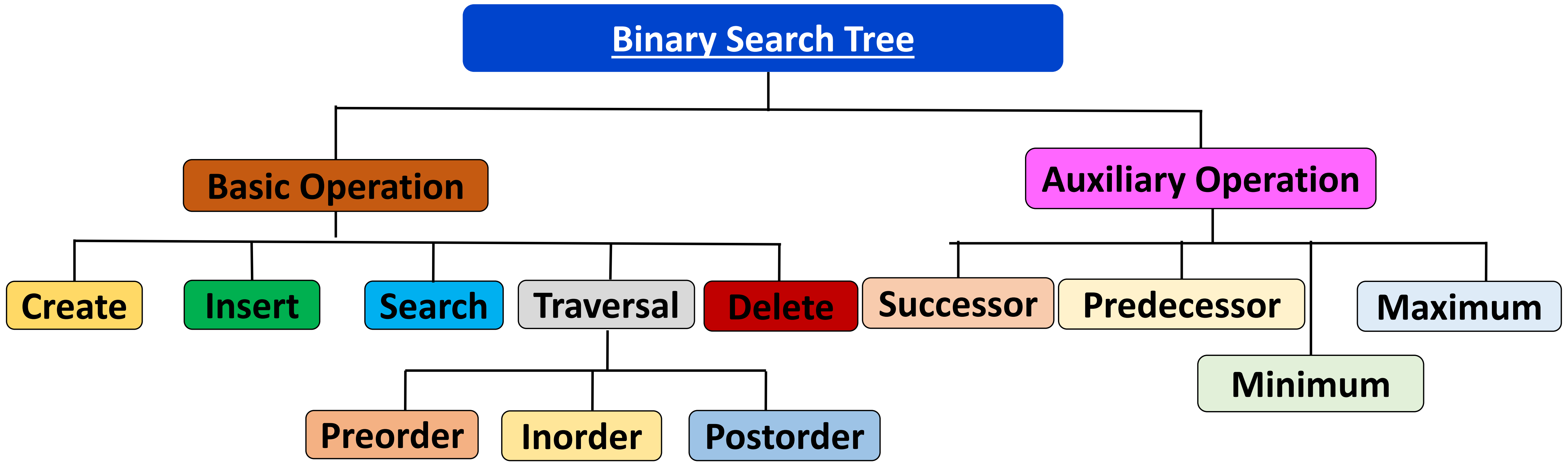
CS2x1:Data Structures and Algorithms

Koteswararao Kondepu

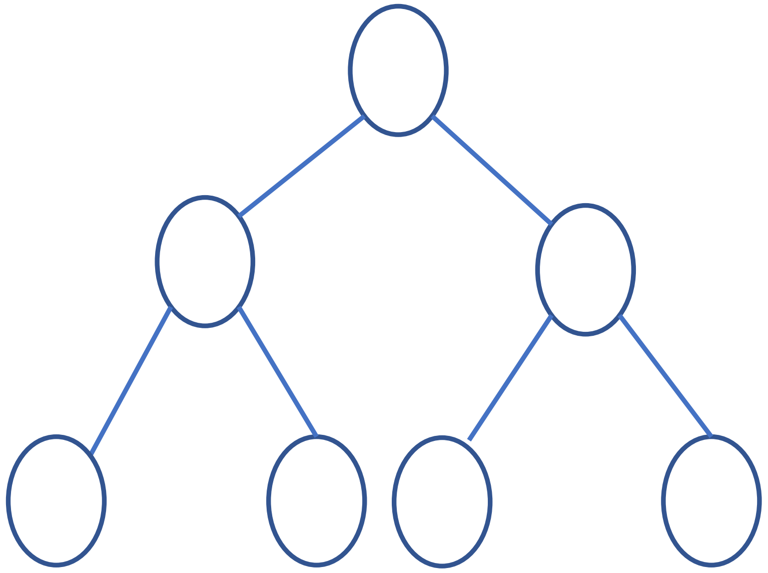
k.kondepu@iitdh.ac.in

Recap: Binary Search Tree



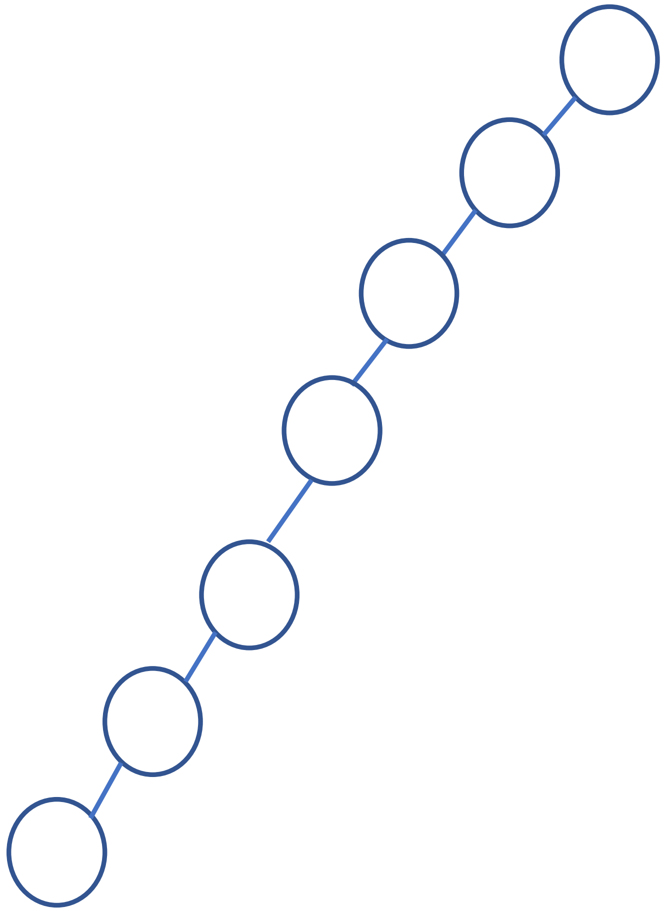
Motivation

Example: 40, 50, 20, 60, 30, 15, 45



Time complexity *O(logn),* where n is the number of nodes

Example: 60, 50, 45, 40, 30, 20, 15

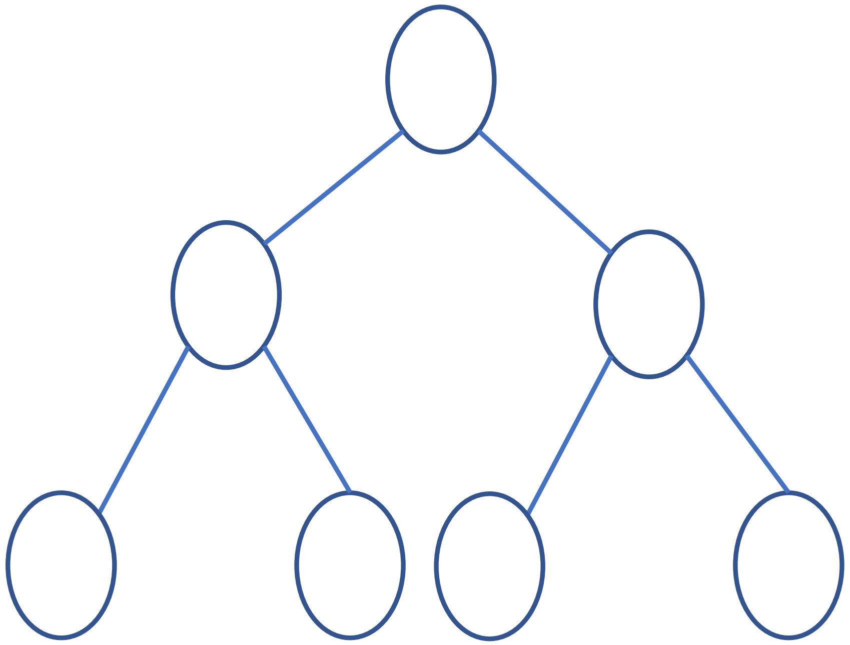


Time complexity *O(n)*

Height Balanced Binary Search Tree

Example:

40, 20, 60, 15, 30, 55, 70



Define: ***Balance Factor***

- A balance factor of a node in a binary search tree is the difference in height between its left and right subtree
- *BF(k) = height (left) – height (right) =* $h_L - h_R$

BF(k) if k = 0 → Perfect Binary Tree

Define: ***Height Balanced Binary Search Tree***

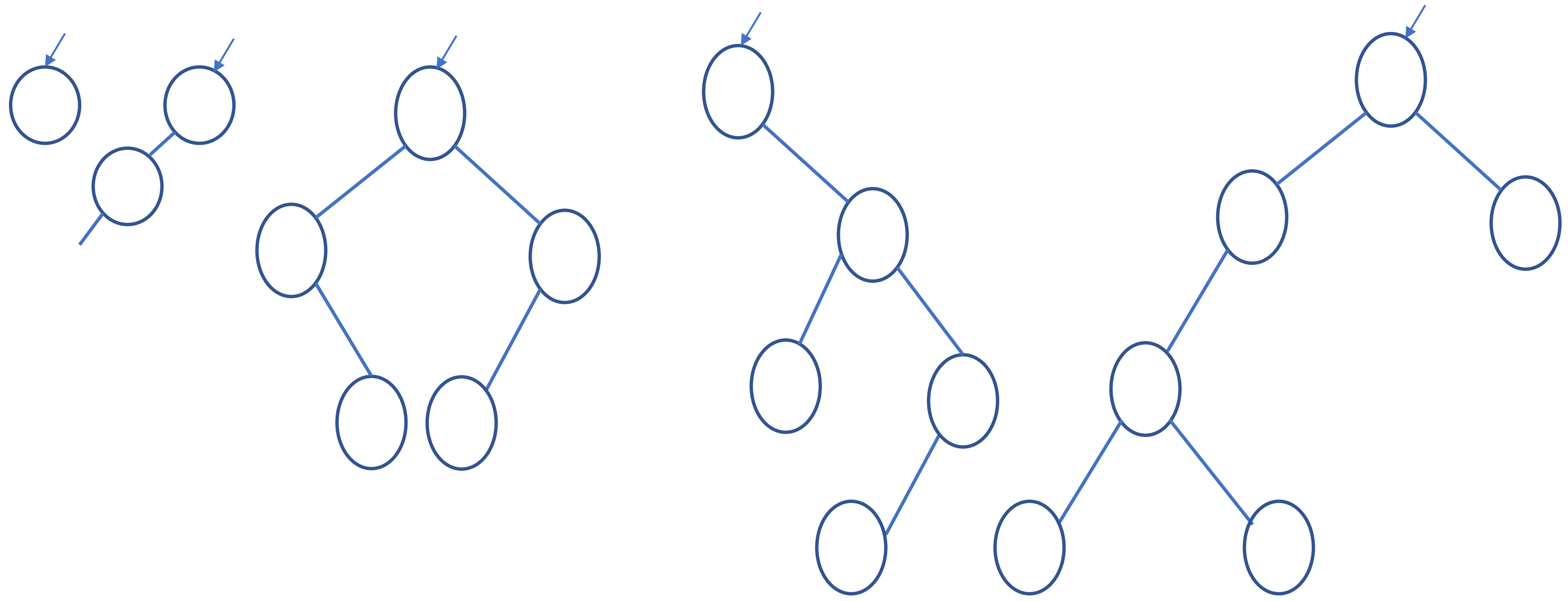
A binary search tree is said to be Height Balanced Binary Search Tree if all its nodes have a balance factor of 1, 0, or -1

$$|BF| = |h_L - h_R| \leq 1$$

Balance Factor

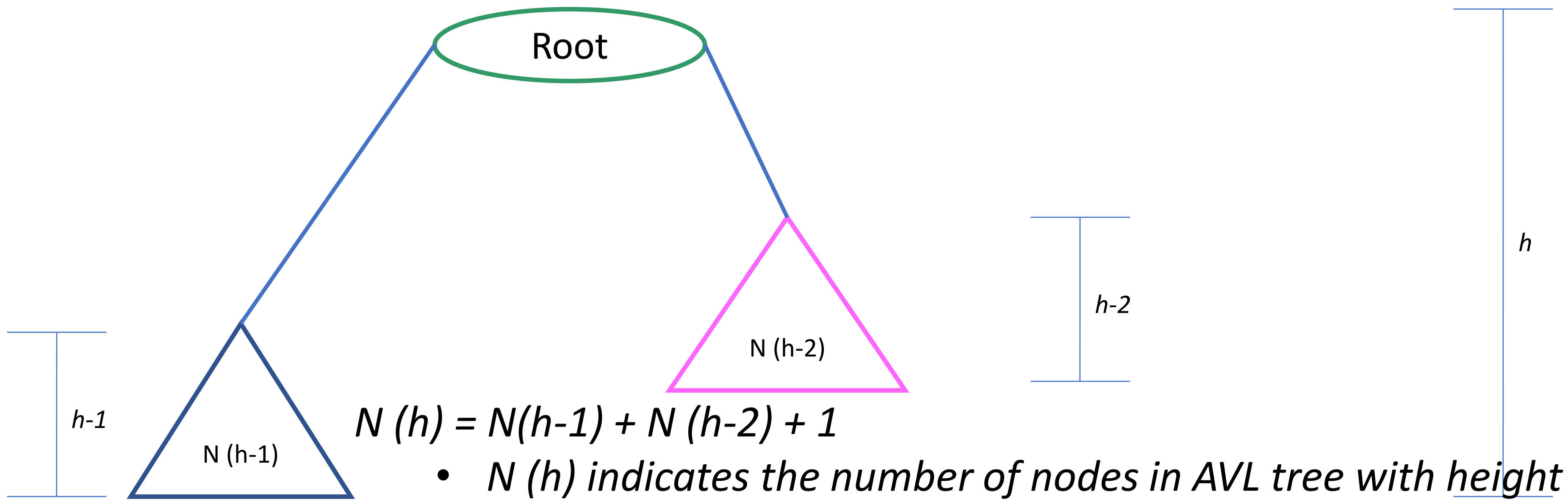
Define: Balance Factor

- A balance factor of a node in a binary tree is the difference in height between its left and right subtree
- *BF(k) = height (left) – height (right)*



AVL (Adelson – Velskii and Landis) Tree

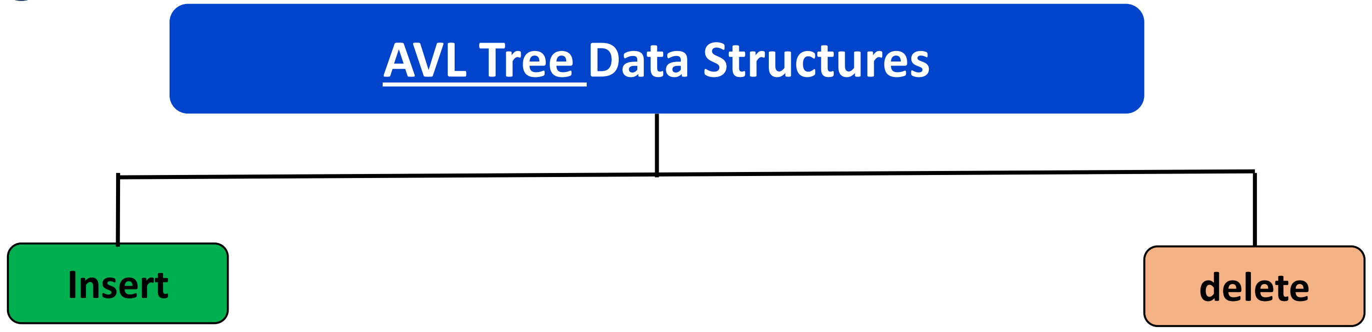
- *Definition: In an AVL Tree, the balance factor for every node must have at (magnitude) most ≤ 1 after every operation (e.g., insertion/deletion)*
- If inserting/deleting a node would result in a binary search tree with *BF>1*, then it *requires to fix immediately* → an AVL Tree is *self-balancing*
- *AVL Tree → should hold both Binary Search Tree (BST) properties and AVL tree properties*

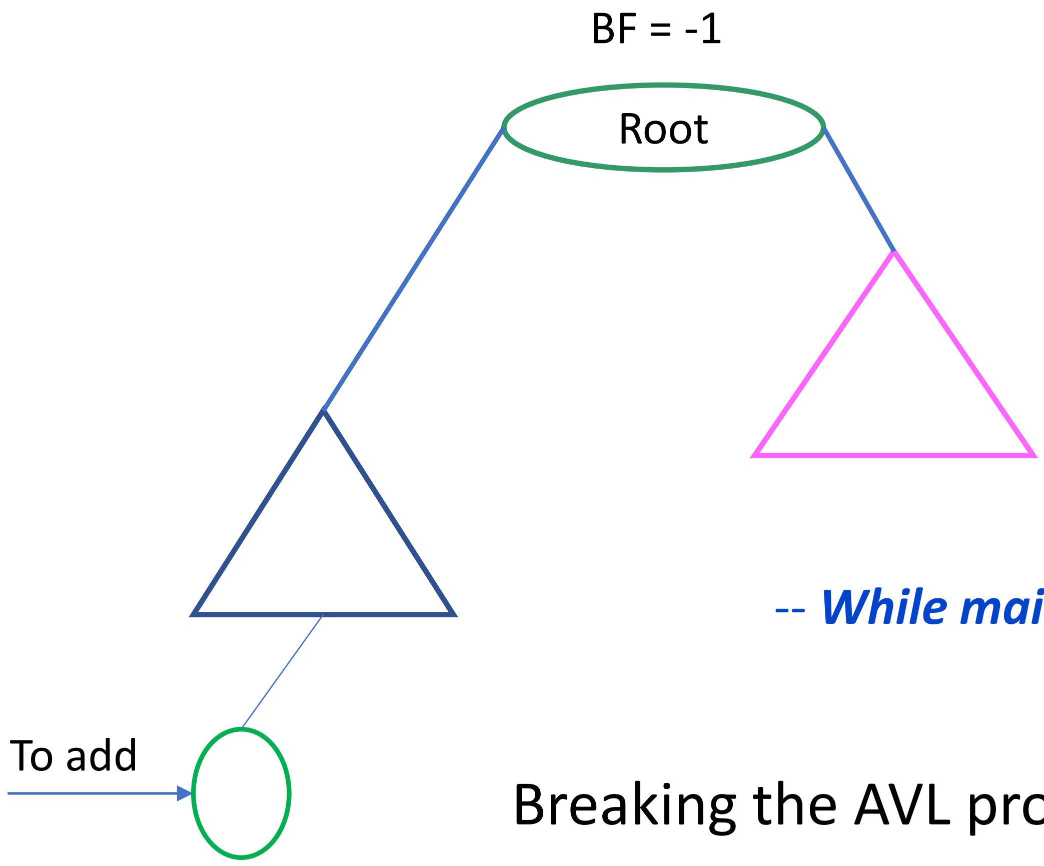


N (h) = N(h-1) + N (h-2) + 1

- *N (h) indicates the number of nodes in AVL tree with height h*
- *N (h-1) indicates the minimum number of nodes with height h-1*
- *N (h-2) indicates the number number of nodes with height h-2*
- *"1" indicates the current node*

AVL Tree Operations

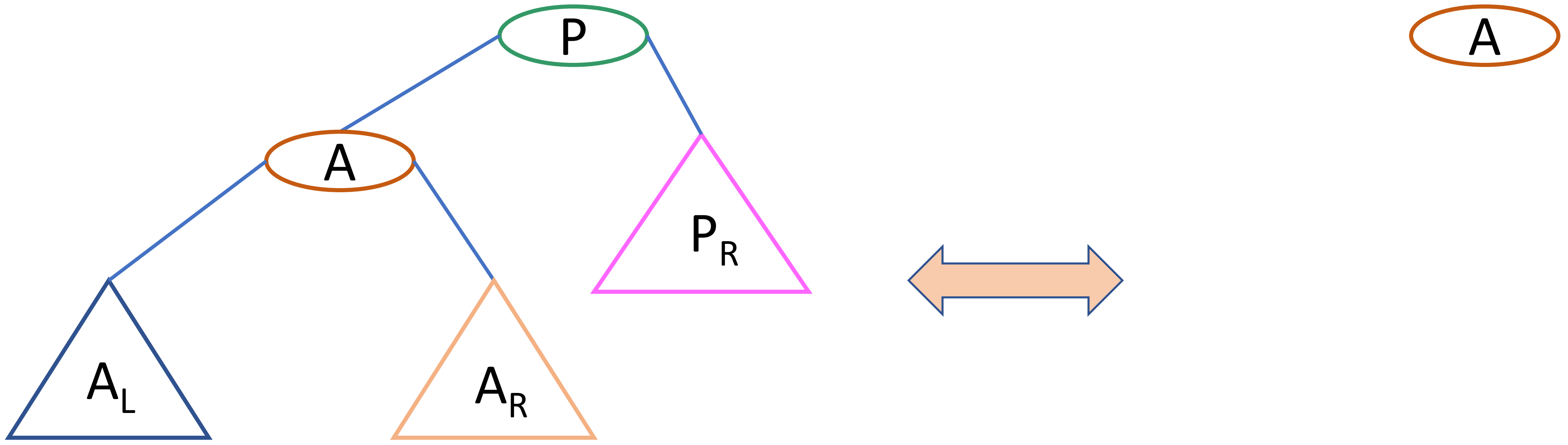




-- While maintaining the BST properties

Breaking the AVL property?

AVL Tree: Example - Rotations



(i) $P > A$

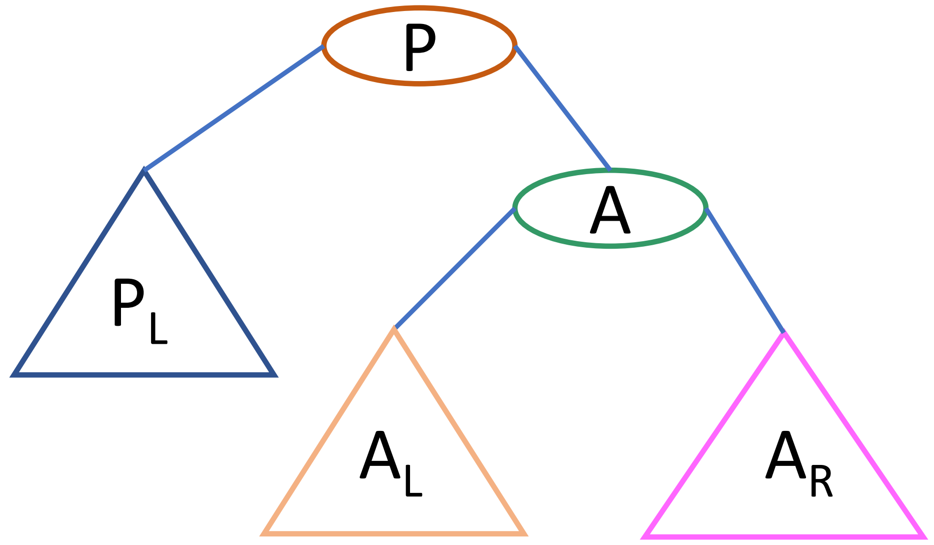
(ii) $P_L > P$

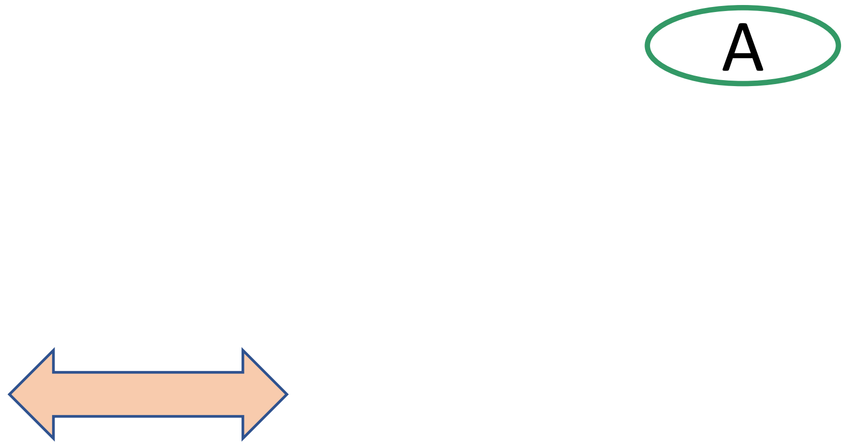
(iii) $A_R > A$ *and* $A_R < P$

(iv) $A_L < A$

- Rotate at the root toward right
- Left child formally becomes the root
- Depth of left subtree A_L is decreased by one
- Depth of right subtree P_R is increased by one
- No changes in the depth of A_R

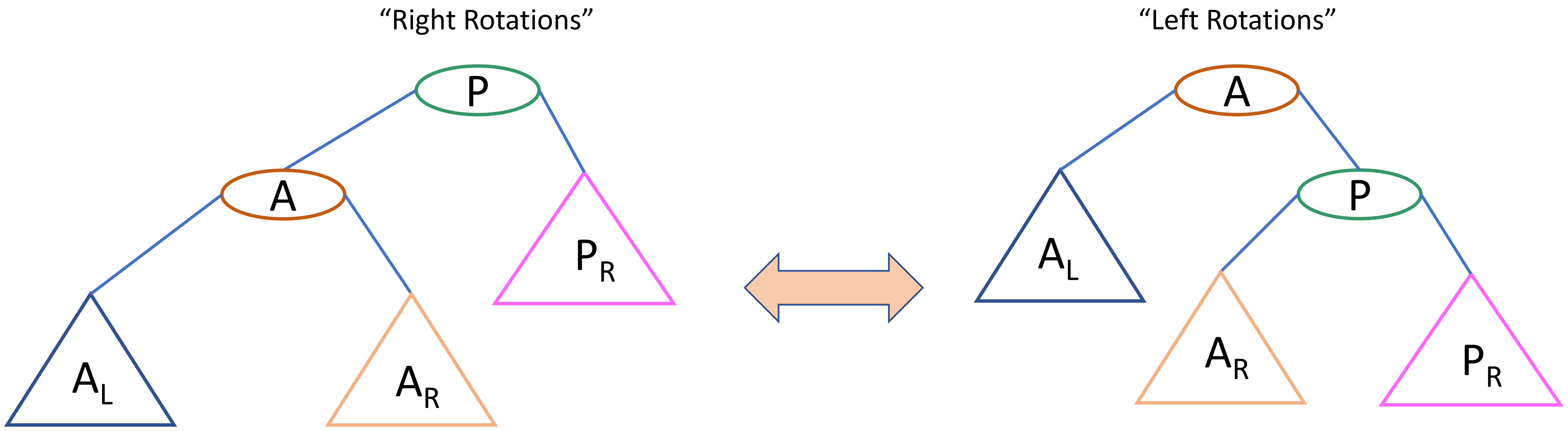
AVL Tree: Example - Rotations





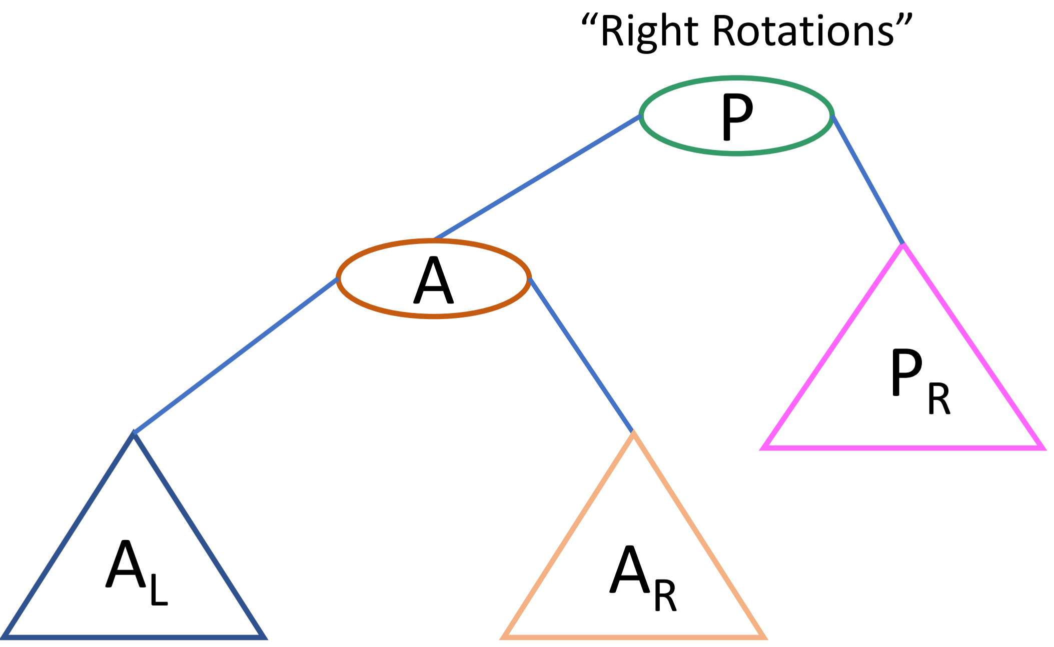
- Rotate at the root toward right
- Right child formally becomes the root
- Depth of left subtree P_L is increased by one
- Depth of right subtree A_R is decreased by one
- No changes in the depth of A_L

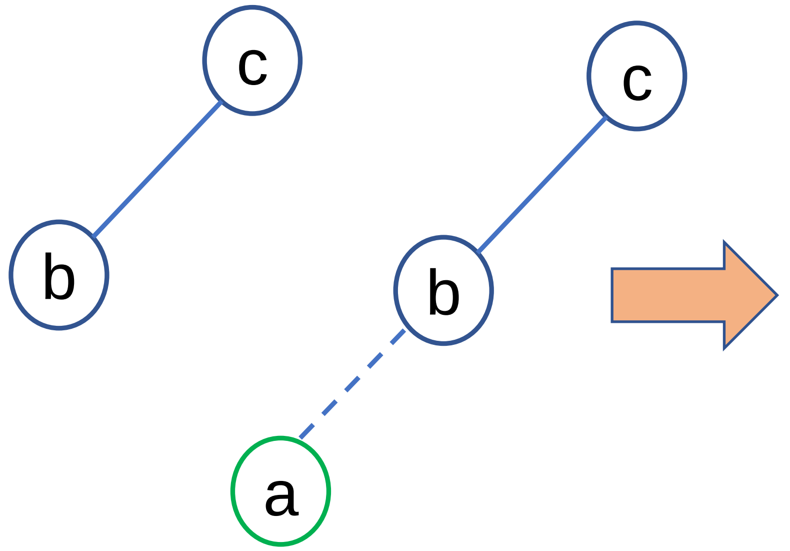
AVL Tree: Example - Rotations



- Right and Left Rotations are mirror images

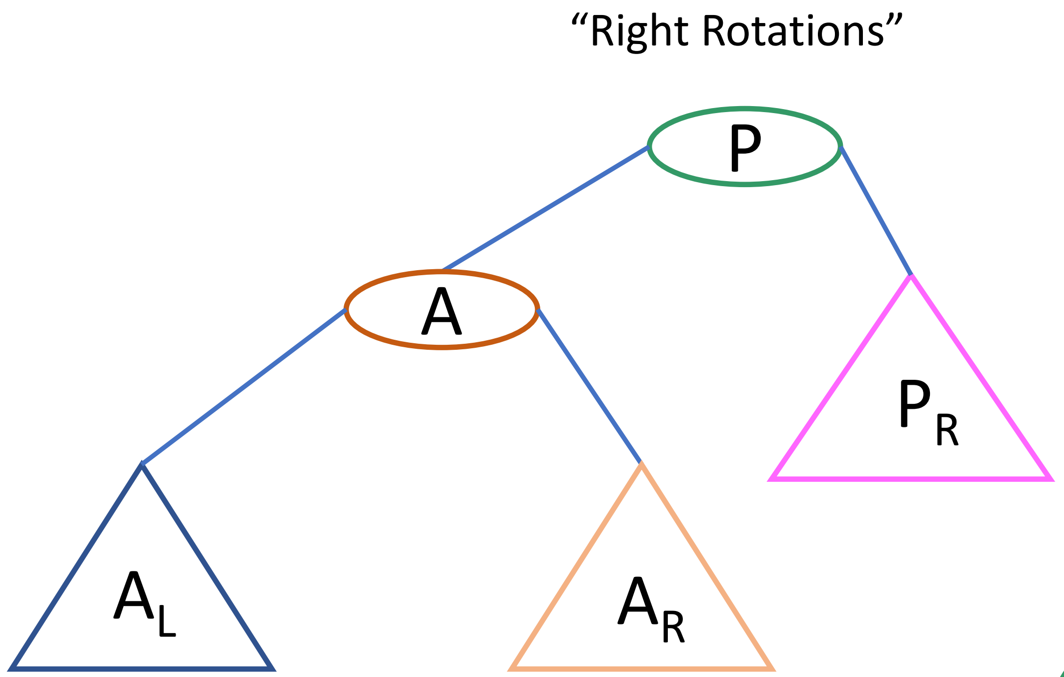
AVL Tree: Right Rotations (1)

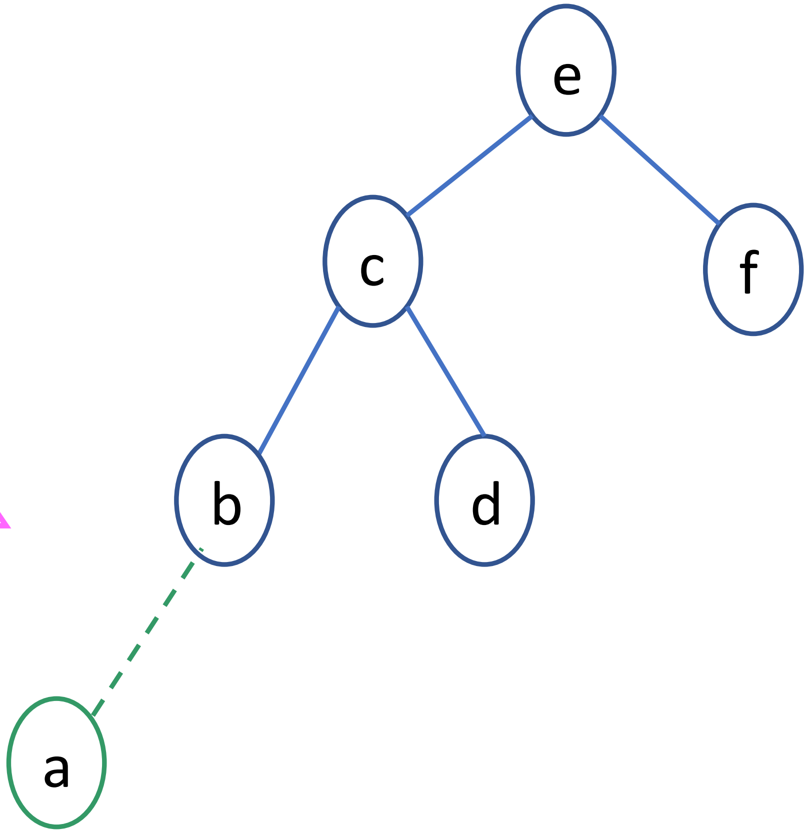




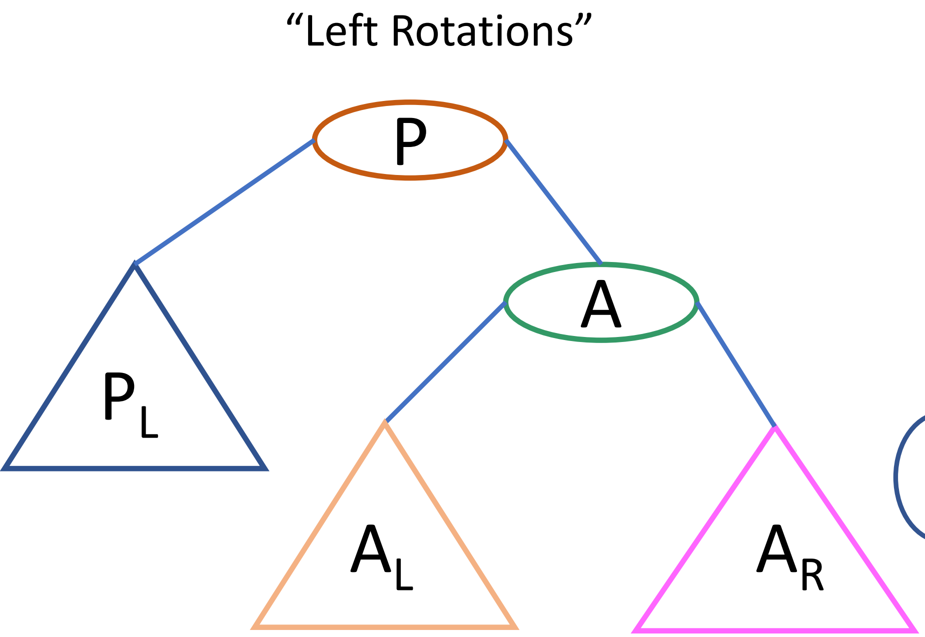
- Rotate at the root toward right
- Left child formally becomes the root
- Depth of left subtree A_L is decreased by one
- Depth of right subtree P_R is increased by one
- No changes in the depth of A_R

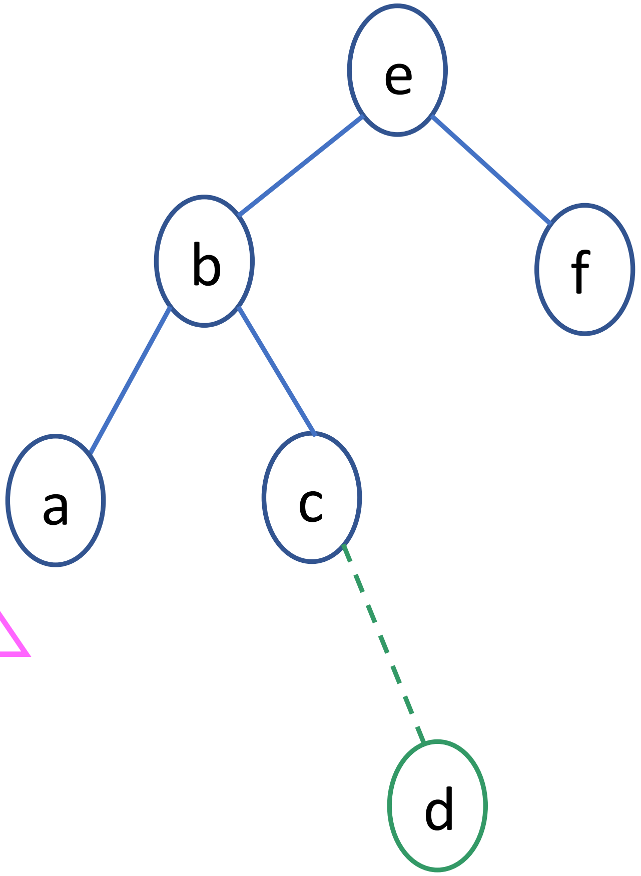
AVL Tree: Right Rotations (2)





AVL Tree: Left Right (LR) Rotations

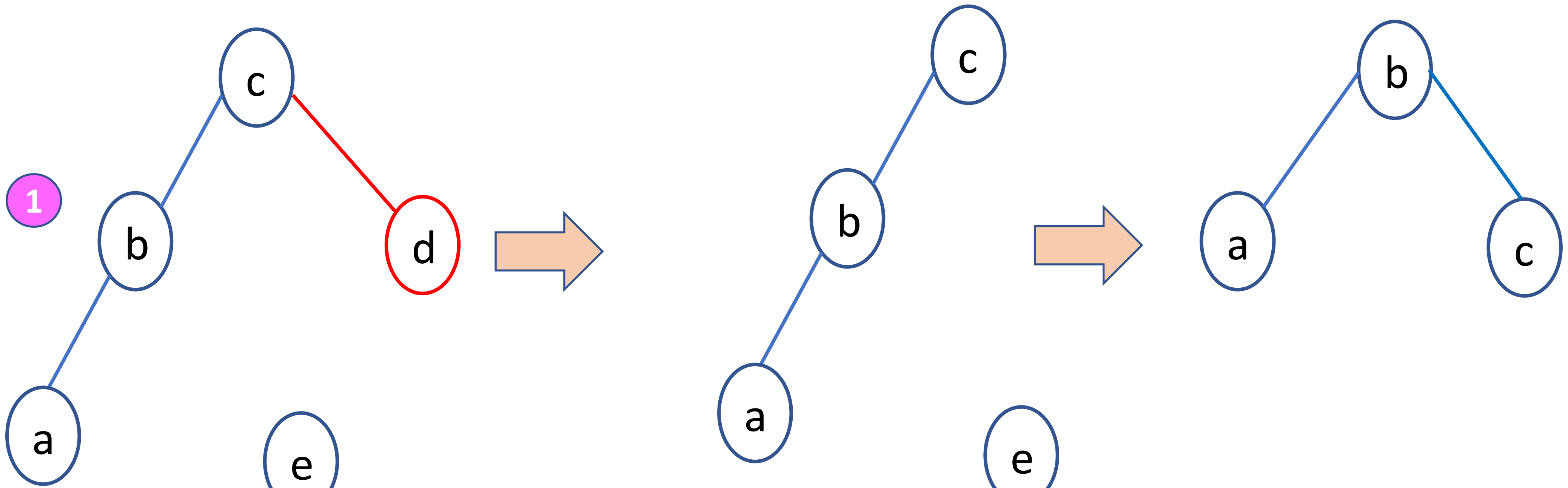


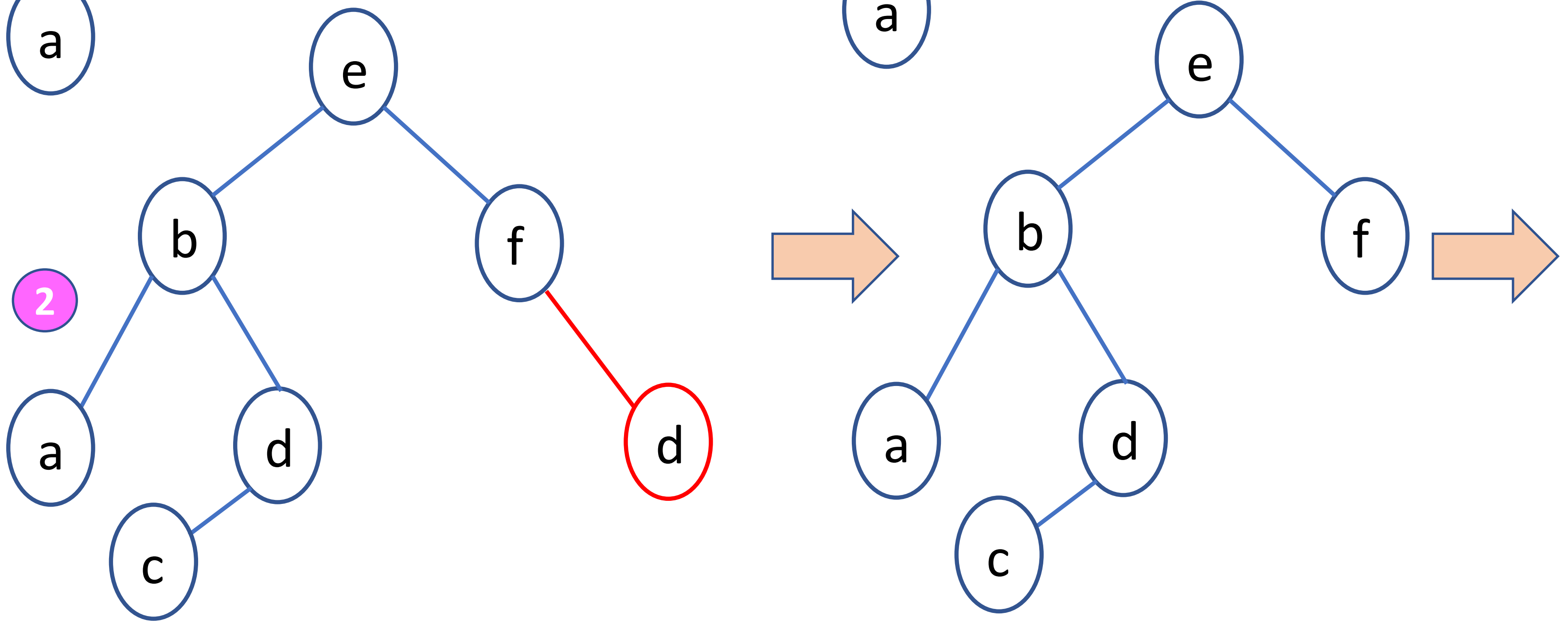


Left Rotation

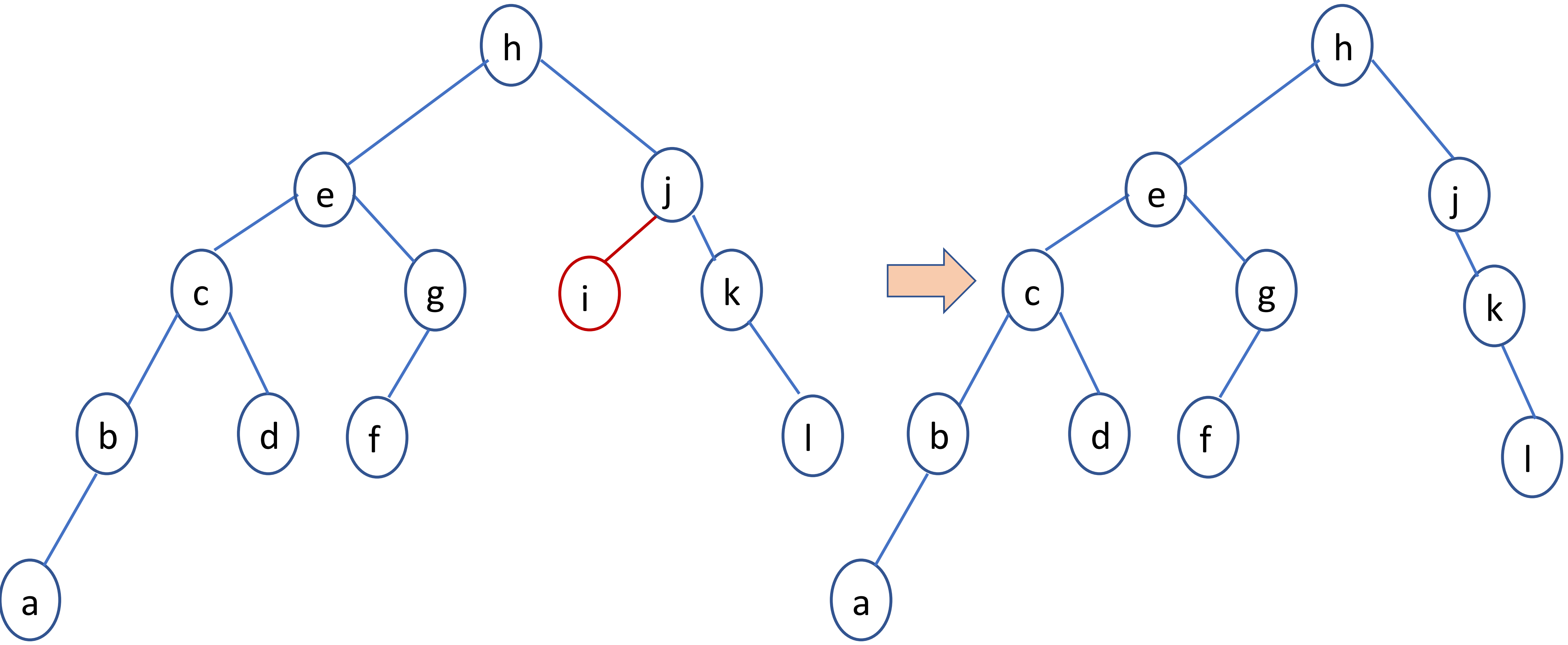
Right Rotation

AVL Tree: Deletion → Left-Right Rotation (1)

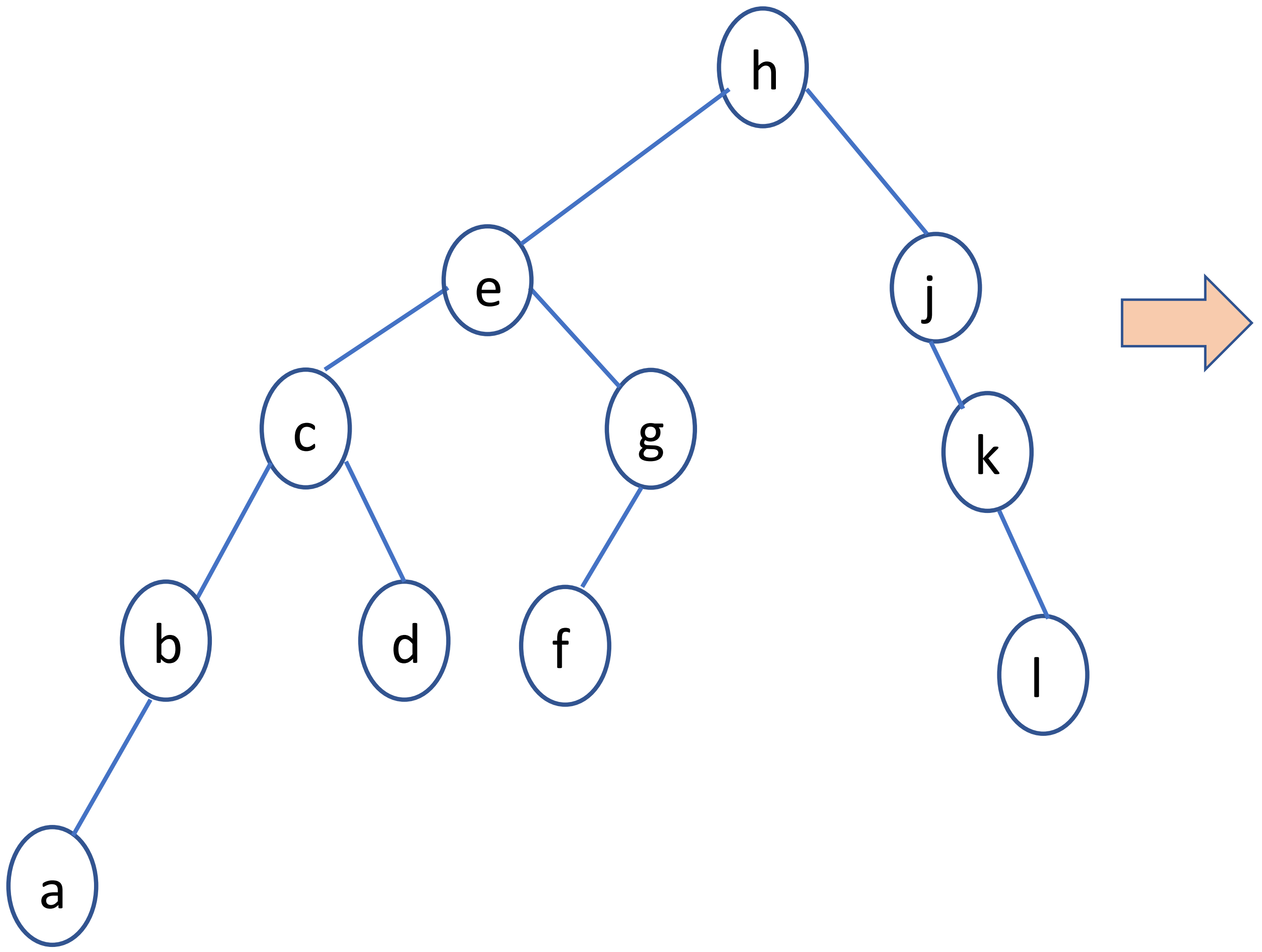




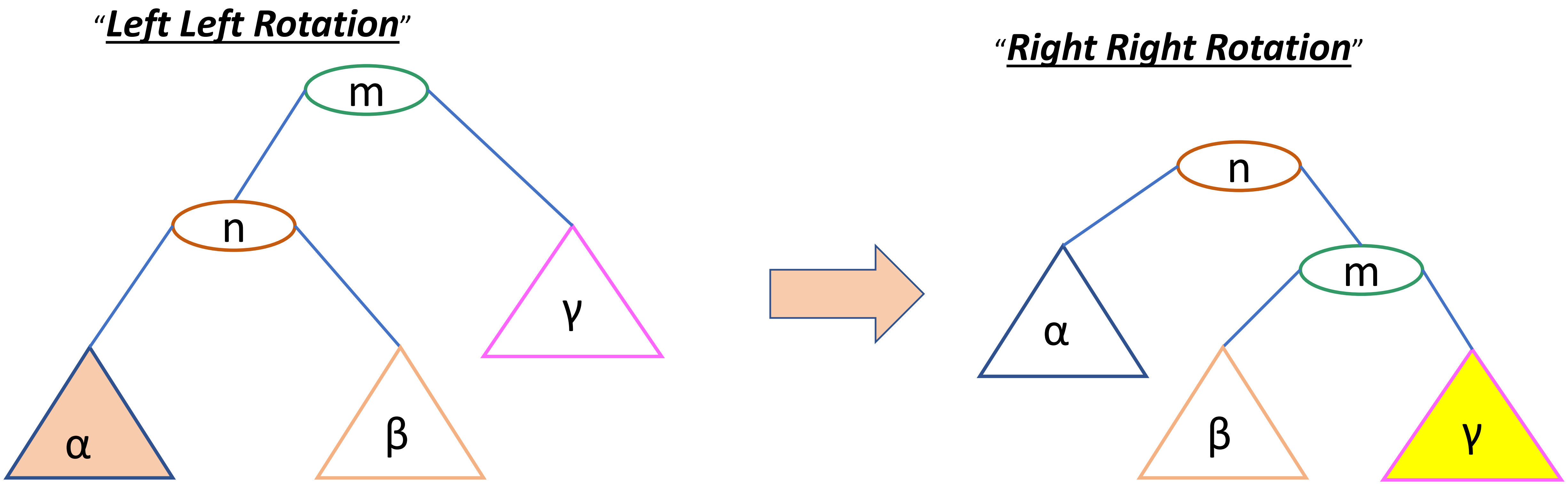
AVL Tree: Deletion → Left-Right Rotation (2)



AVL Tree: Deletion → Left-Right Rotation (3)



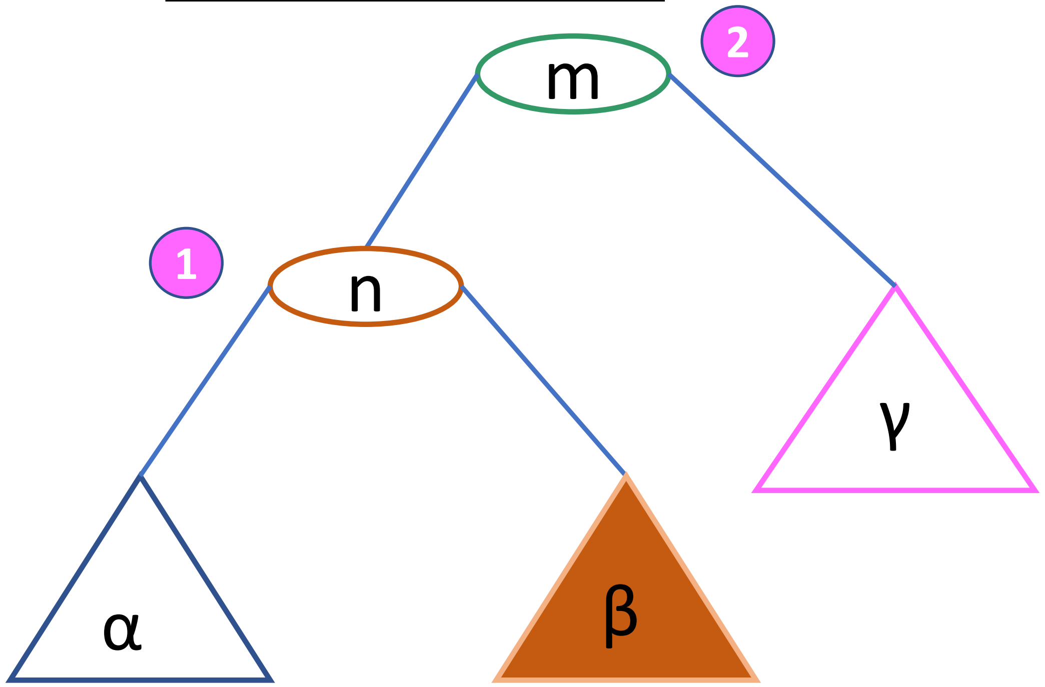
AVL Tree: Rotation

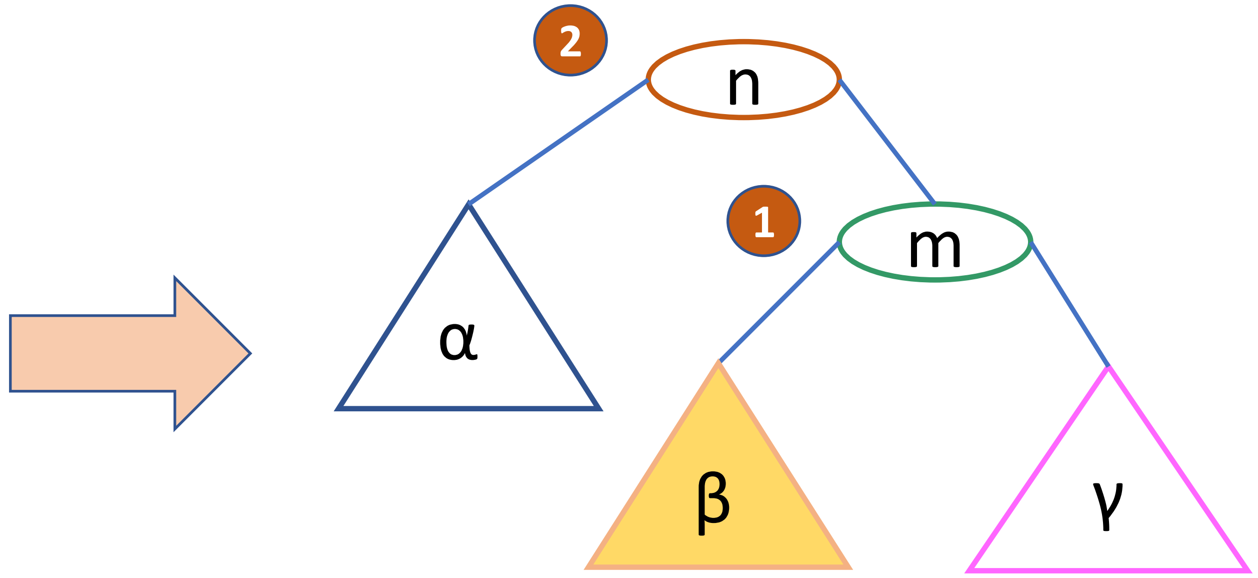


AVL Tree: Rotation

“***Left Right Rotation***”

“***Right Left Rotation***”

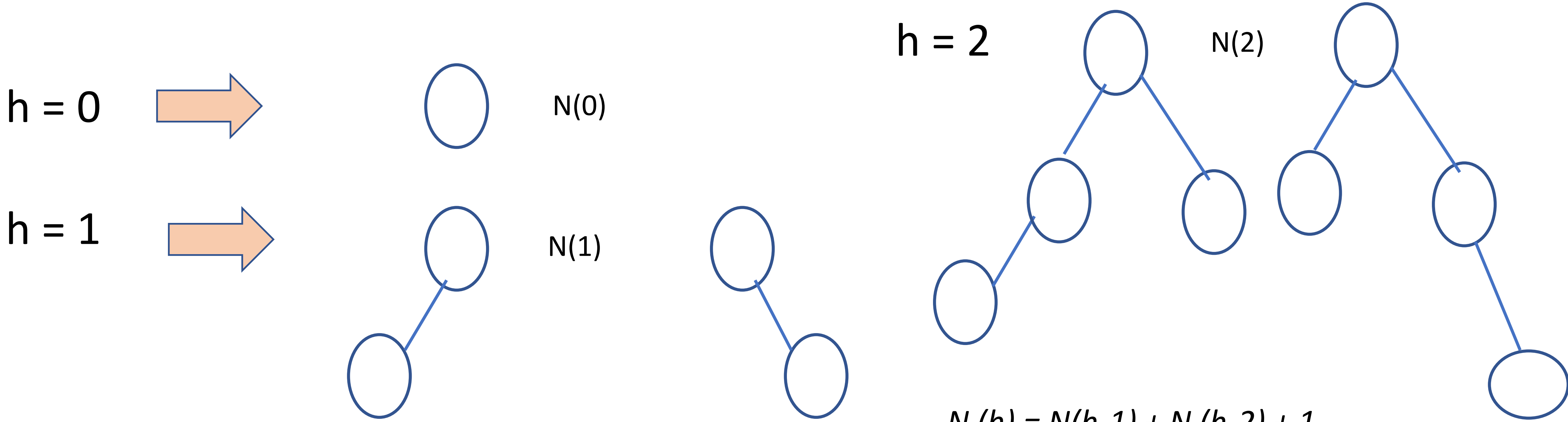




1 Left Rotation 2 Right Rotation

1 Right Rotation 2 Left Rotation

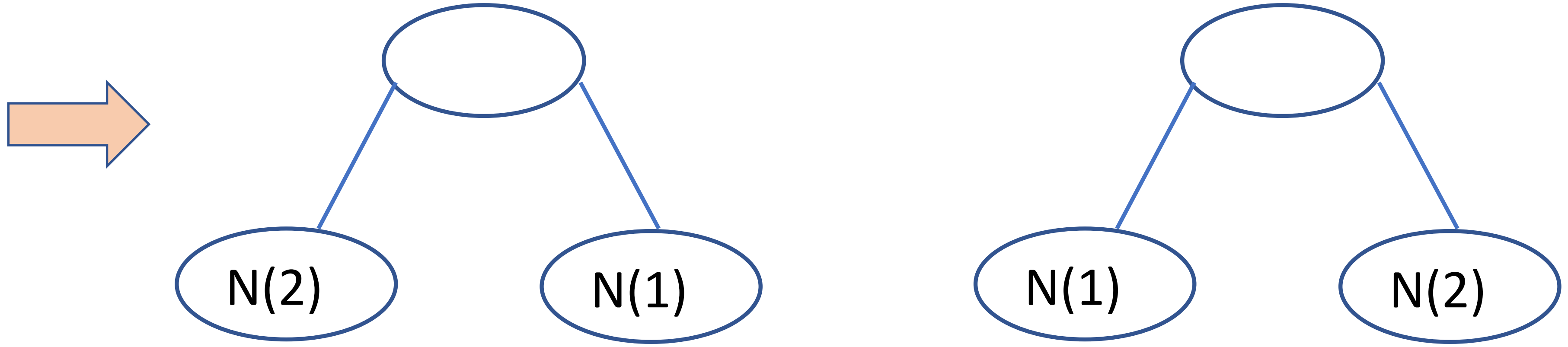
What is the minimum Number of nodes in an AVL tree of height h



$N(h) = N(h-1) + N(h-2) + 1$

What is the minimum Number of nodes in an AVL tree of height h?

h = 3



N (3) = N(2) + N (1) + 1

N (h) = N(h-1) + N (h-2) + 1

What is the number of nodes in an AVL tree of height 6?

$N(0) = 1$

$N(1) = 2$

$N(2) = 4$

$N(3) = 7$

$N(4) = 12$

$$N(h) = N(h-1) + N(h-2) + 1$$

How many different trees can be formed with a minimal AVL tree height h?

$N(0) = 1$
$N(1) = 2$

$N(2) = 2 * N(1) * N(0)$

$N(h) = 2 * N(h-1) * N(h-2);\ h \geq 2$

Consider the following statements

S1 : An insertion in an AVL with n nodes requires Θ(n) rotations.
S2 : Finding the minimum value in an AVL tree containing n elements takes O(logn) time
S3 : Both Insert and find in an AVL tree takes O(logn) time.
S4: Finding the k-th largest item in a standard AVL tree implementation containing n elements takes O(n) time.
S5 : A set of numbers are inserted into an empty BST in sorted order and inserted into an empty AVL tree in random order. Listing all elements in sorted order from the BST is O (n), while listing them in sorted order from the AVL tree is O(logn)

thank you!

email:
k.kondepu@iitdh.ac.in