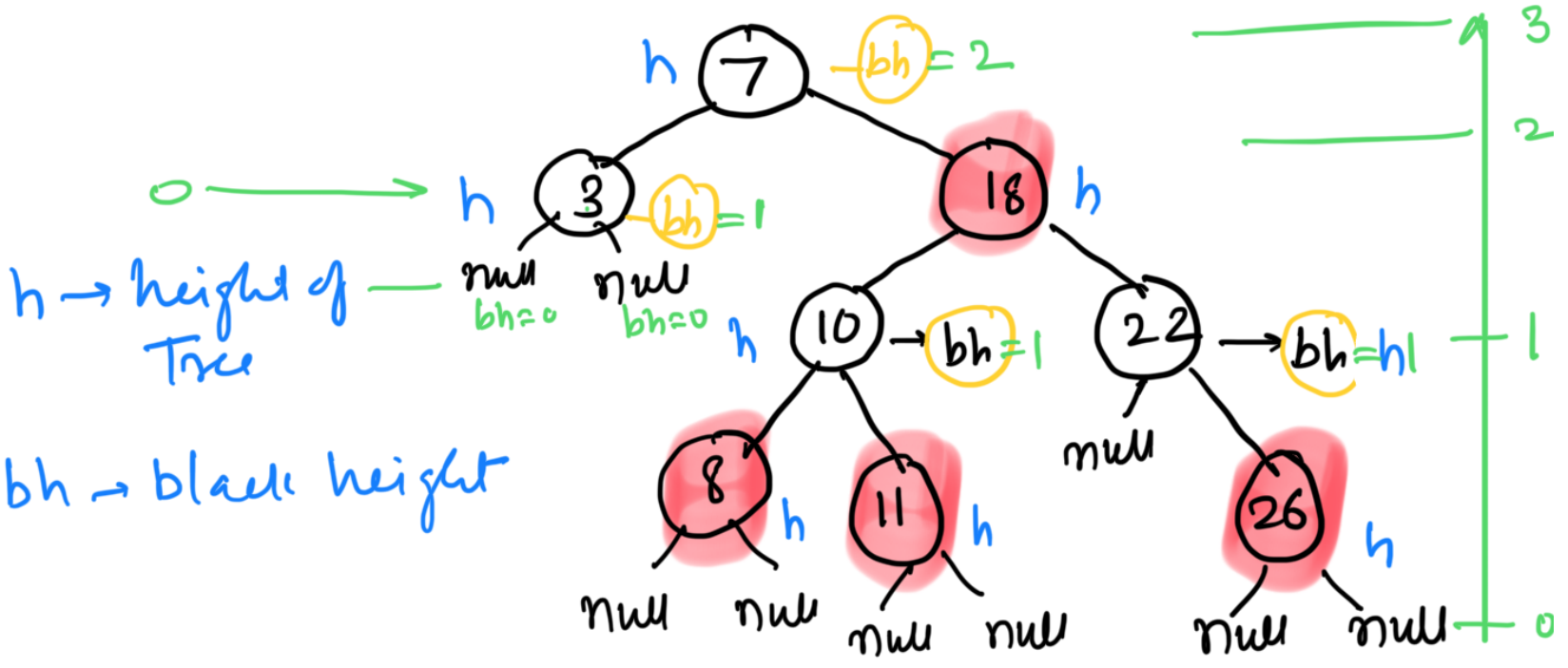
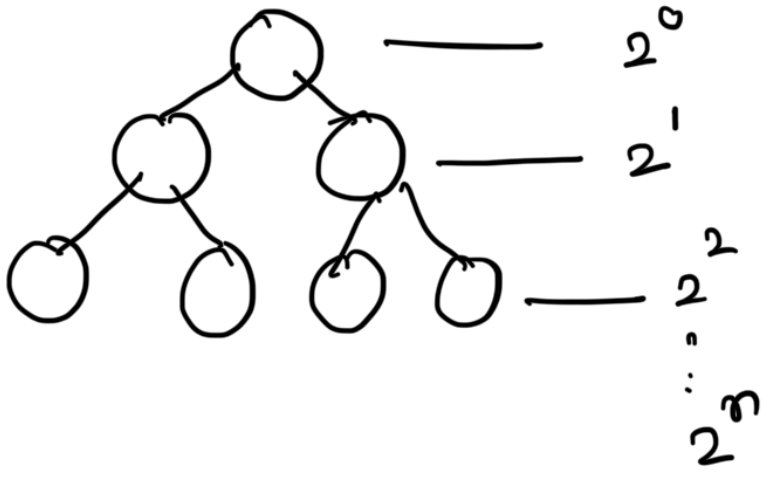
Red Black Tree





$$2^n = k$$

Apply the logs in both sides

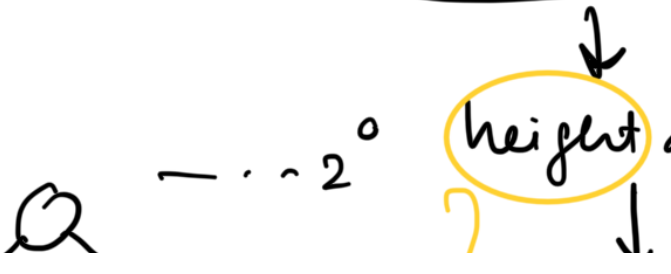
$$k = \log_2 n$$

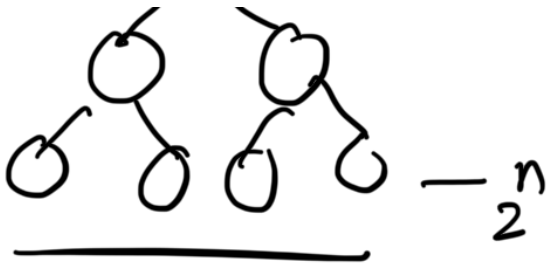
Binary Tree ← $O(\log n)$

Application → R & B Tree — $\log n$
BST — $\log n$
AVL — $\log n$

→ Balanced Tree → BST (based on BST)

→ Search-tree → data structure





$\frac{n}{2}$

$O(\log n)$

BT → 2^n → elements are distributed

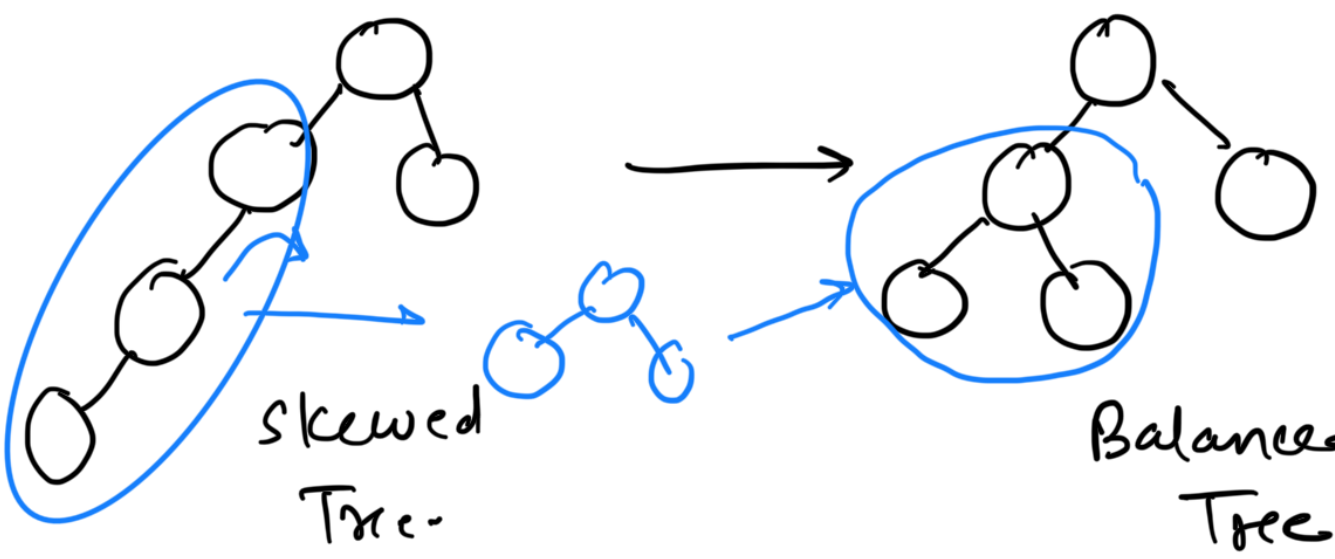
BST → LC < Root ≥ RC

R & BT → BTree data structr + color (Red & Black)

properties

1. Every node is either Red or Black
2. The root and leaves (Null) are Black
3. If node is Red, then its parent is Black
4. All simple paths from any node x to a descendant leaf have the same number of black nodes
   = blackheight(x)

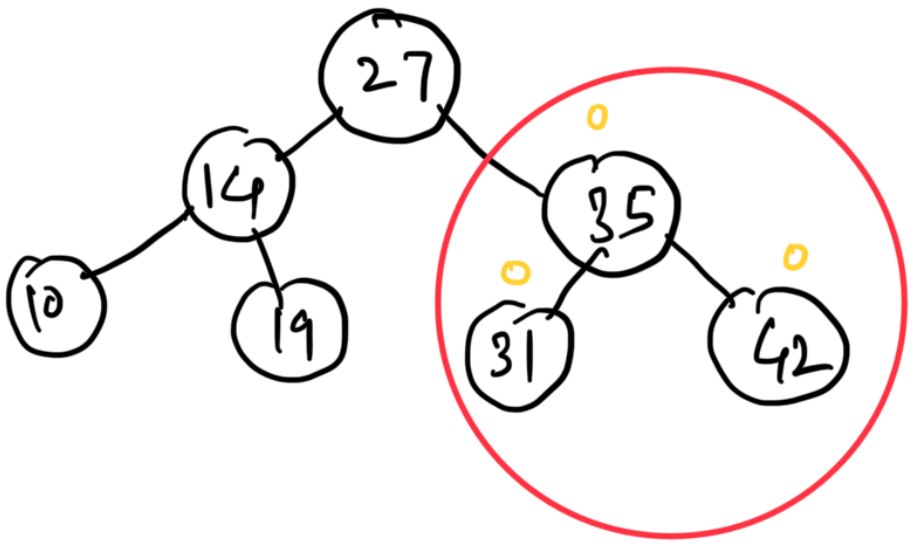
AVL Tree – Adelson-Velskii, Landis



Parameter for balancing → Balance factor

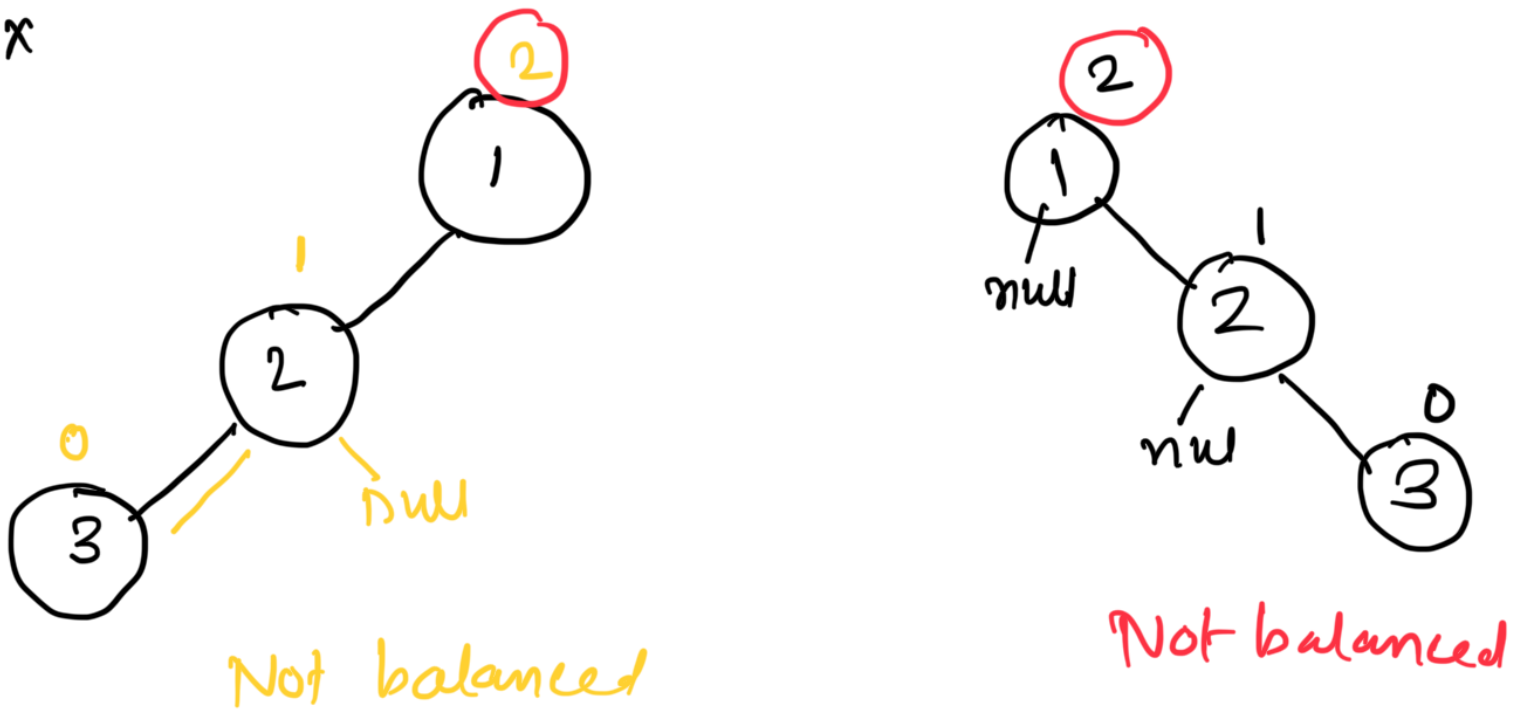
for balancing ⟵ Identifies the imbalance

Rotations ⟶ Impostation



$$\left[h(\text{left subtree}) - h(\text{Right subtree}) \right] \geq 1$$

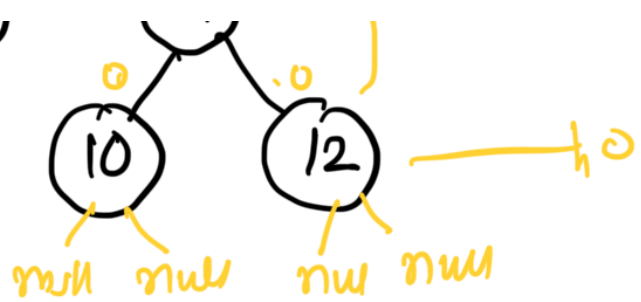
Ex



$$BF \leq \pm 1$$

$$BF = h(LS) - h(RS)$$

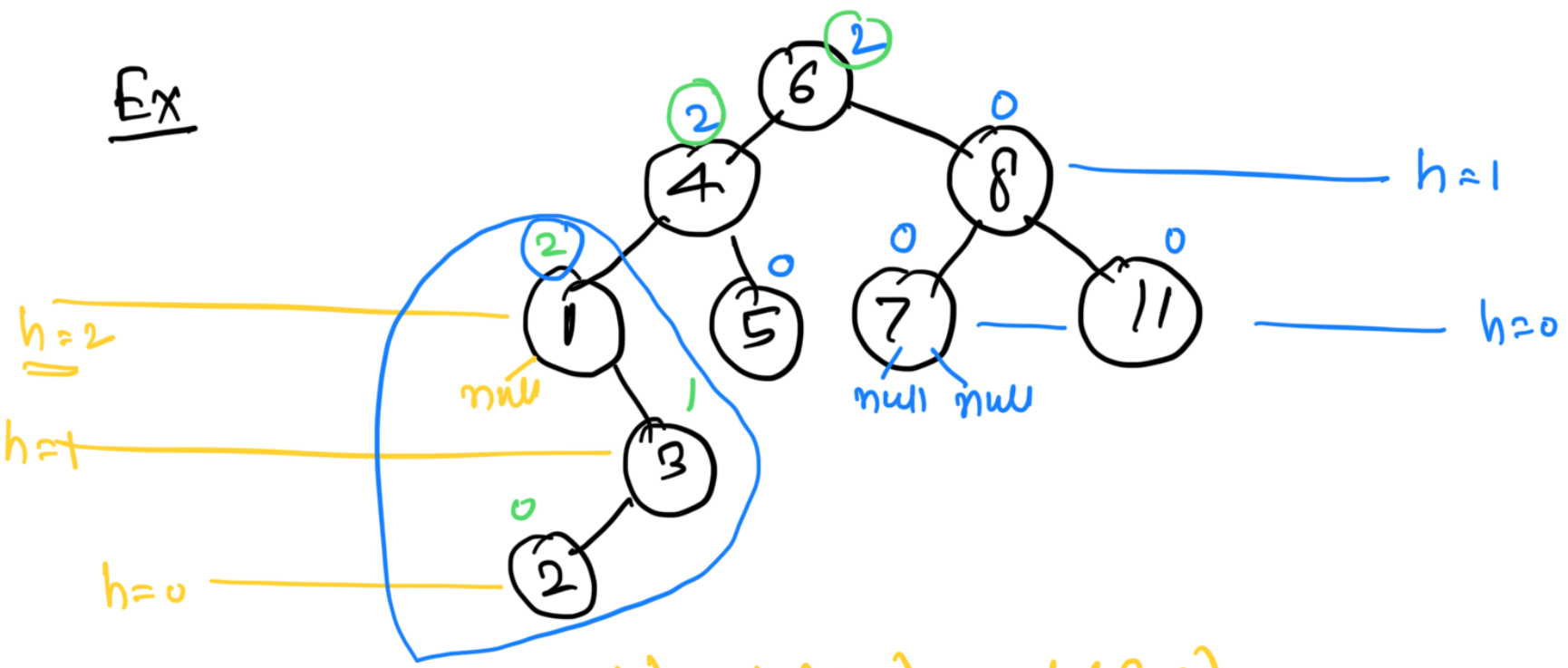
Ex



Balanced

h BF

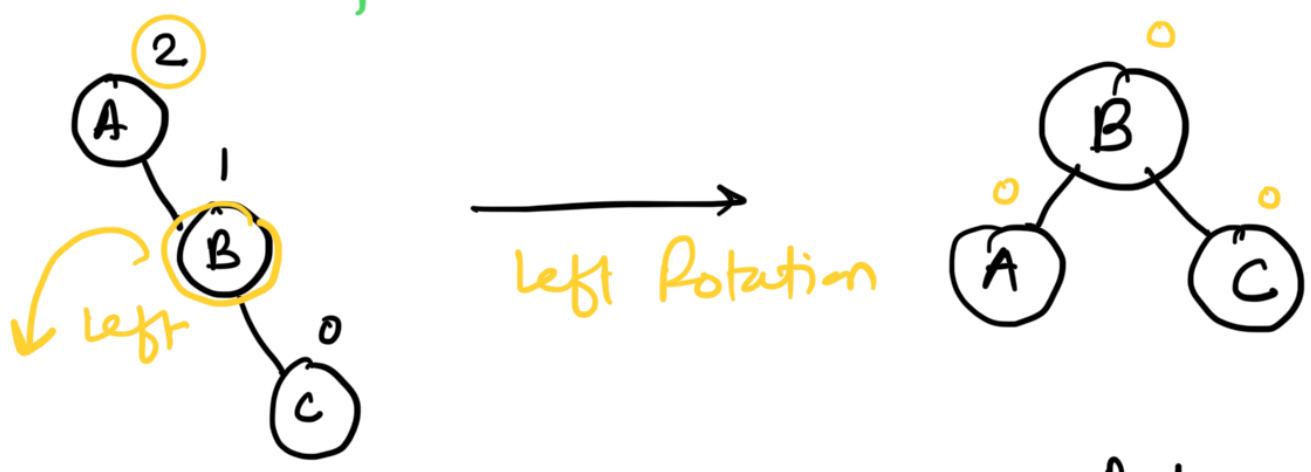
Ex



$$BF(1) = h(LS) - h(RS)$$
$$= 0 - 2$$
$$= -2$$
$$\approx 2$$

Balancing → Rotations →

Case 1

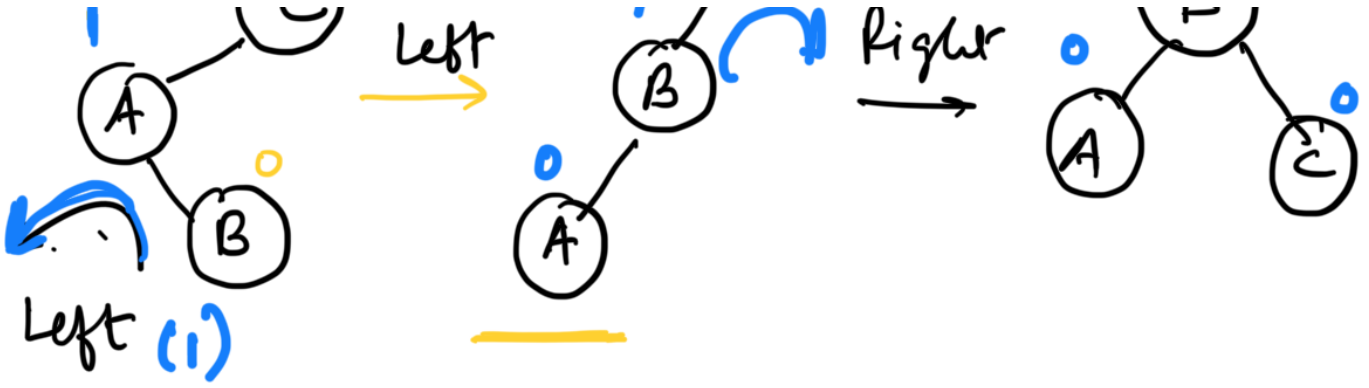


Unbalanced

Balanced

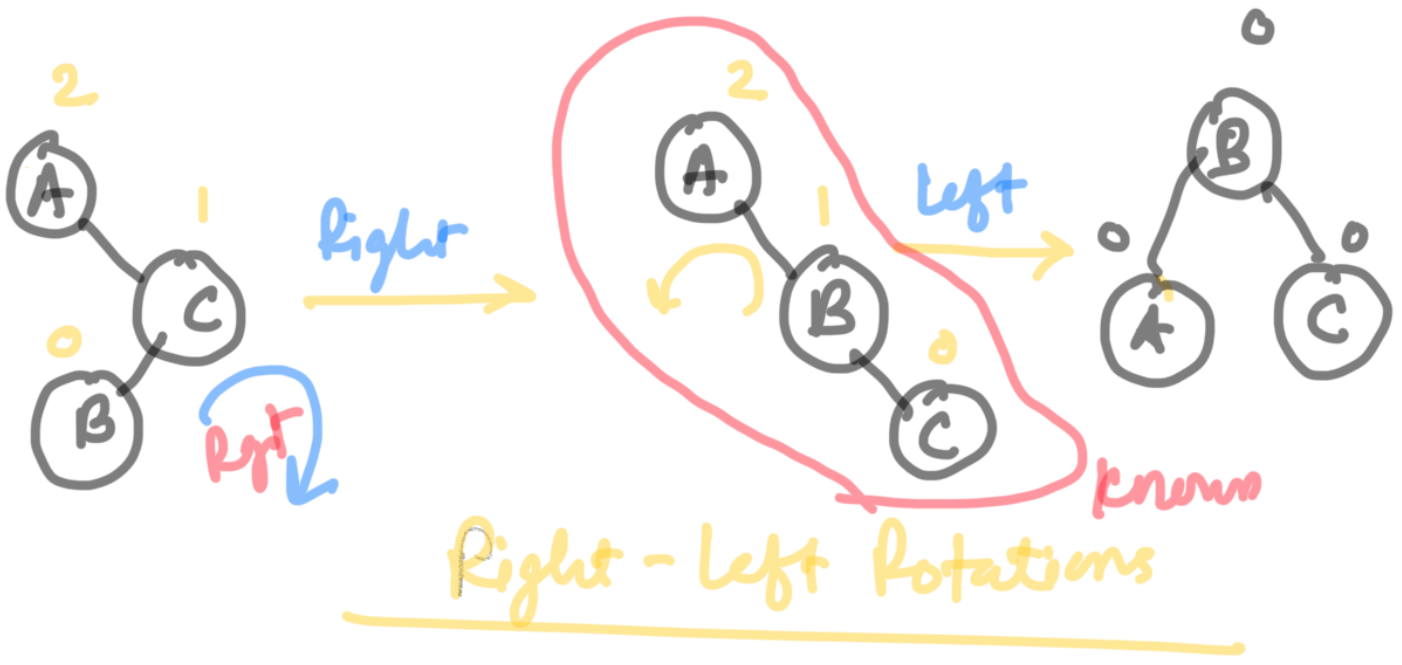
Case 2

Case 3



Left - Right Rotation

Case 4



Right - Left Rotations