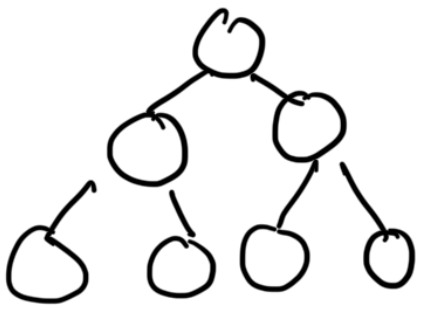
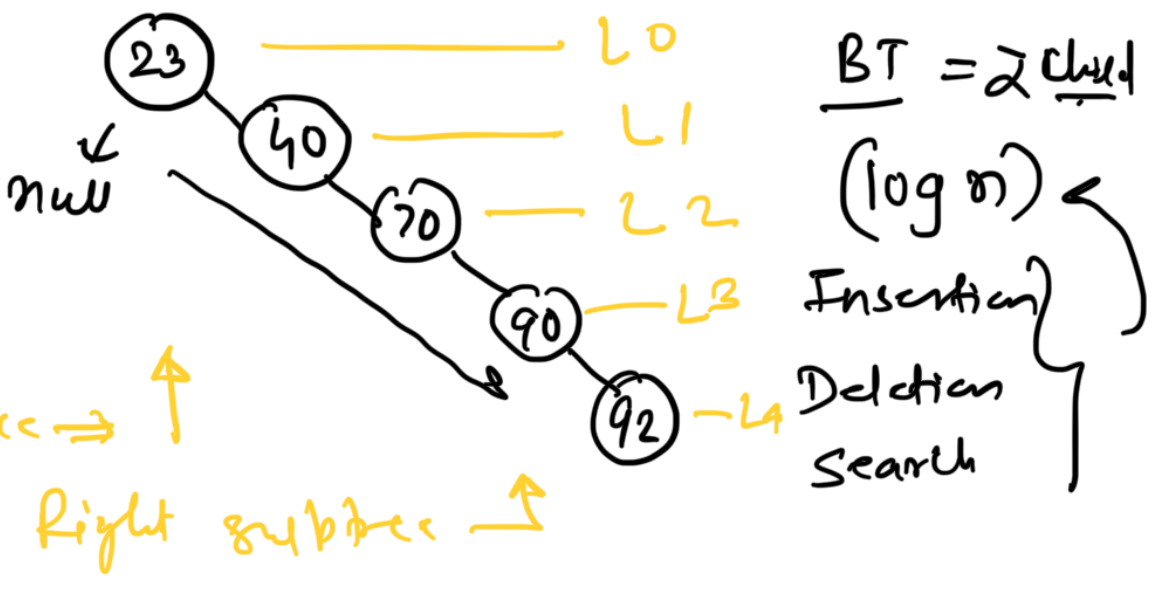
Balance Tree

- Balanced Search Tree
  - height Balance Tree





BT = 2 child

(log n)

Insertion

Deletion

Search

Need Balancing Tree ⇒ Height Balance

Ex: - 1. AVL Tree ✓

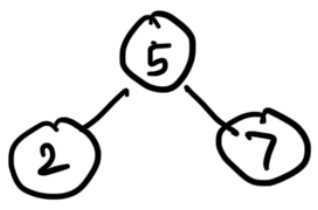
2. 2-3 Trees

3. 2-3-4 Trees

4. B-trees, B+ trees

5. Red-Black Tree ✓

AVL Tree -



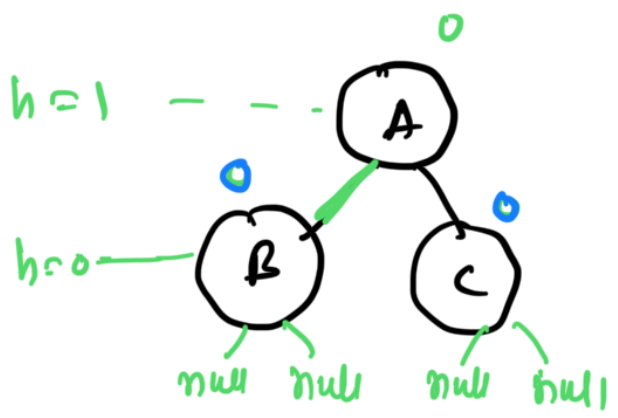
- Adelson - Velsky - Landis
  - AVL Tree
- height balance tree — search tree
- Balance factor = {0, ±1}
  = {-1, 0, +1}

formula

- Balance factor = height of left subtree − height of Right subtree

- Every BST is applicable to AVL

- Insertion, Deletion & Search ⇒ O(log n)



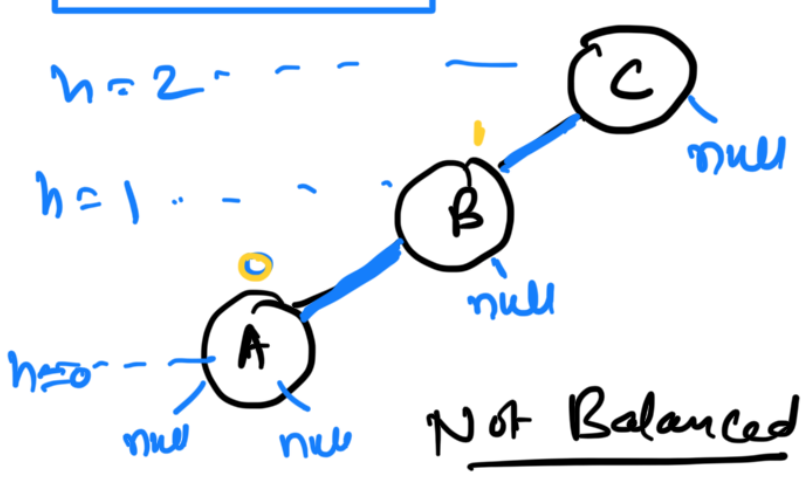
Nod(A) = h(LST) − h(RST)

= 1 − 1

= 0 ⟵

Balanced

Leaf node = 0



Not Balanced

BF(B) = h(LST) − h(RST)

= 1 − 0

= 1 ⟵ Balanced

BT(A) = h(LST) − h(RST)

= 0 − 0

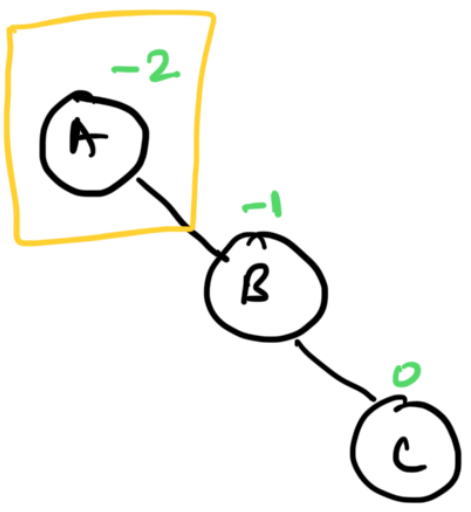
= 0

BF(C) = h(LST) − h(RST)

= 2 − 0

= 2 ⟵ Not Balanced

Ex:

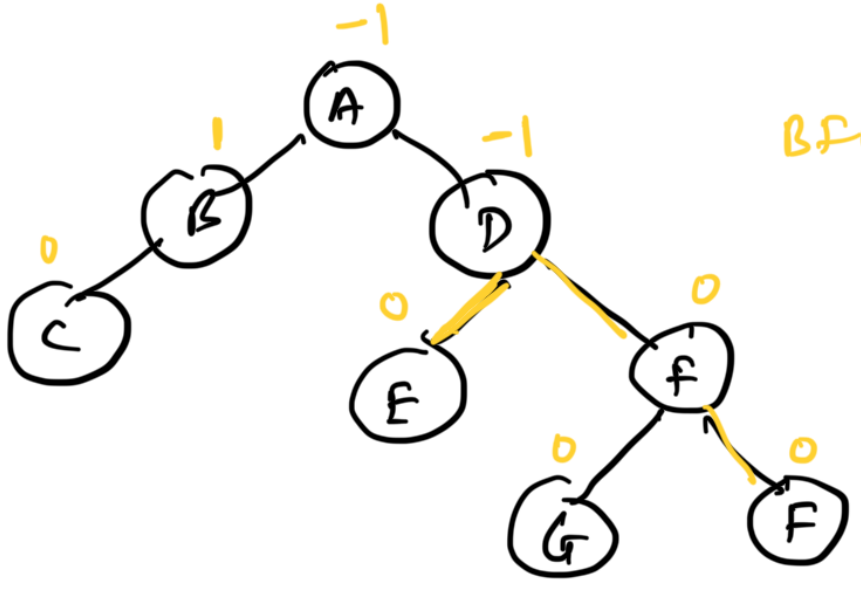


BF(B) = 0 − 1 = −1

BF(A) = 0 − 2

= −2

Ex:



Balanced

BF(D) = (1 − 2)

= −1

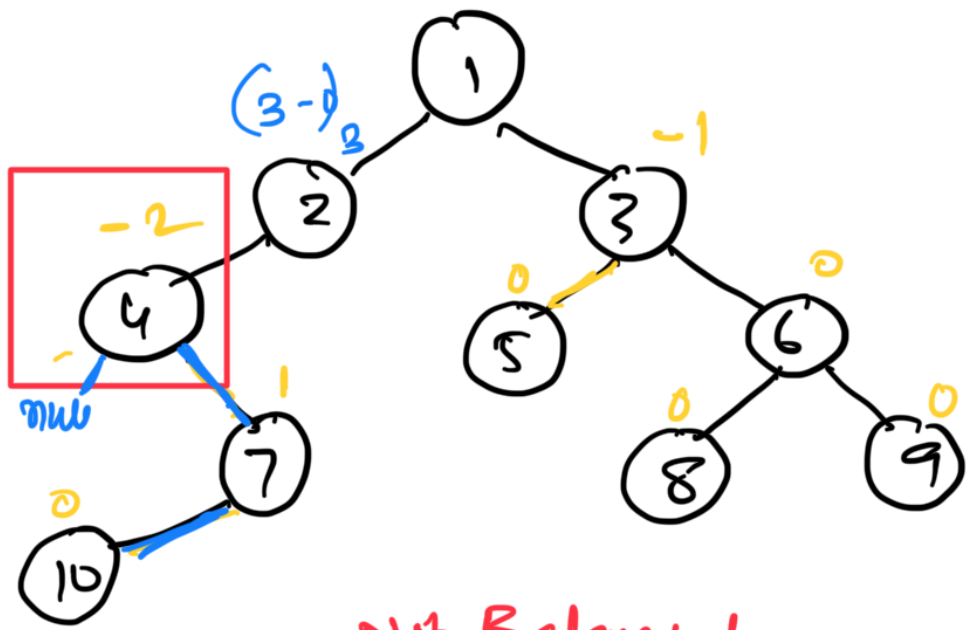
BF(B) = 1 − 0

= 1

BF(A) = 2 − 2

= 0

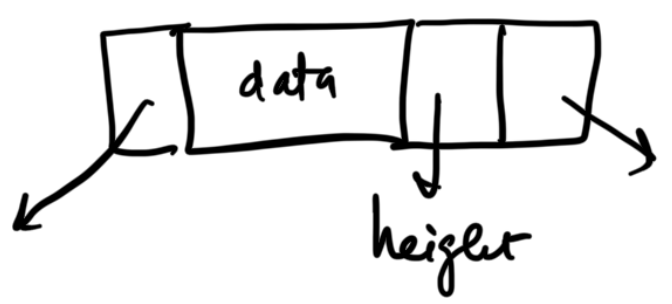
Ex:

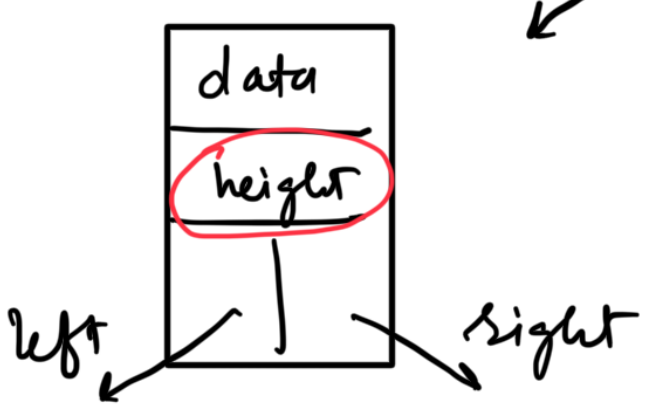


Not Balanced

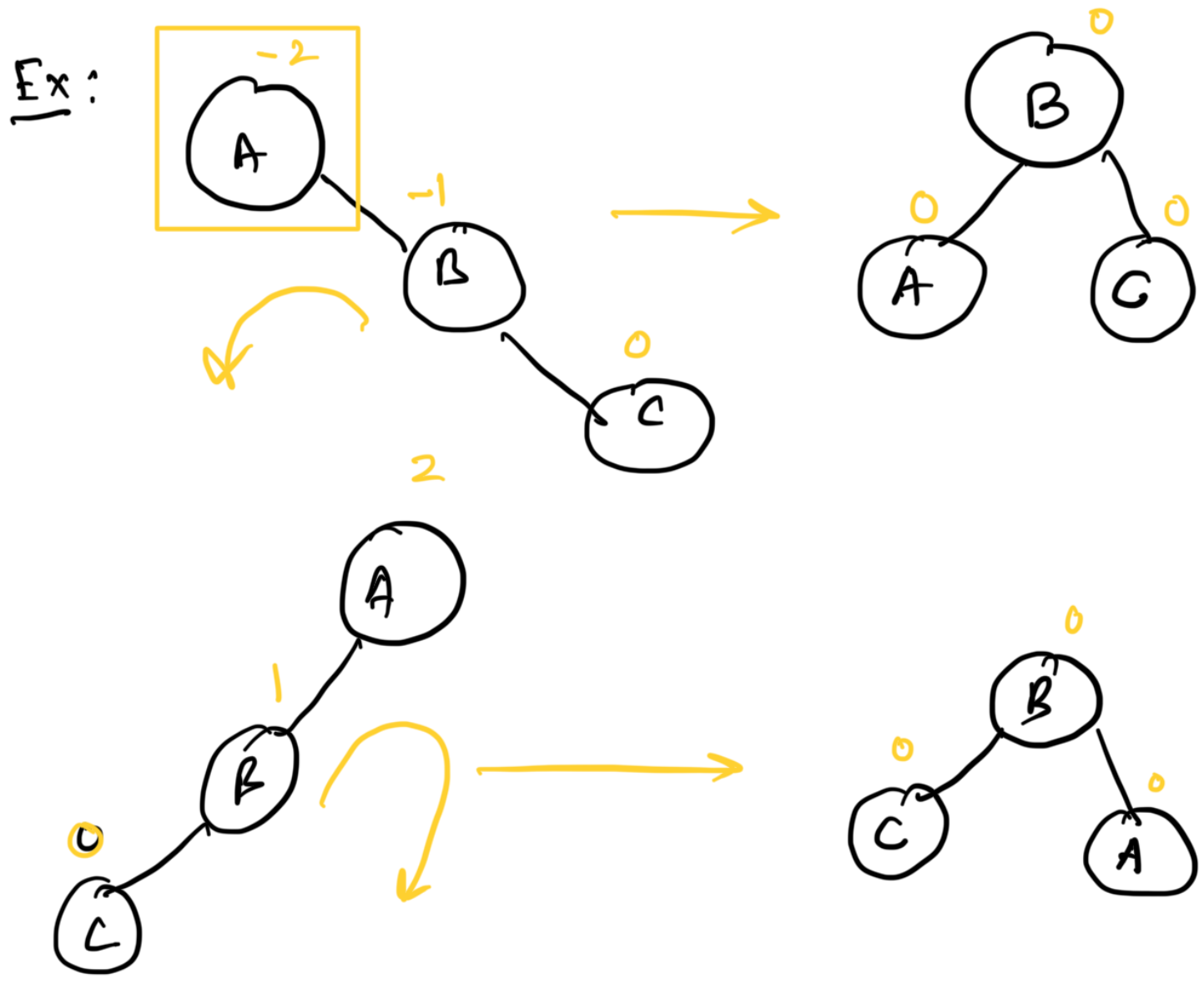
BF(4) = 0 - 2 = -2

AVL Tree





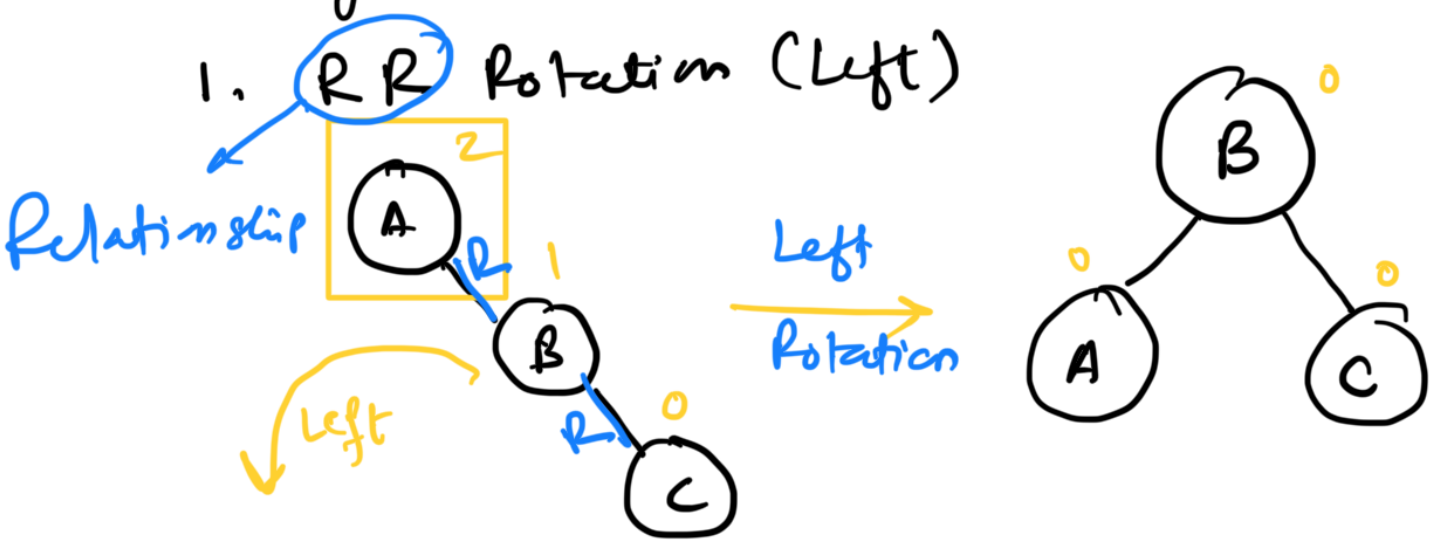
Ex:



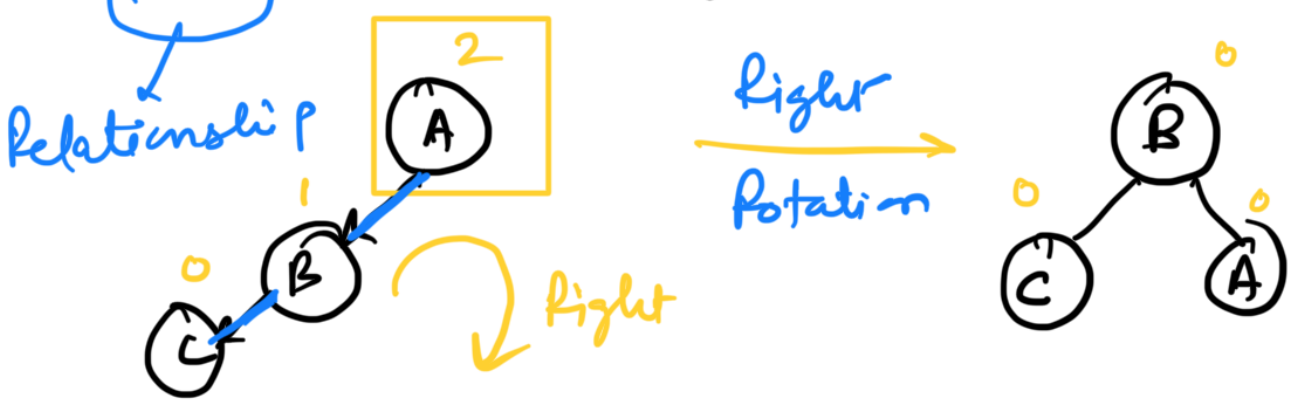
Types of Rotations

1. Single Rotation

1. RR Rotation (Left)

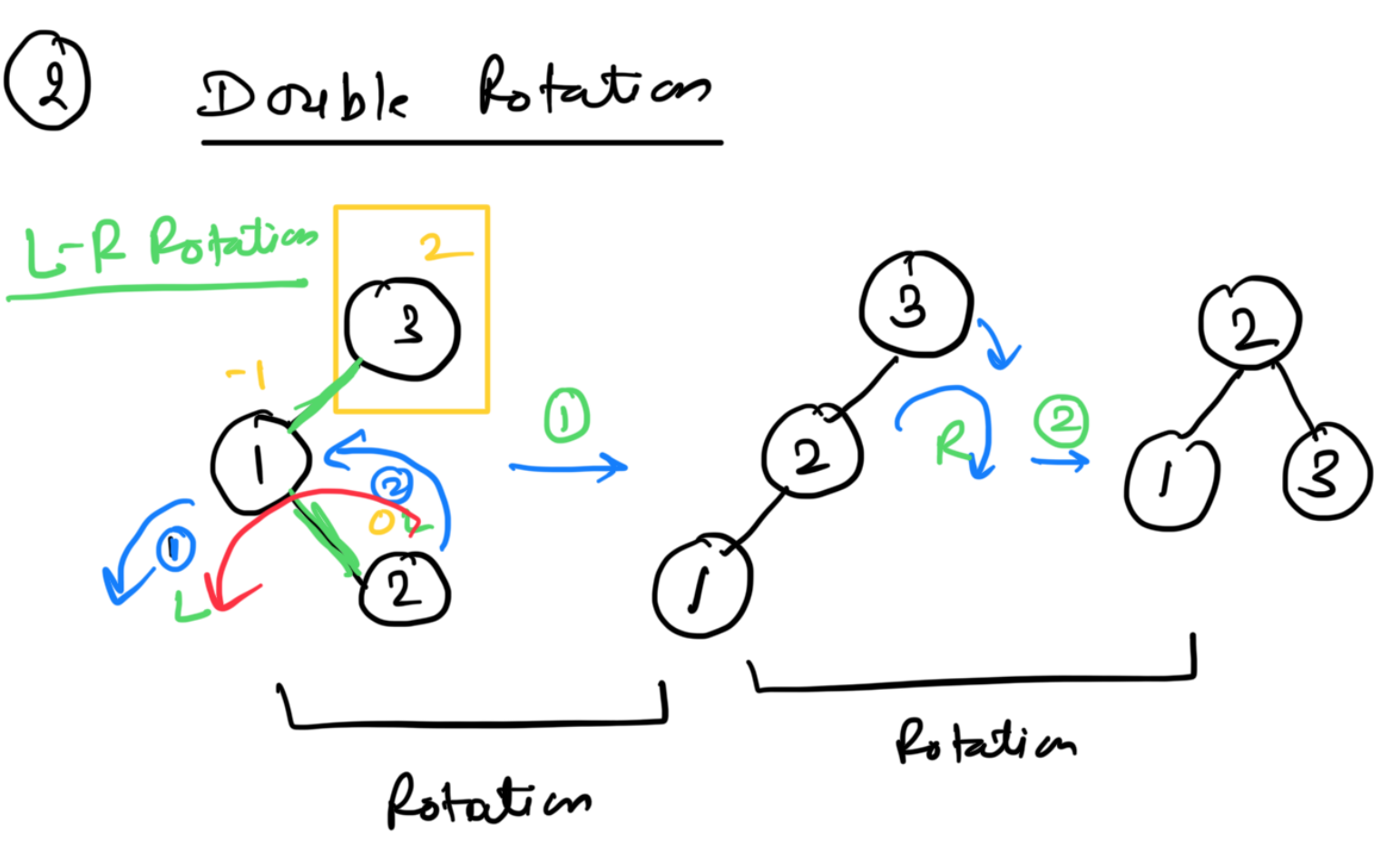


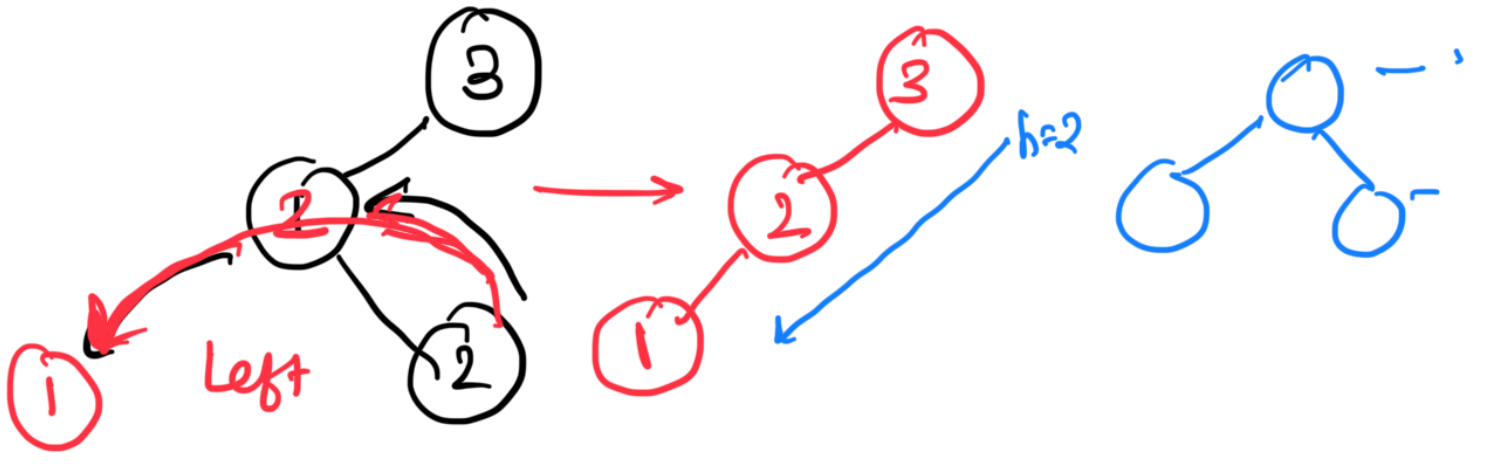
2. LL Rotation (Right)

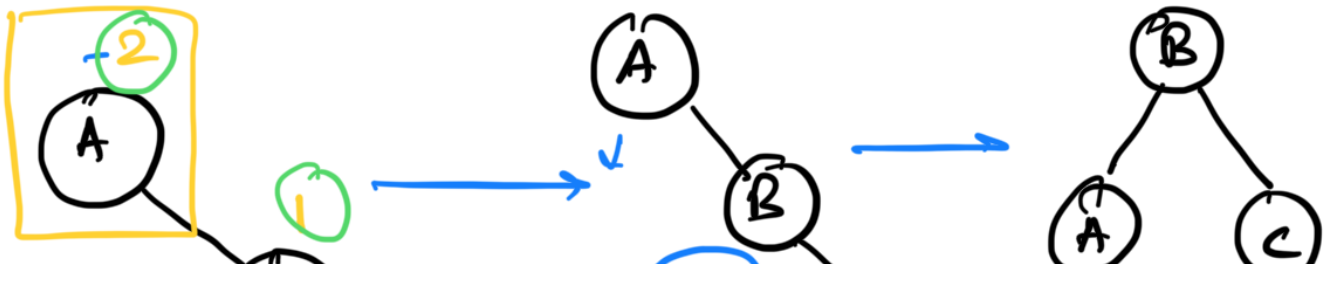


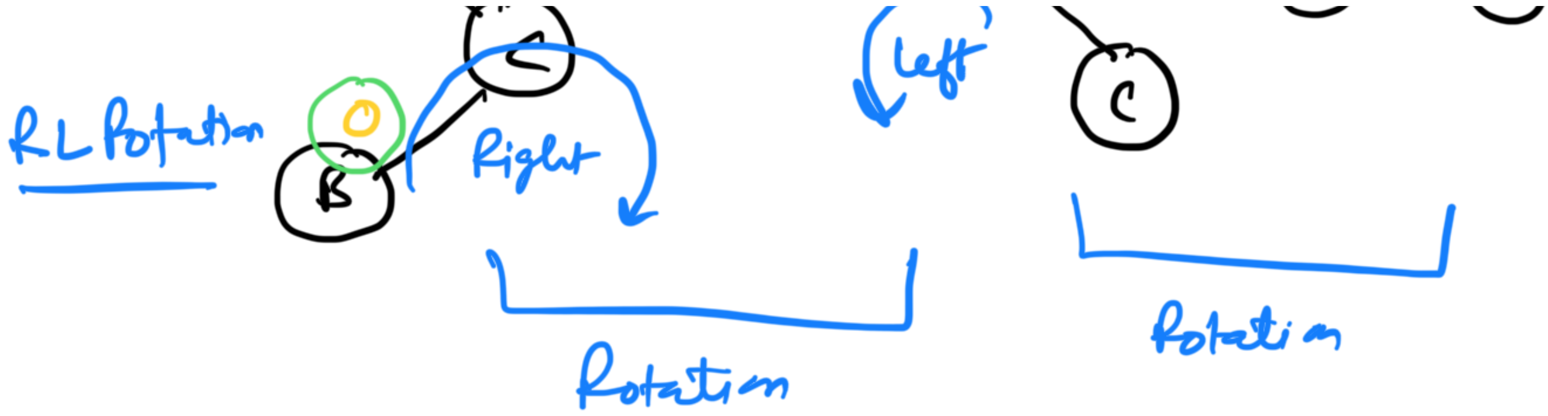
(2) Double Rotations

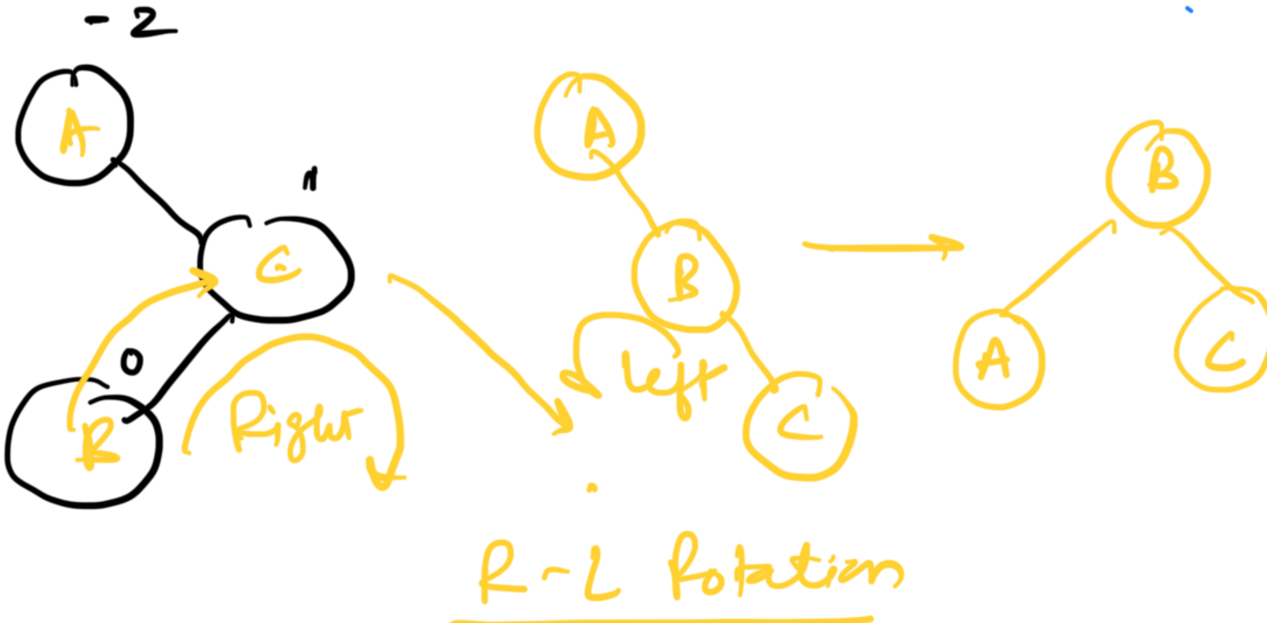
L-R Rotation

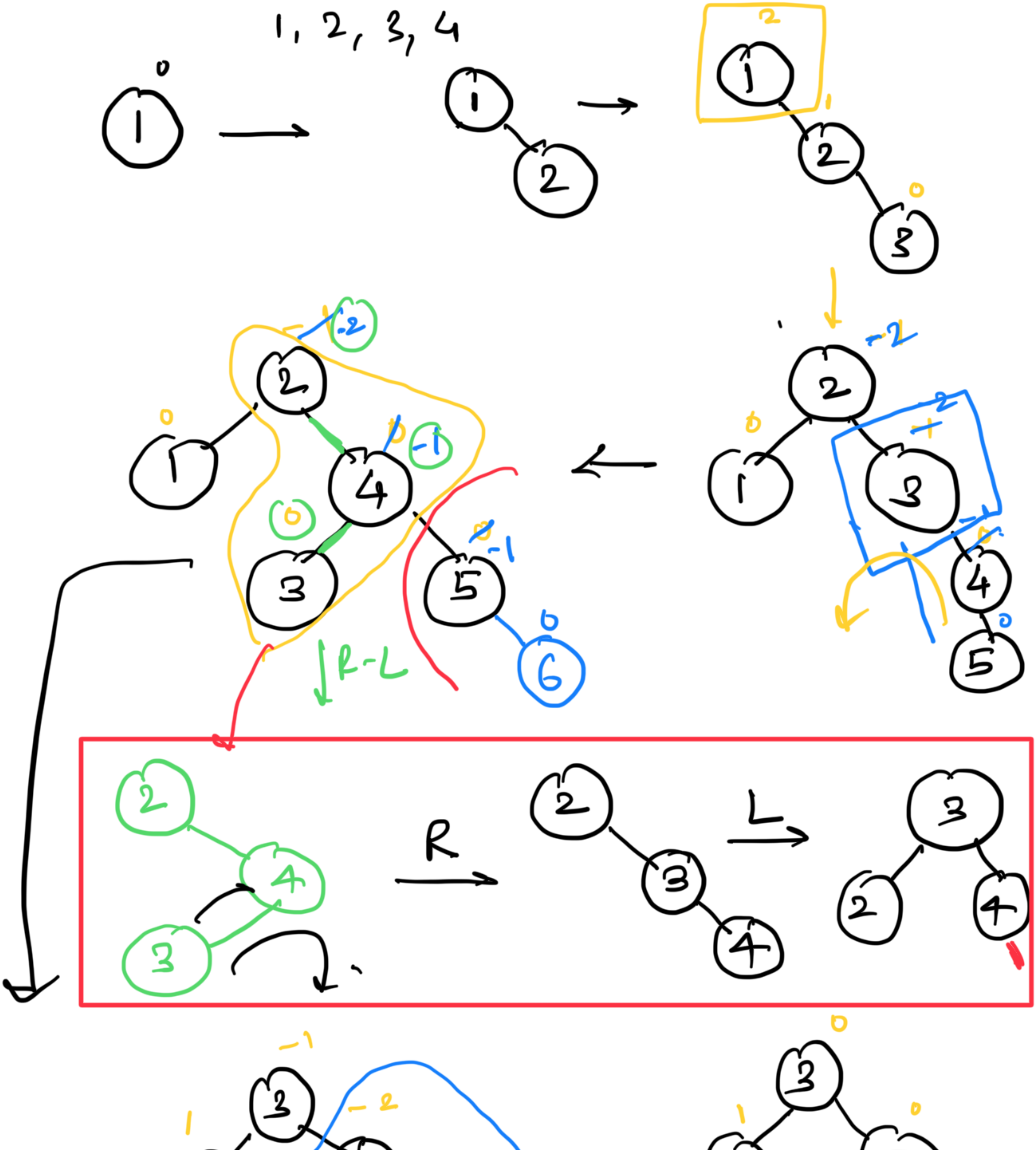


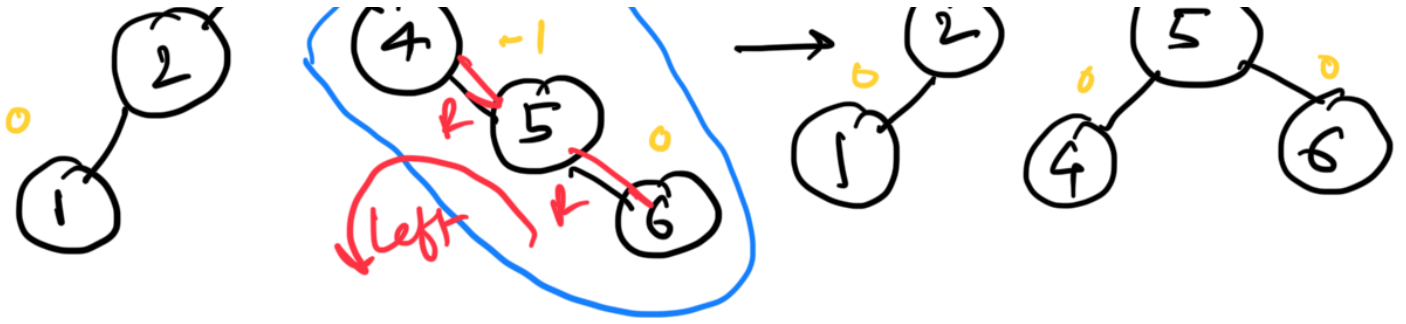





RL Rotation
Right
Left
Rotation
Rotation


-2
A
C
B
Right
Left
R-L Rotation


1, 2, 3, 4
R-L
R
L



AVL → BST → BT

{Rotation} { L < Root } {0,2}
2 Single R { R > Root }
2 Double R

Red-Black Tree → data structure

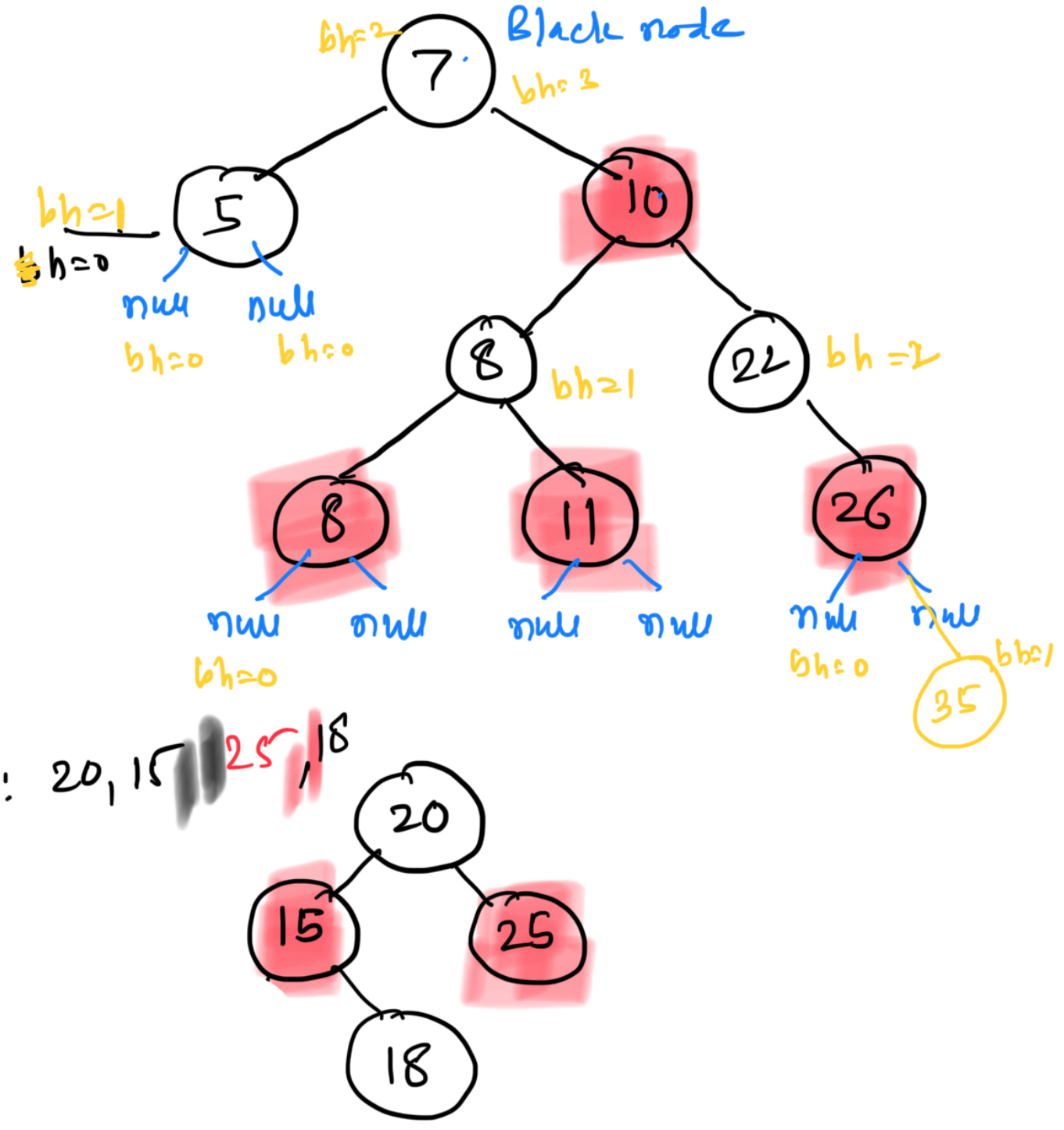
- BT - {0,2}
- BST - {LC < Root, RC > Root}
- Coloring property -
  - Every node in the tree is either red or black
- Balance property -
  - R & B tree ensures that no path is more than twice as long as any other path by maintaining coloring property.

Properties -

1. Every node is either red or black
2. The root and leaves (null nodes) are black nodes
3. If a node is red, then the parent is black

4. All simple paths of any node 'x' to the descendents leaf are as of same number of black nodes.

= black height (x)



Searching, sorting

Linear

Arrays → dynamic → Application

Stack } Basic → Appl^ns

Queue }

Circular queue }

Linked list

DLL

Non linear

Trees

- BT
- BST *
- AVL * → Rotation
- Red & Black

- Graph
  - Directed
    Undirected
  - Traversal
    DFS

Algorithms

- Recursion
- Backtracking
- Brute & force
- Divide & Conquer
- Greedy Algo (Min, Max)
  - Priority Queue ← Heap
- Dynamic Programming
- Time Complexity

Dfs

- Spanning Tree
  - MST
    - weighted MST
      - Kruskal
      - Prims
  - Shortest path
    - Dijakstra's Algo
    - floydwarsh
    - Bellmanford