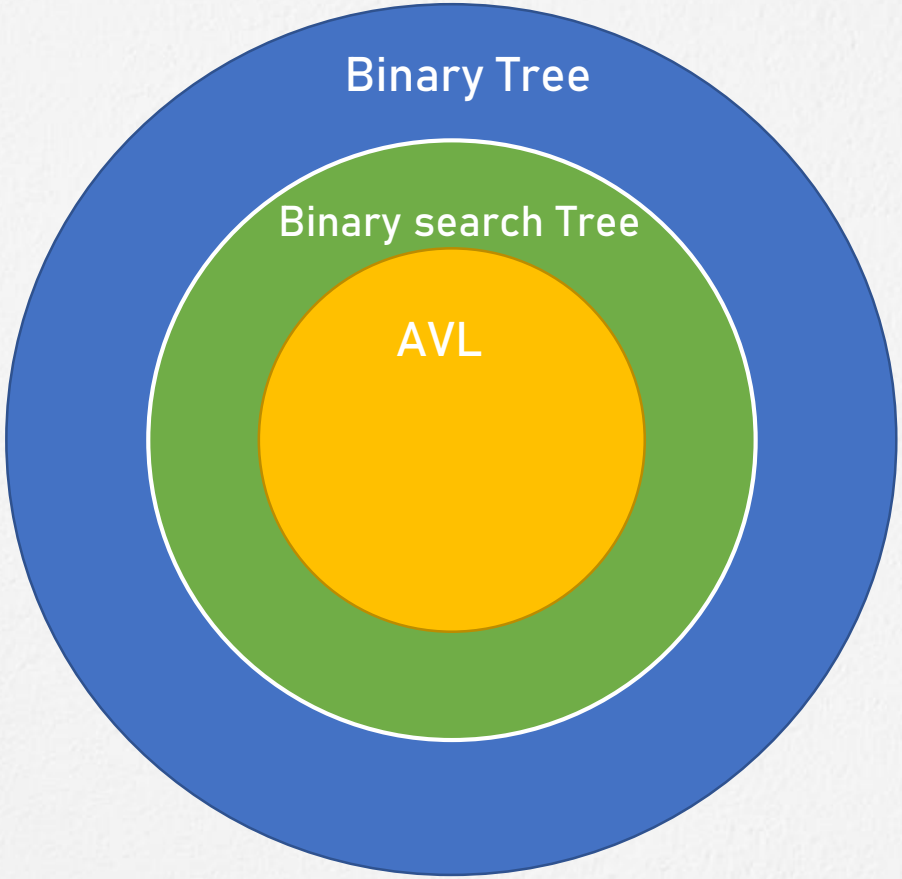
  Velsky

  Landis

AVL tree

- AVL Tree is defined as height balanced binary search tree.
- In AVL tree each node is associated with a balance factor which is calculated by subtracting the height of its right sub-tree from that of its left sub-tree.

AVL Tree



Why AVL Tree

- AVL tree controls the height of the binary search tree.
- The time taken for all operations in a binary search tree of height h is O(h).
- For skewed BST it can be extended to O(n) (worst case).
- By limiting this height to log n, AVL tree imposes an upper bound on each operation to be O(log n) where n is the number of nodes.

Why AVL Tree

Data = 55, 40, 38

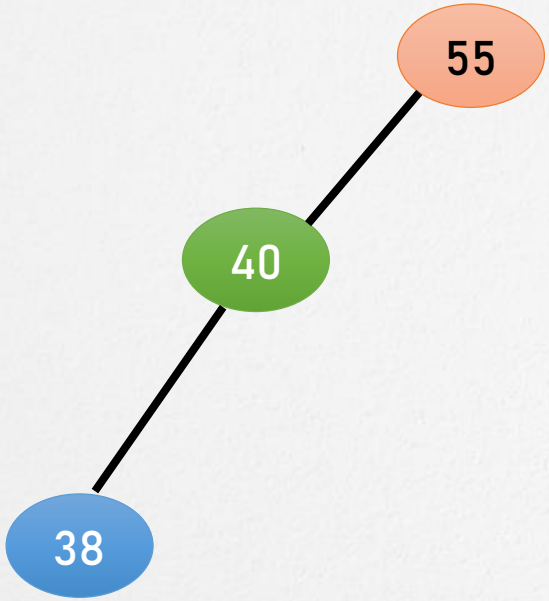


Fig. A

Data = 55, 60, 65

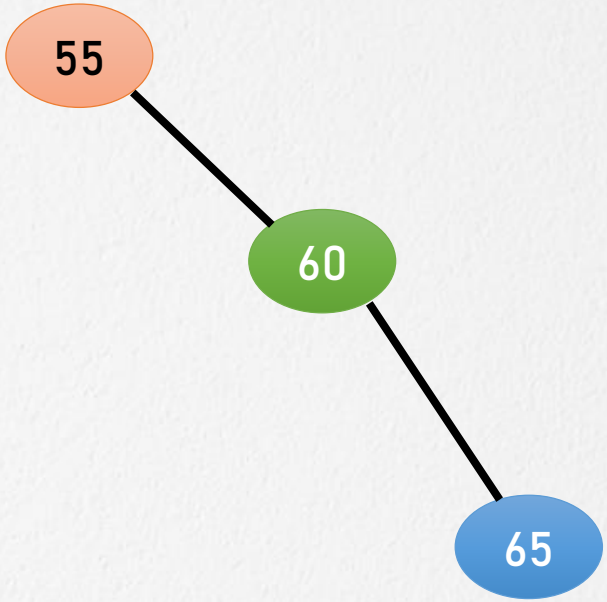
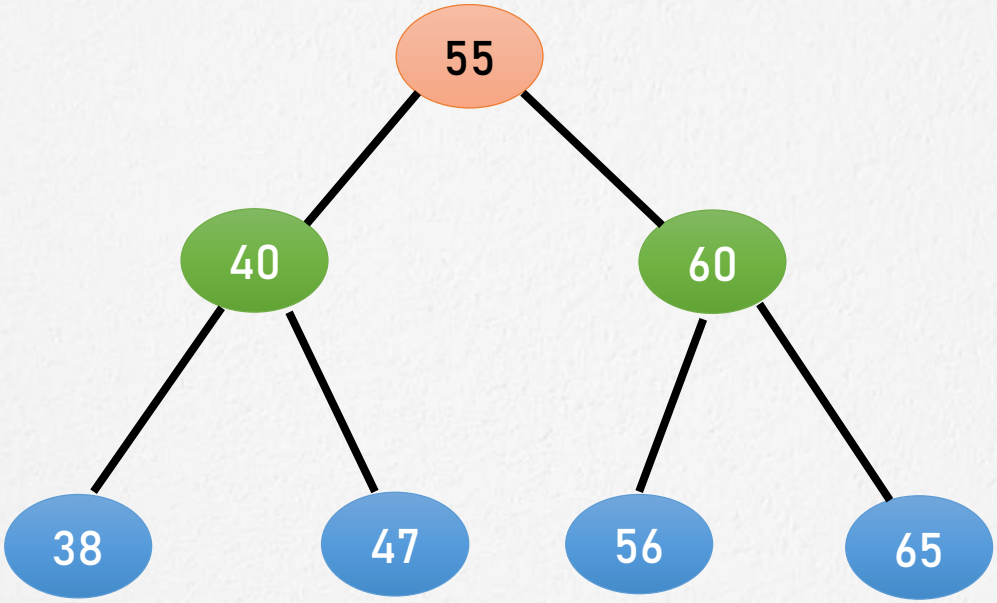


Fig. B

AVL tree (Balanced)



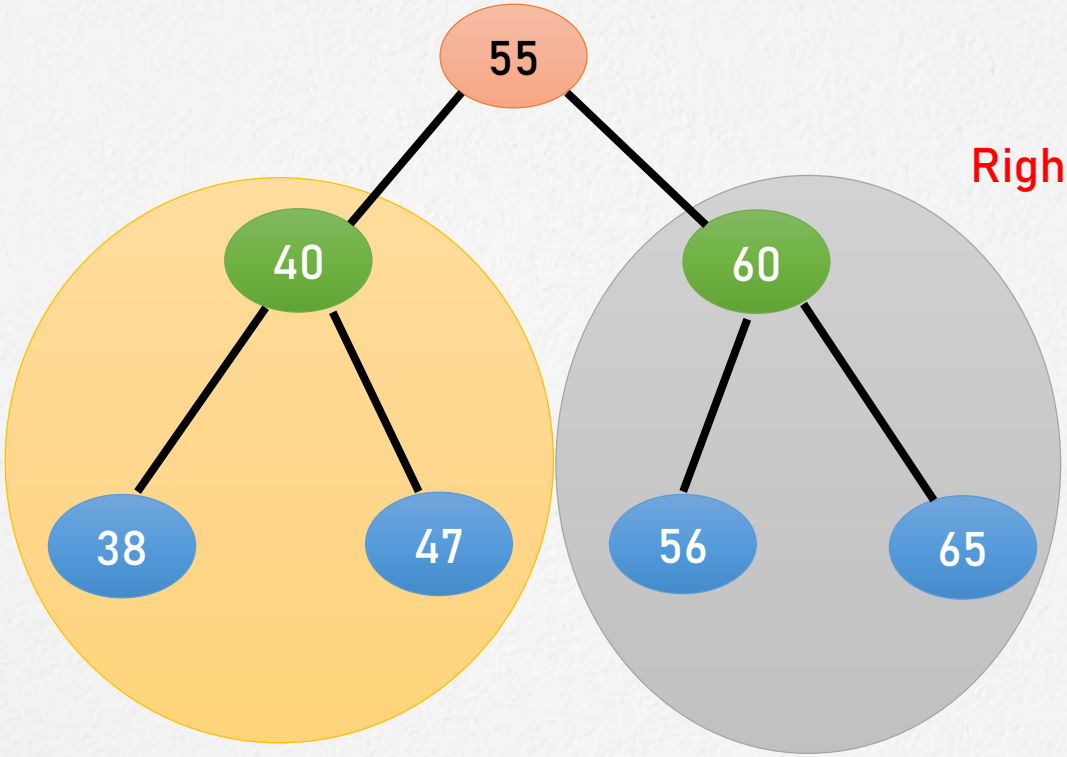
Balance Factor

- Balance factor of a node in an AVL tree is the difference between the height of the left sub tree and that of the right sub tree of that node.
- Balance Factor = (Height of Left Sub tree - Height of Right Sub tree) or (Height of Right Sub tree - Height of Left Sub tree)
- Balance factor value are: -1, 0 or 1.

Balance Factor

- If balance factor of any node is 1, it means that the left sub-tree is one level higher than the right sub-tree.
- If balance factor of any node is 0, it means that the left sub-tree and right sub-tree contain equal height.
- If balance factor of any node is -1, it means that the left sub-tree is one level lower than the right sub-tree.

AVL tree



Left Sub Tree = 2

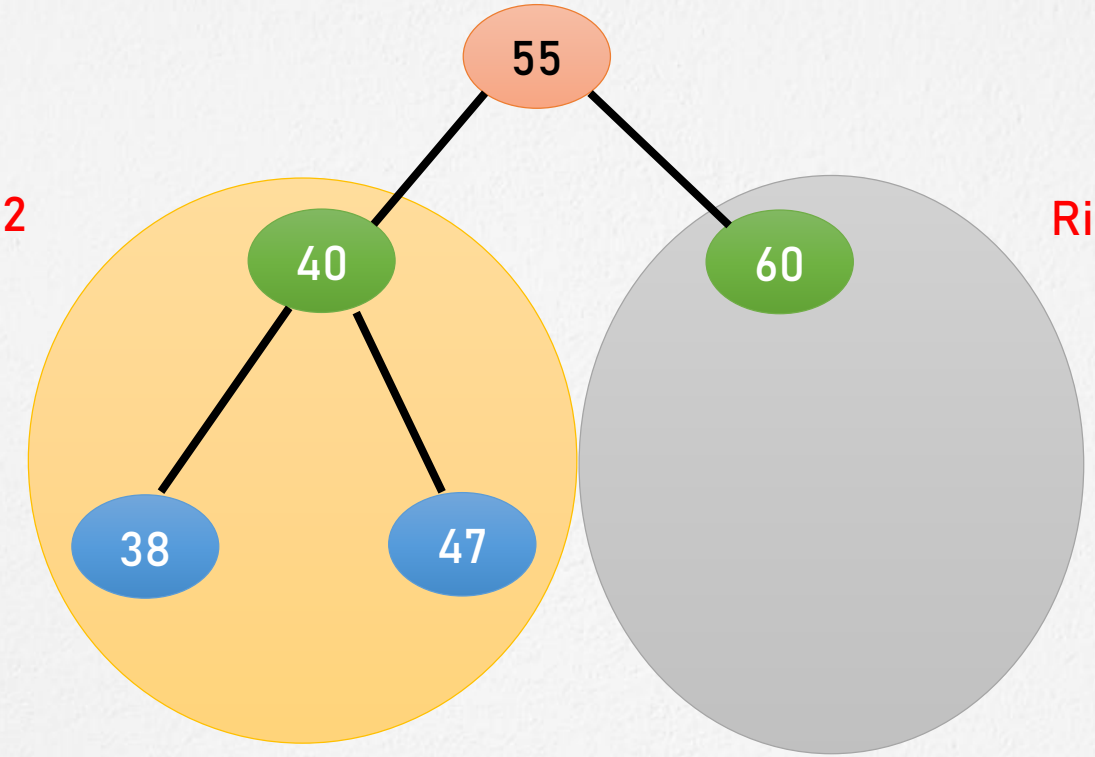
Right Sub Tree = 2

Balancing factor = 0

2-2 = 0 (Balanced Tree)

0 = (0, 1, -1)

AVL tree (Left Heavy Tree)



Left Sub Tree = 2

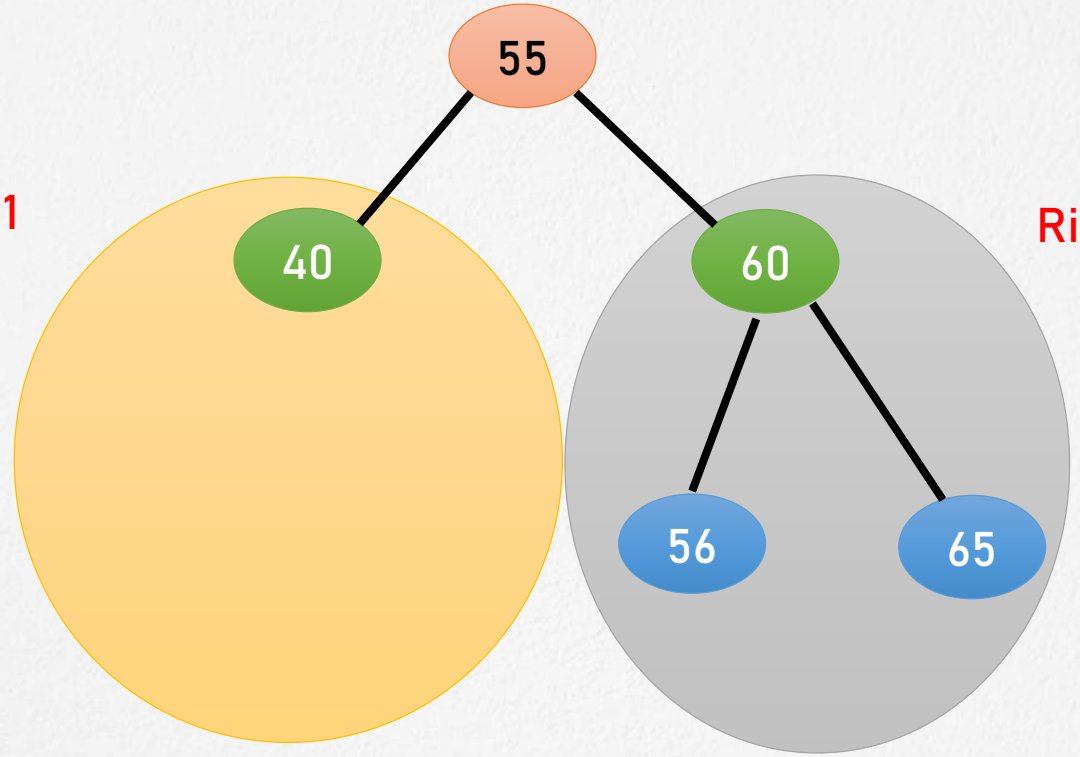
Right Sub Tree = 1

Balancing factor = 1

2-1 = 1 (Left Heavy Tree)

0 = (0, 1, -1)

AVL Tree (Right Heavy Tree)



Left Sub Tree = 1

Right Sub Tree = 2

Balancing factor = -1

1-2 = -1 (Right Heavy Tree)

0 = (0, 1, -1)

AVL Tree operations : Insertion

- Insertion: Insertion in AVL is same as binary search tree.
- Insertion may lead to violation in the tree property and therefore the tree may need balancing.
- The tree can be balanced by applying rotations.

AVL Tree operations: Deletion

- Deletion: Deletion in AVL is same as binary search tree.
- Deletion may lead to violation in the tree property and therefore the tree may need balancing.
- The tree can be balanced by applying rotations.

AVL Tree Rotations

- Left rotation
- Right rotation
- Left-Right rotation
- Right-Left rotation

That's all for now...