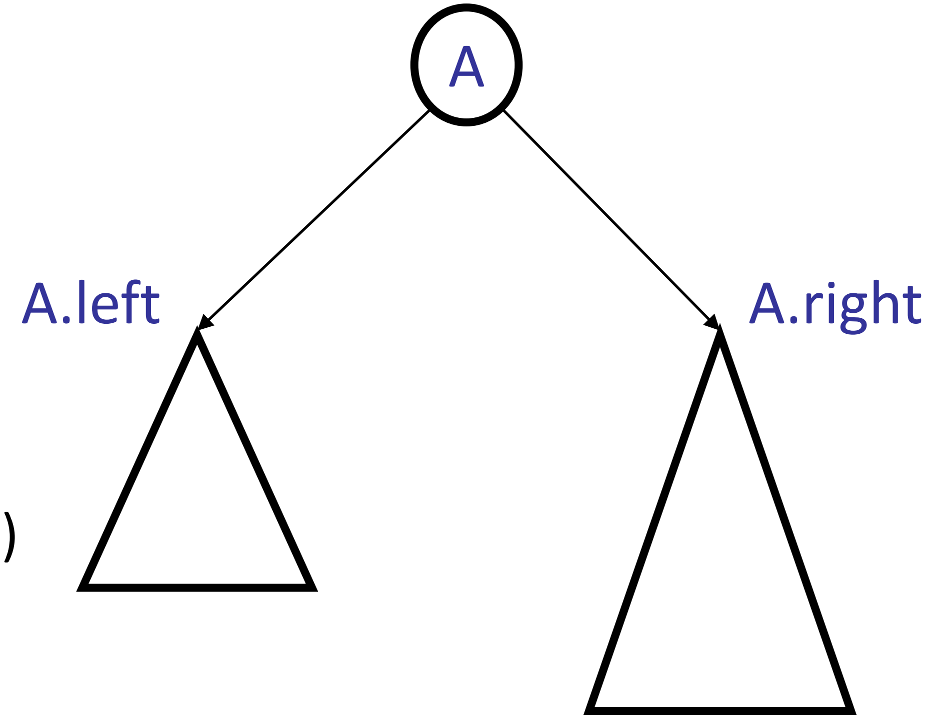
AVL Tree

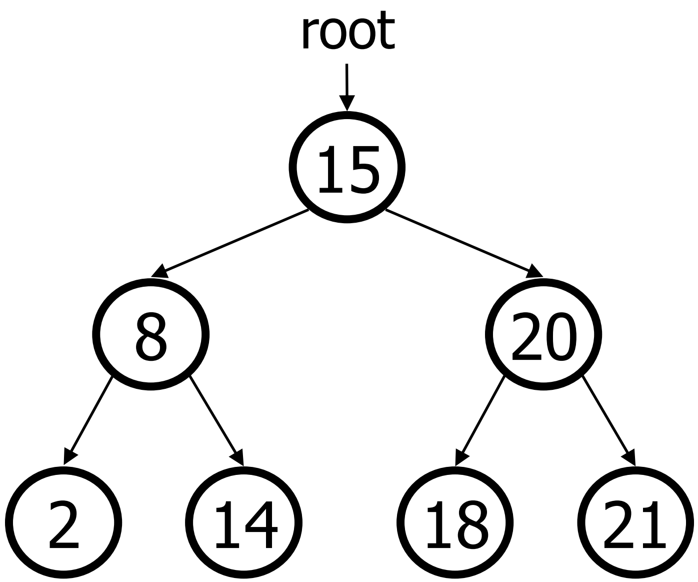
Height of a tree

- Max number of nodes in path from root to any leaf
  - Height(empty tree) = 0
  - height(a leaf) = ?
  - height(A) = ?
  - Hint: it's recursive!
  - height(a leaf) = 1
  - height(A) = 1 + max(
    height(A.left), height(A.right))



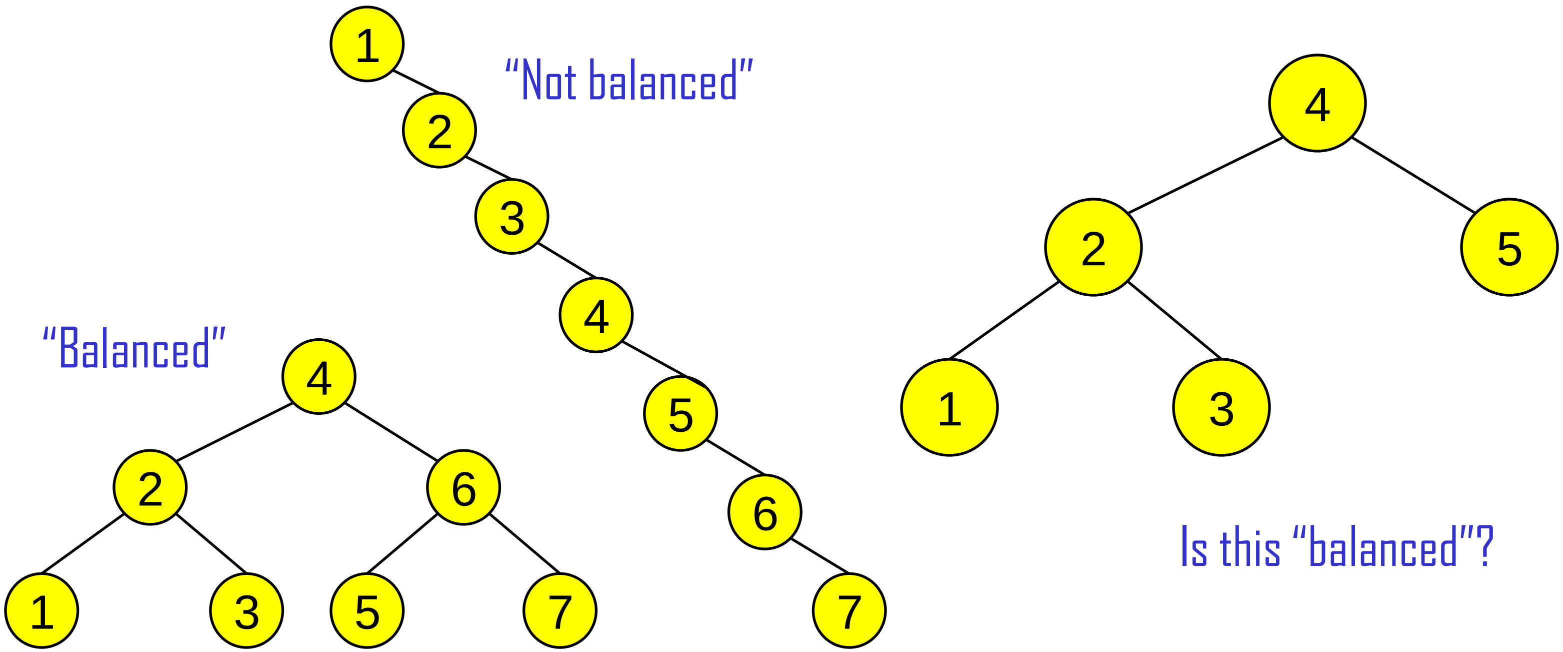
Some height numbers

- For binary tree of height h:
  - max # of leaves: 2^{h-1}
  - max # of nodes: $2^h - 1$
  - min # of leaves: 1
  - min # of nodes: h



Trees and balance

- Disadvantage(BST) : height can be N-1
- Worst Case insertion and deletion : O(N) time
- We want a tree with small height



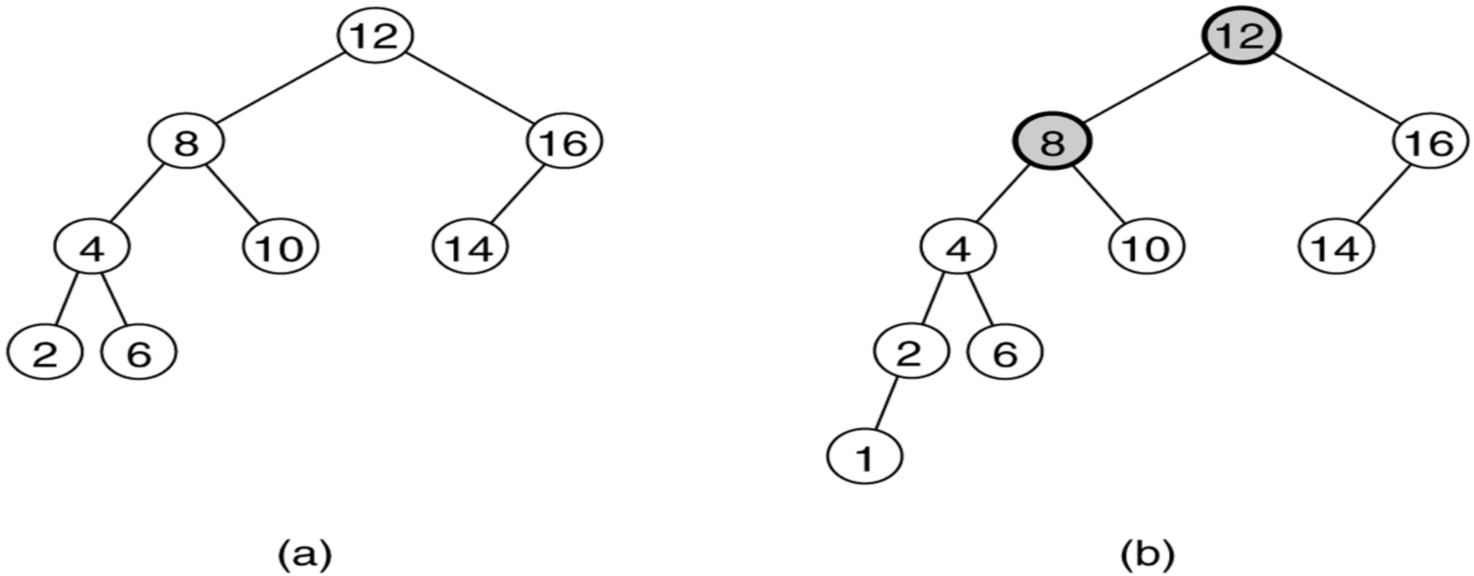
binary tree is said to be balanced if for every node, height of its children differ by at most one

AVL trees - Adelson-Velsky & Landis

- AVL tree is a self balanced binary search tree.
- **binary search tree** + **balanced tree**
- Balanced binary tree : Difference between the heights of left and right subtrees of every node in the tree is either **-1, 0 or +1**.
- Every node maintains a extra information known as **balance factor**.
- 1962 - Introduced by **Adelson-Velsky** and **Landis**.

AVL tree examples

- Two binary search trees:
  - (a) an AVL tree
  - (b) not an AVL tree (unbalanced nodes are darkened)

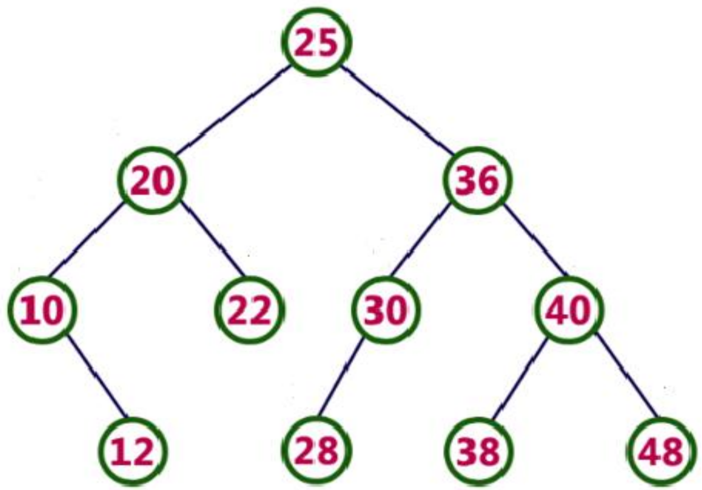


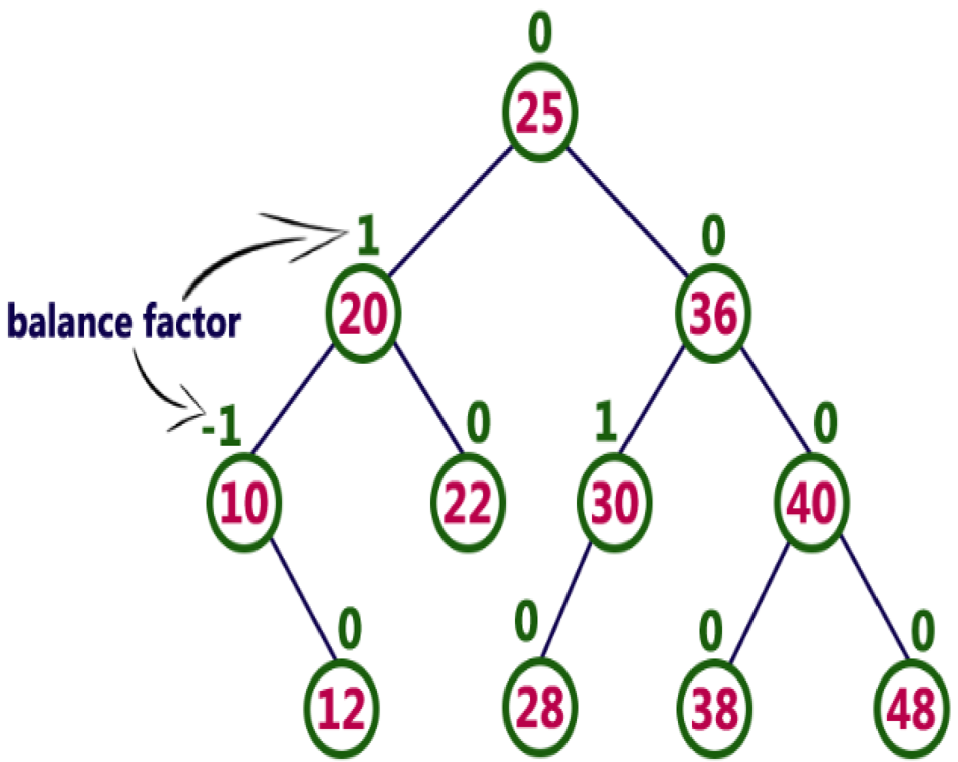
An AVL tree is a balanced binary search tree with balance factor of every node is either -1, 0 or +1.

Balance factor = Height of Left Subtree – Height of Right Subtree

AVL trees

Every AVL Tree is a binary search tree but all the Binary Search Trees need not to be AVL trees



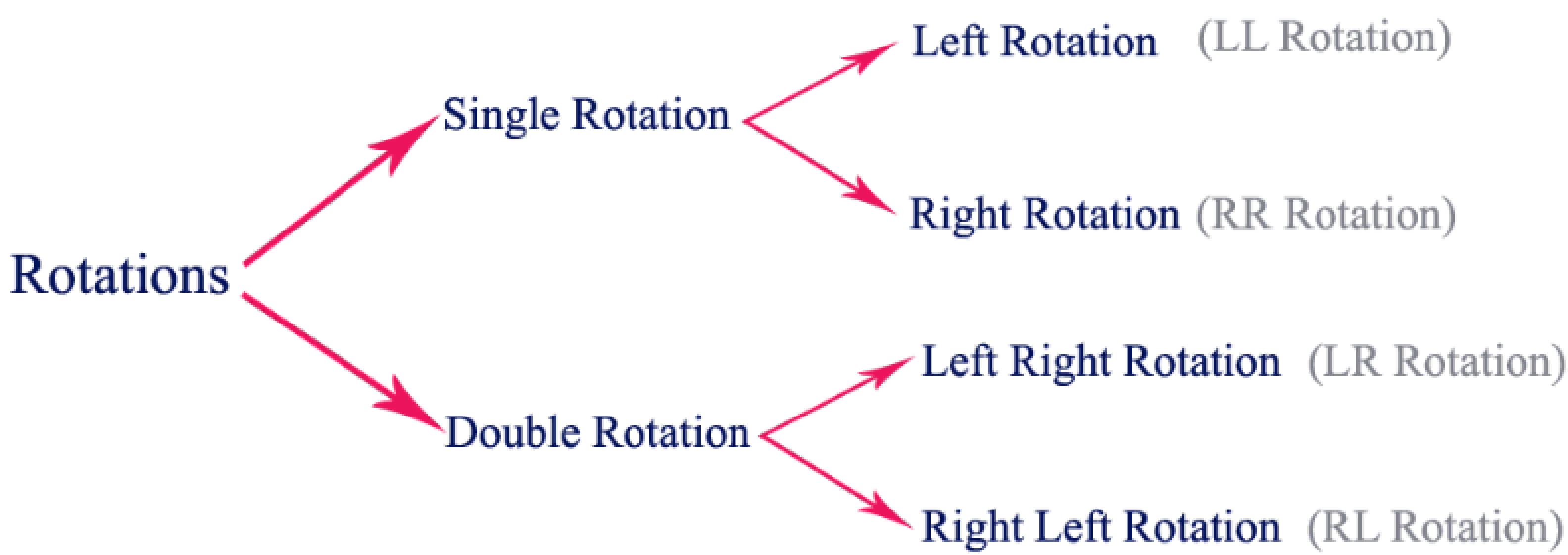


AVL Tree Rotations

- After performing every operation like insertion and deletion we need to check the balance factor of every node in the tree.
- If every node satisfies the balance factor condition then we conclude the operation otherwise we must make it balanced.
- Rotation operations are used to make a tree balanced.

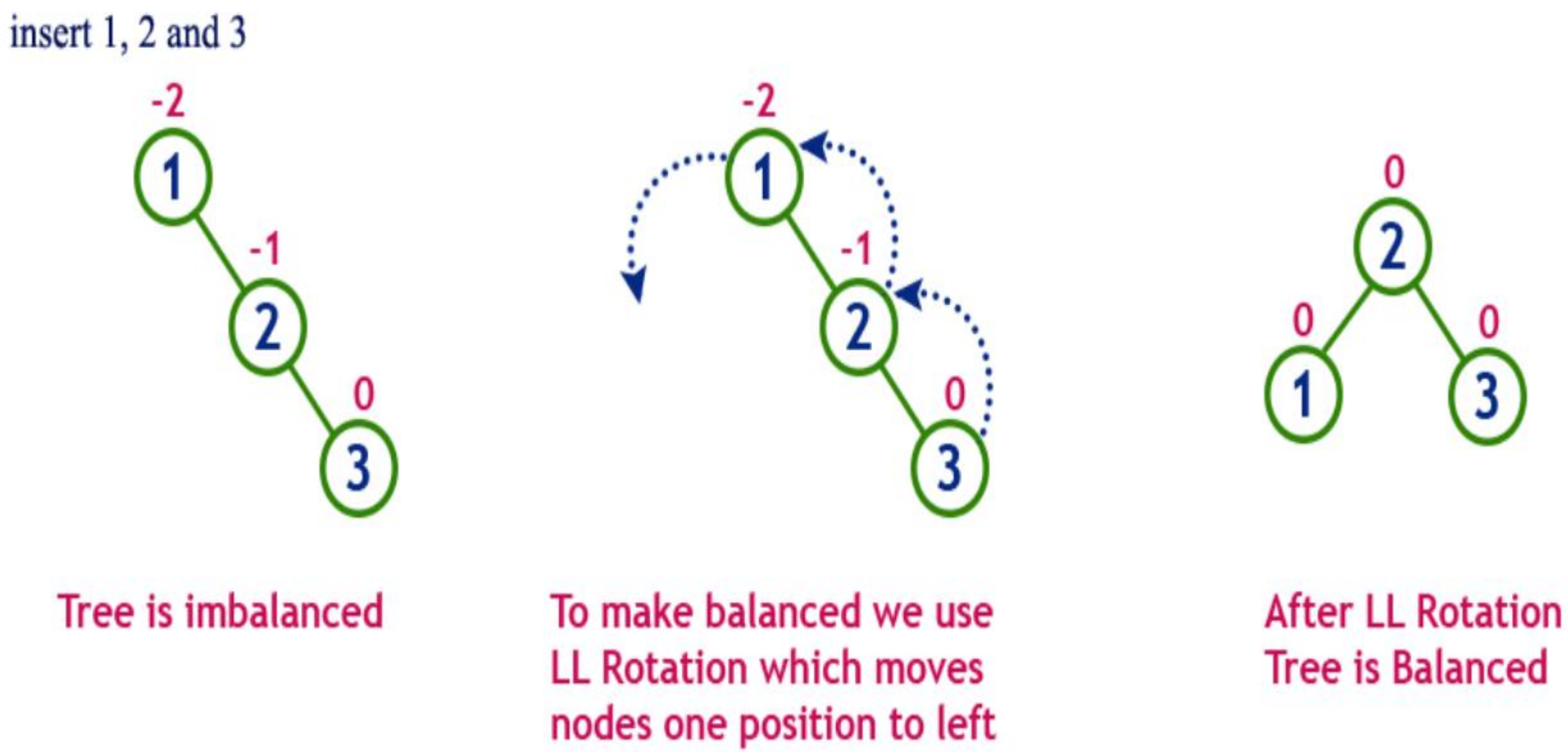
AVL Rotations

Rotation is the process of moving the nodes to either left or right to make tree balanced.



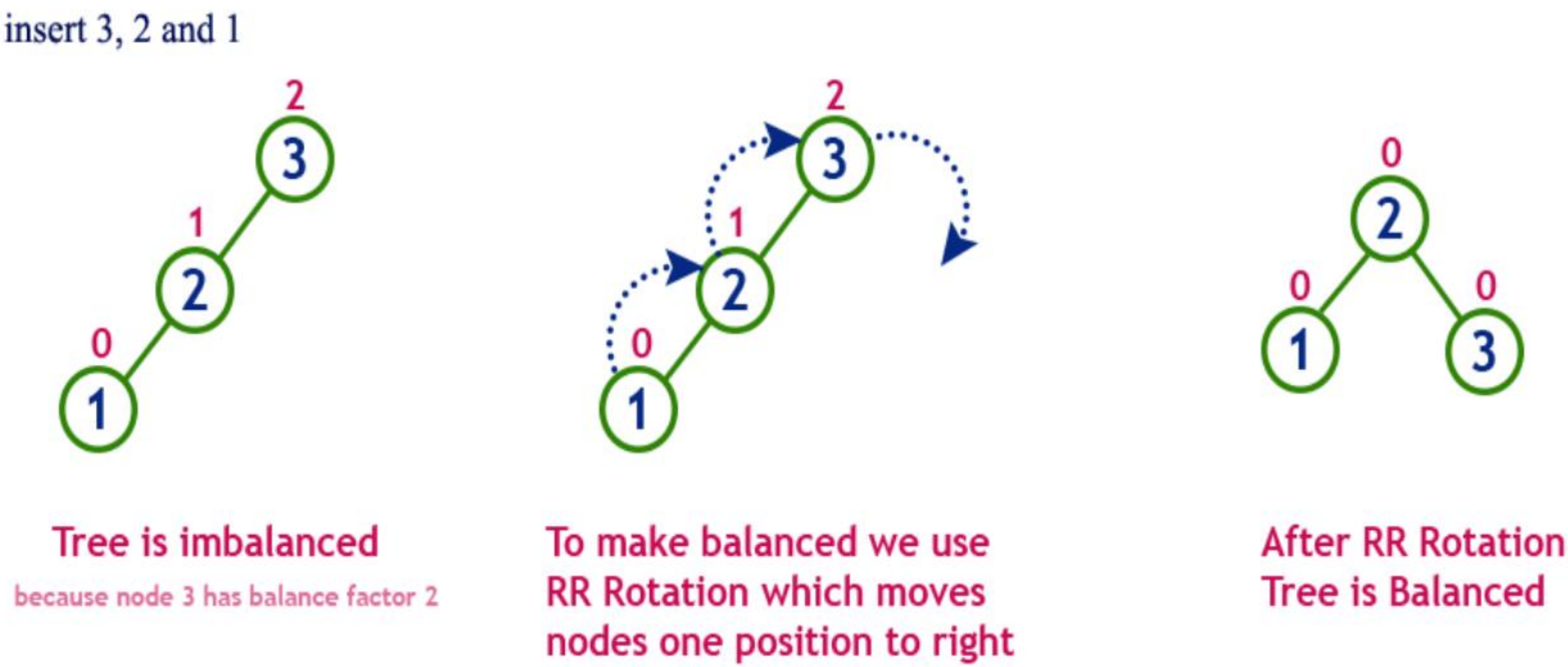
Single Left Rotation (LL Rotation)

Every node moves one position to left from the current position.



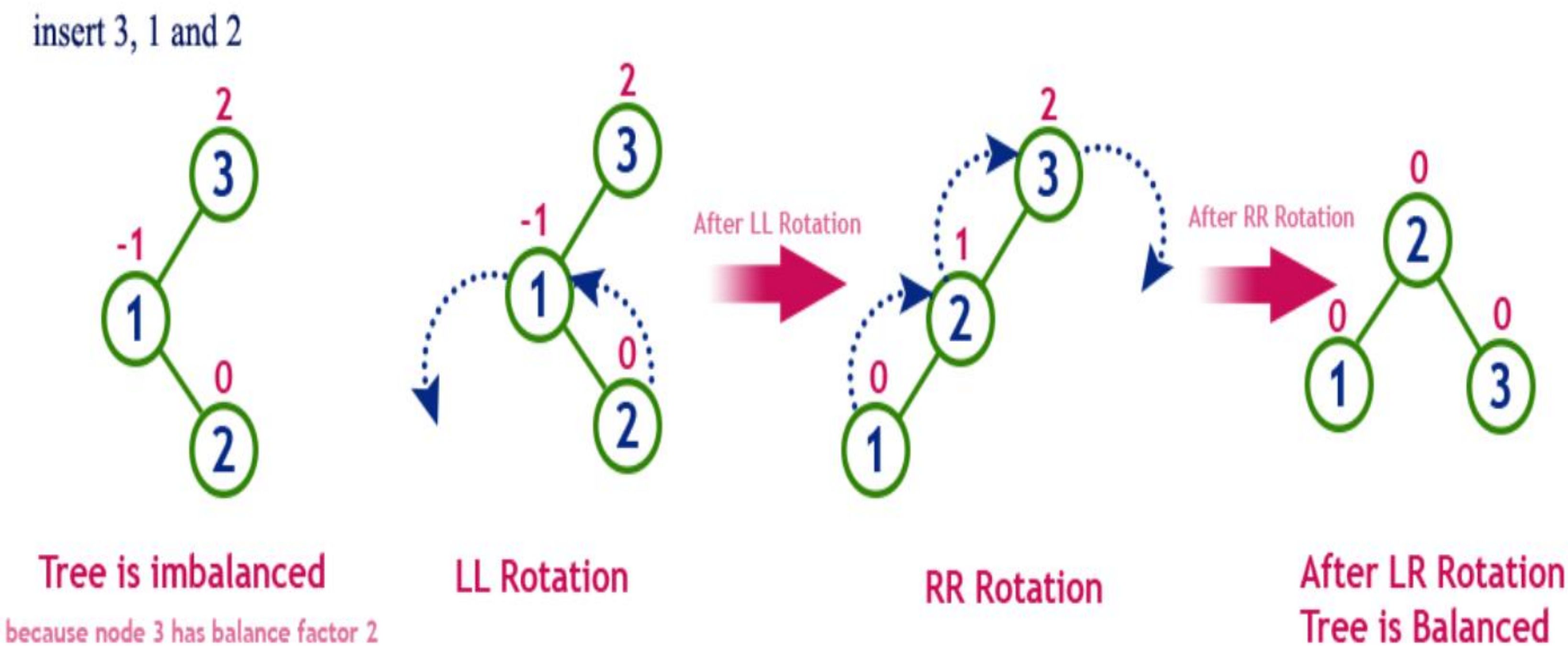
Single Right Rotation (RR Rotation)

Every node moves one position to right from the current position



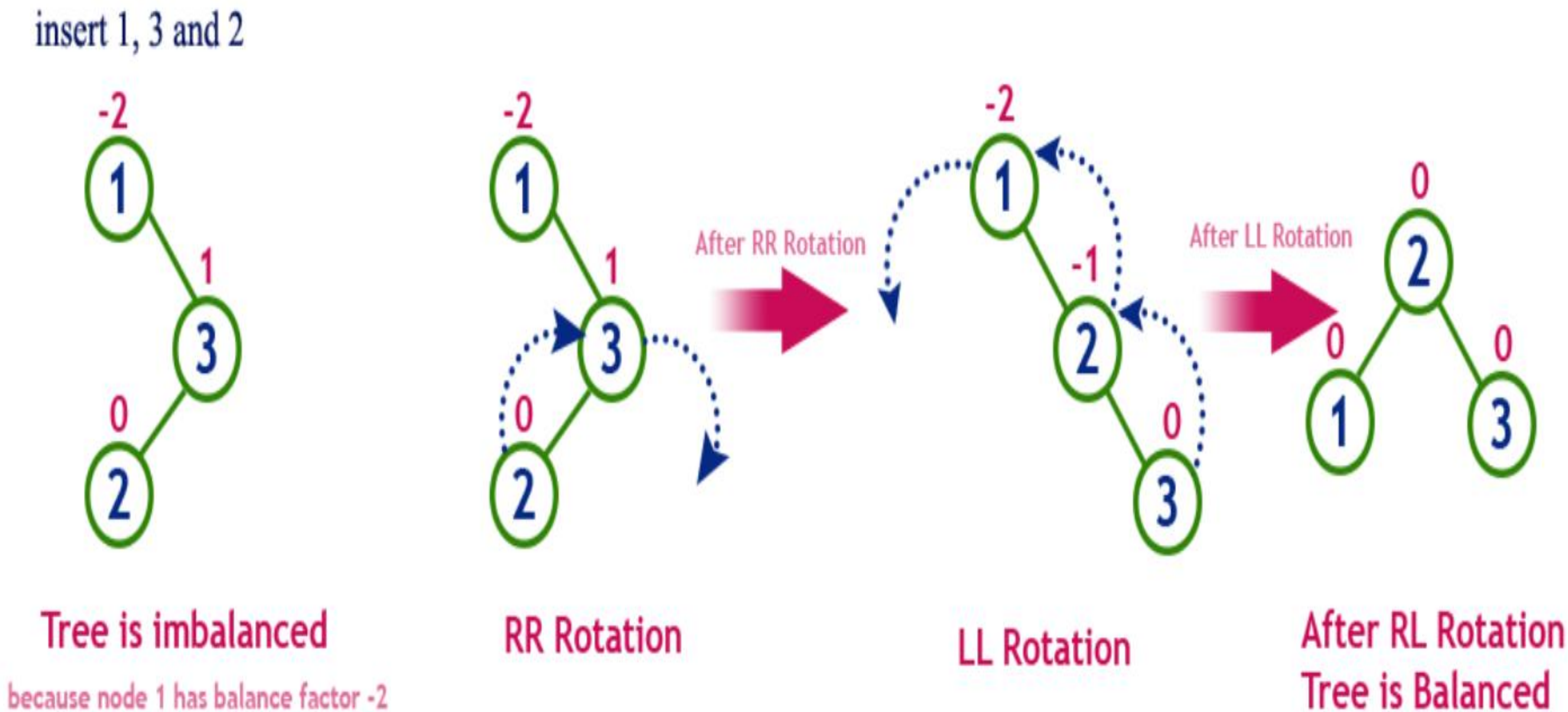
Left Right Rotation (LR Rotation)

- Combination of single left rotation followed by single right rotation.
- First every node moves one position to left then one position to right from the current position



Right Left Rotation (RL Rotation)

- Combination of single right rotation followed by single left rotation.
- First every node moves one position to right then one position to left from the current position



Operations on an AVL Tree

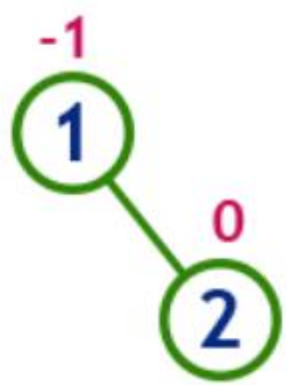
- Insertion/Deletion
  - **Step 1:** Insert/delete new element into tree using BST insertion
  - **Step 2:** check the **Balance Factor** of every node
  - **Step 3:** If the **Balance Factor** of every node is more then step4
  - **Step 4:** Perform the suitable **Rotation** to make it balanced.

Insertion

insert 1

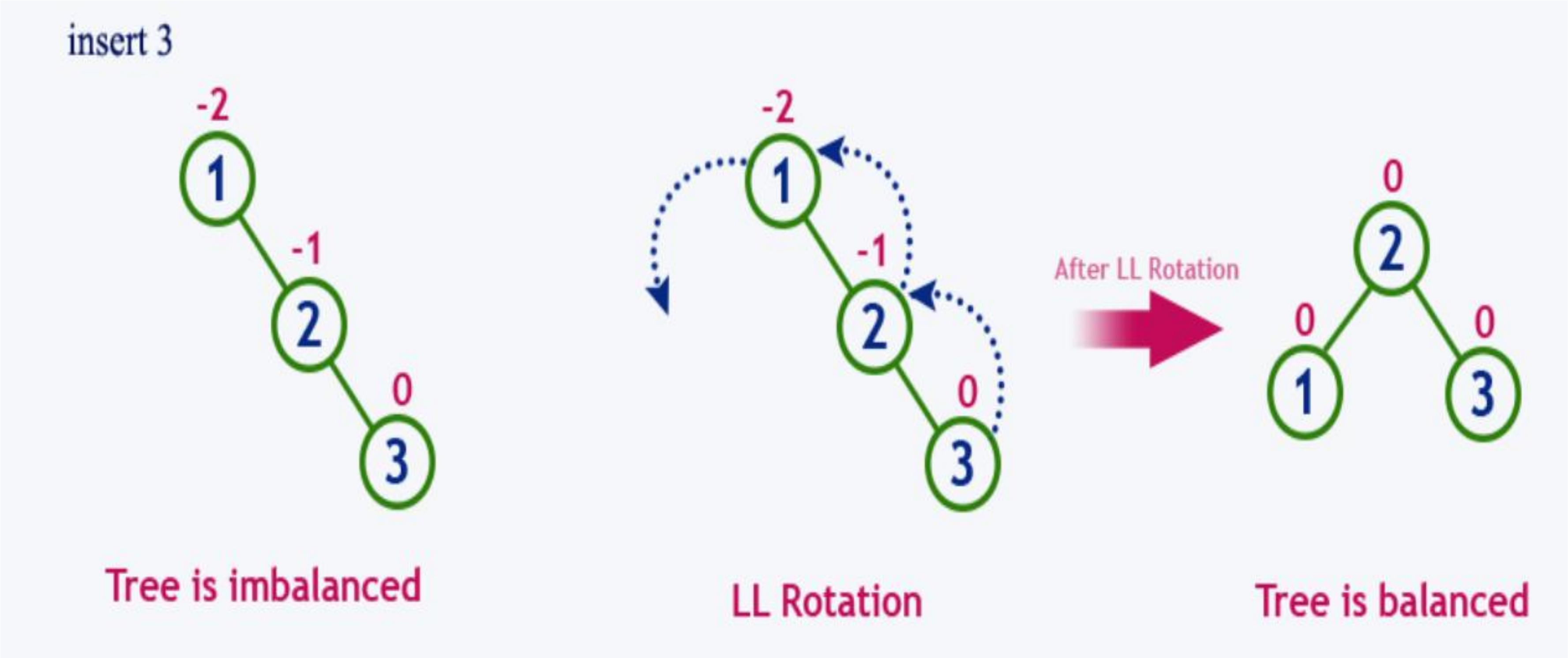
Tree is balanced

insert 2

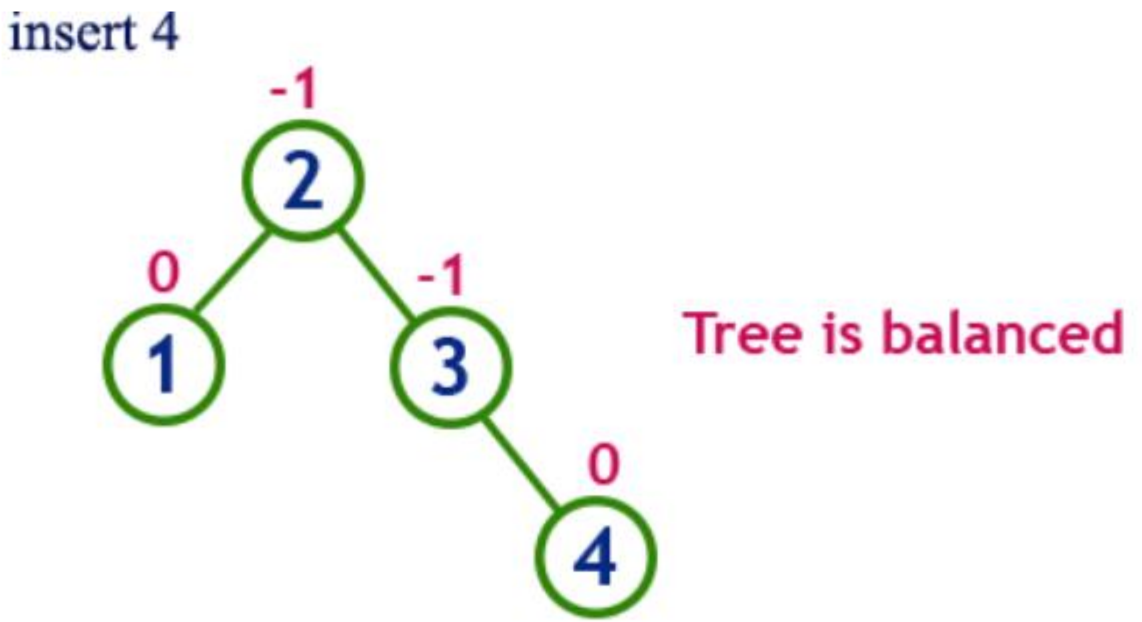


Tree is balanced

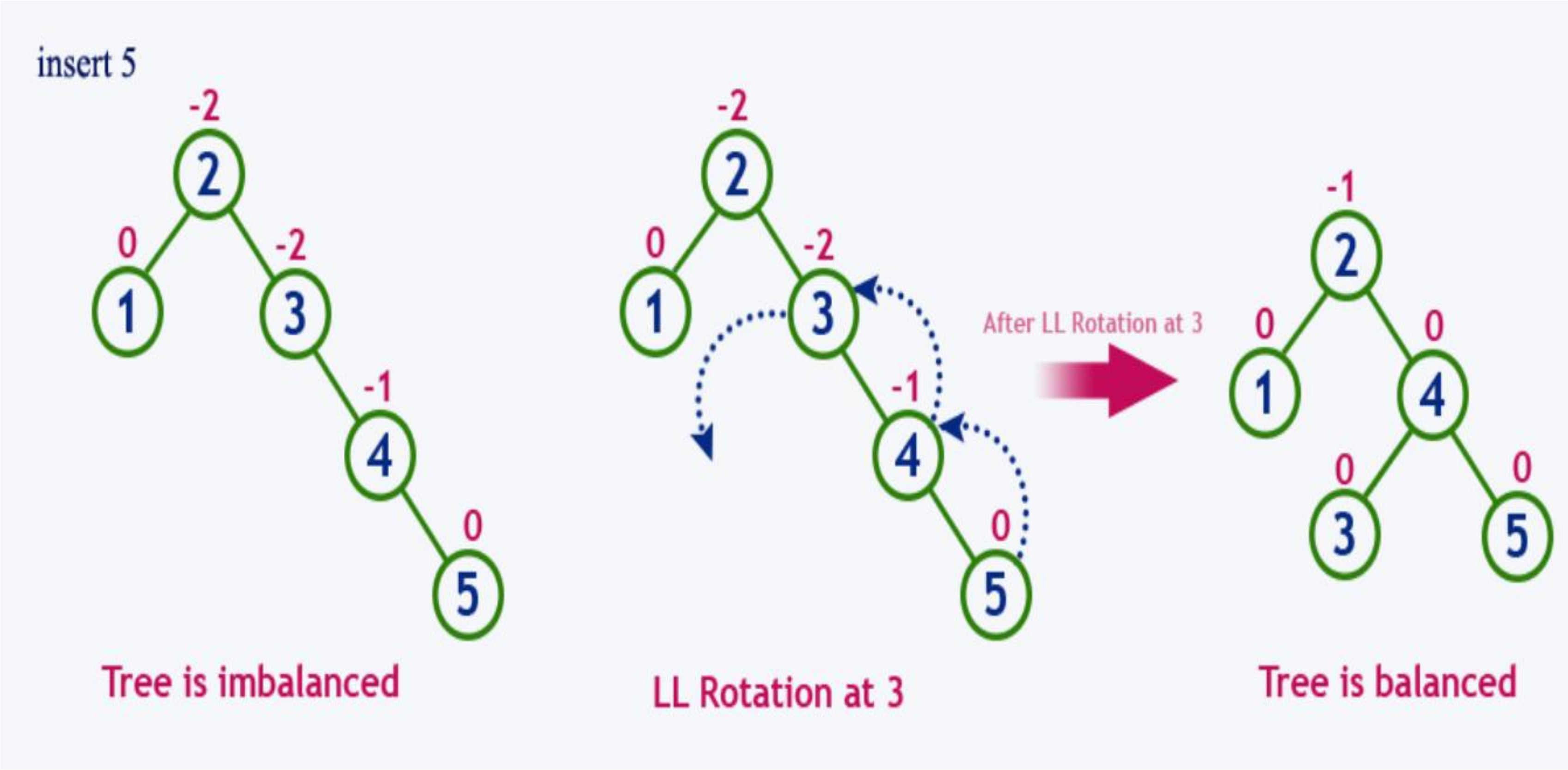
Insertion


insert 3
-2
1
-1
2
0
3
Tree is imbalanced
-2
1
-1
2
0
3
LL Rotation
After LL Rotation
0
2
0
1
0
3
Tree is balanced

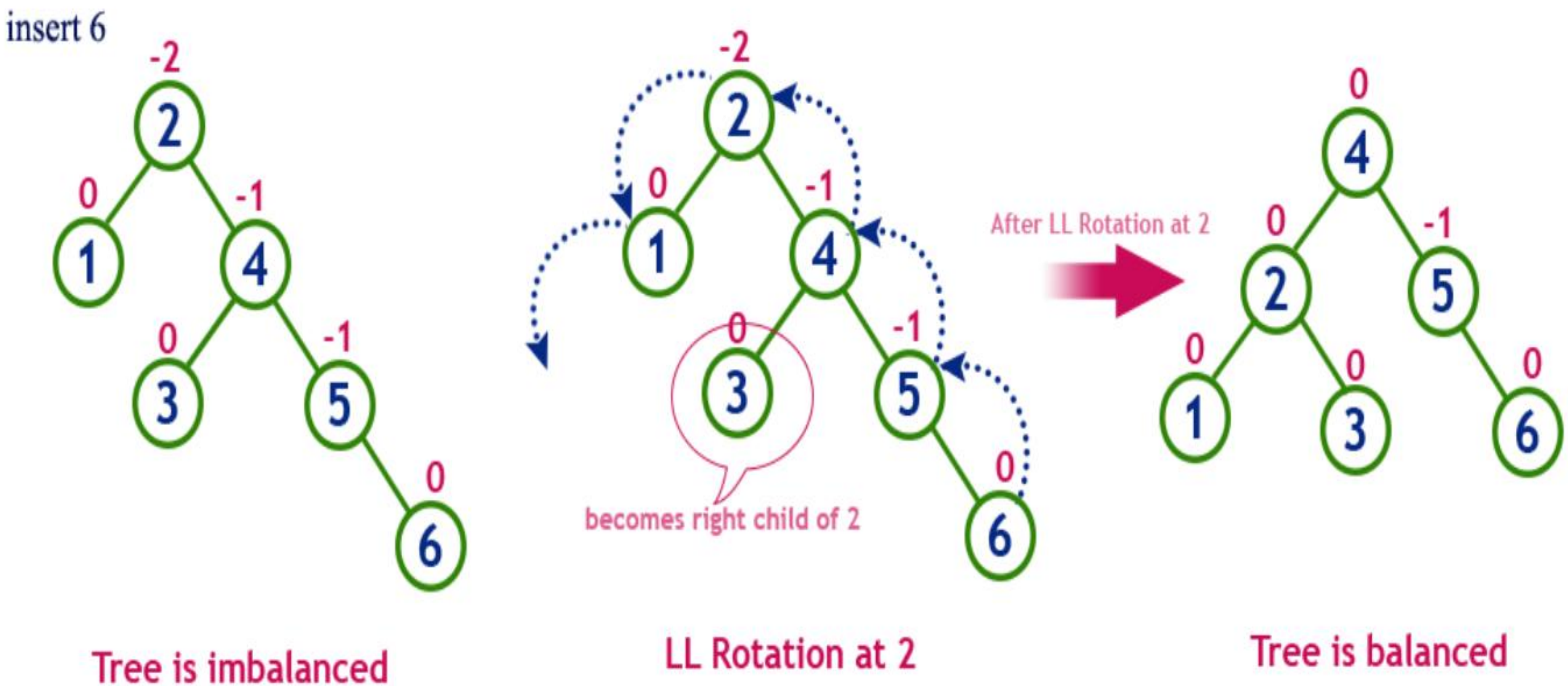
Insertion



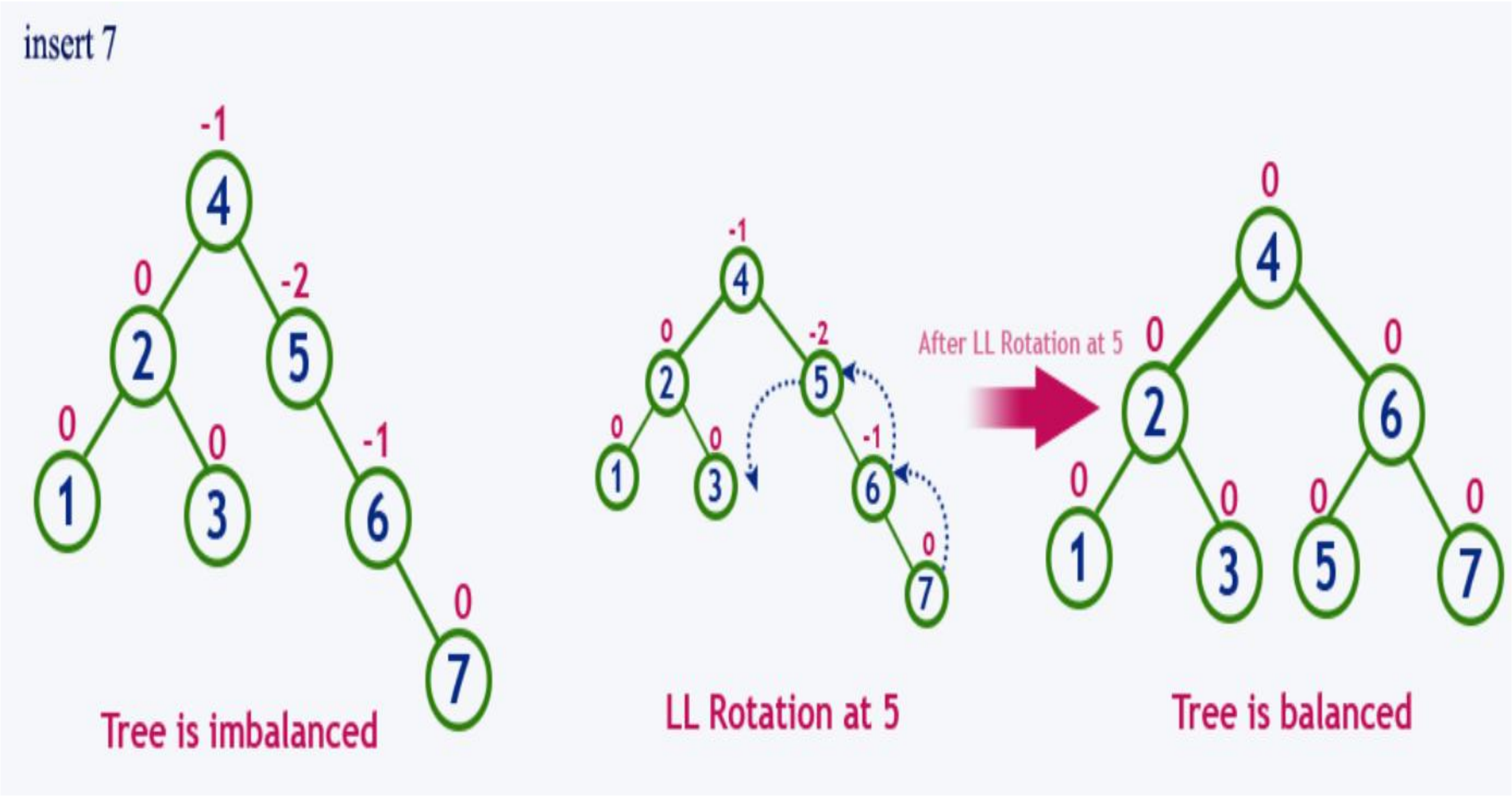
Insertion


insert 5
-2
2
0
-2
1
3
-1
4
0
5
Tree is imbalanced
-2
2
0
-2
1
3
-1
4
0
5
LL Rotation at 3
After LL Rotation at 3
-1
2
0
0
1
4
0
0
3
5
Tree is balanced

Insertion

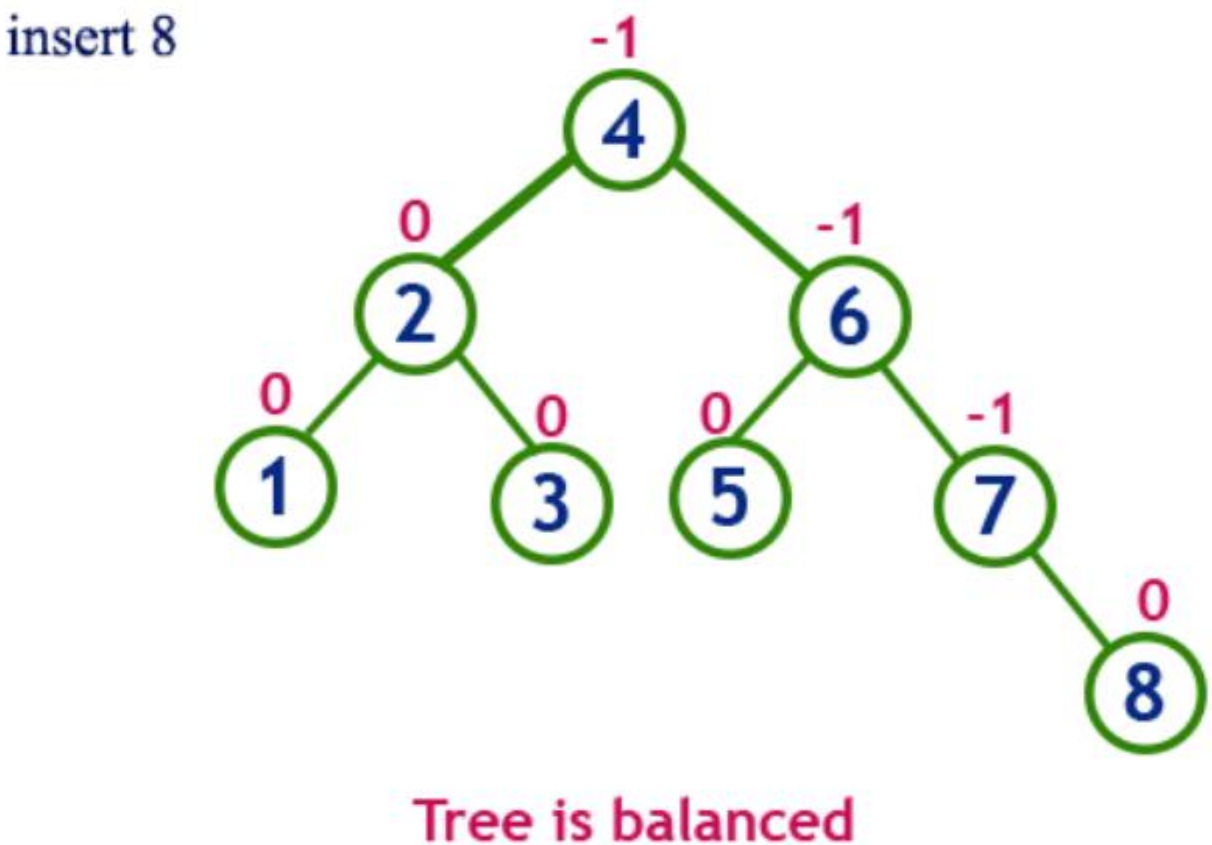

insert 6
-2
2
0
1
-1
4
0
3
-1
5
0
6
Tree is imbalanced
-2
2
0
1
-1
4
0
3
-1
5
0
6
becomes right child of 2
LL Rotation at 2
After LL Rotation at 2
0
4
0
2
-1
5
0
1
0
3
0
6
Tree is balanced

Insertion

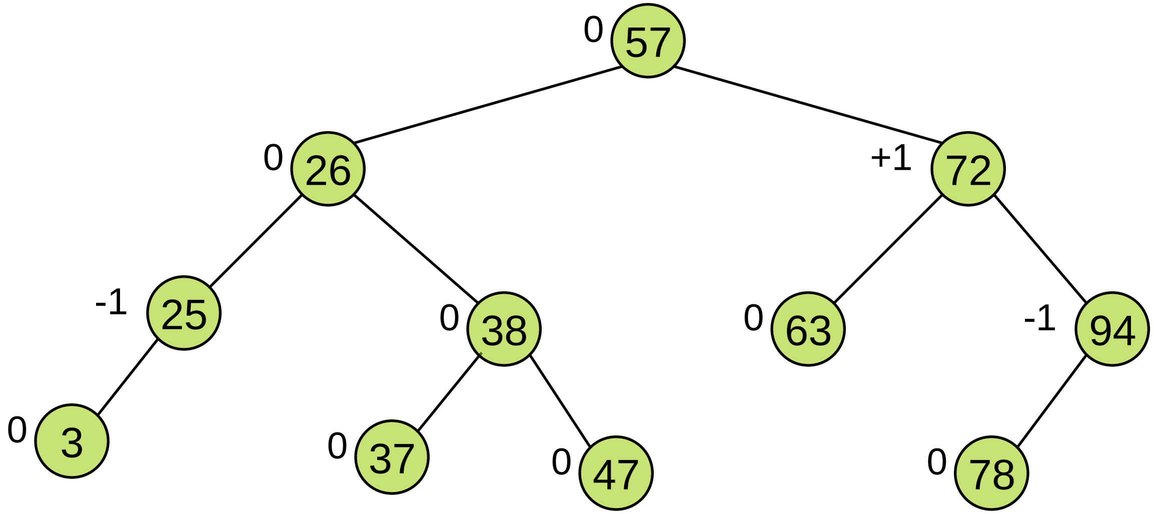


insert 7
-1
4
0
2
-2
5
0
1
0
3
-1
6
0
7
Tree is imbalanced
-1
4
0
2
-2
5
0
1
0
3
-1
6
0
7
LL Rotation at 5
After LL Rotation at 5
0
4
0
2
0
6
0
1
0
3
0
5
0
7
Tree is balanced

Insertion

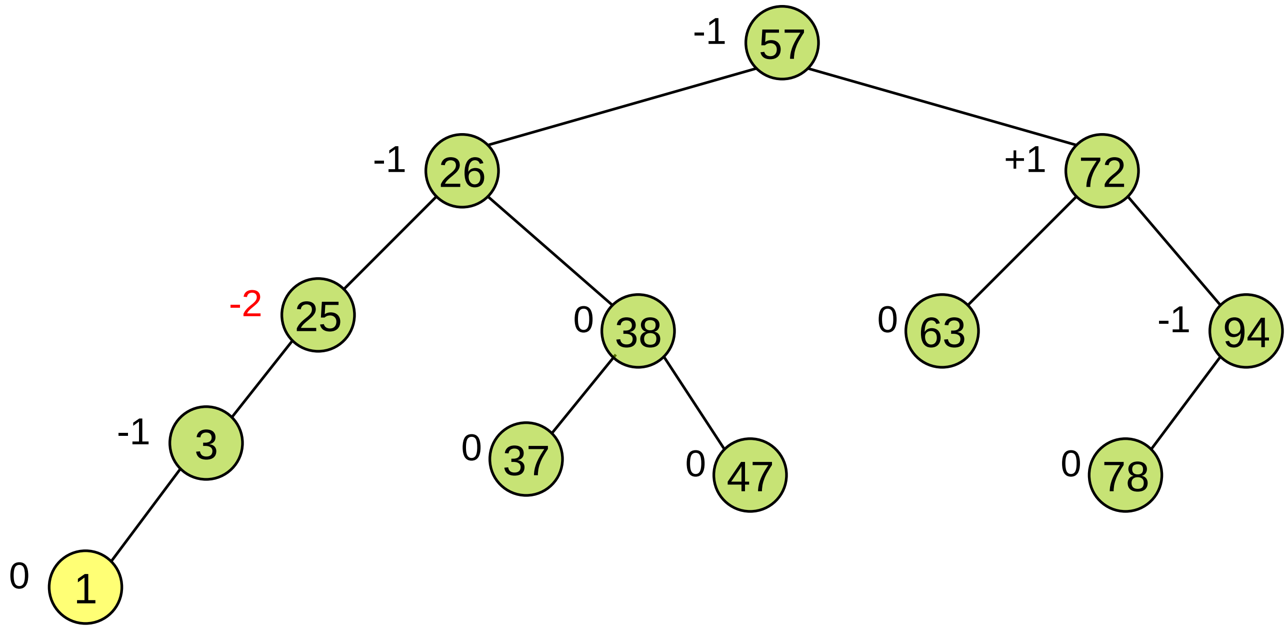


Initial AVL tree with balance factors:



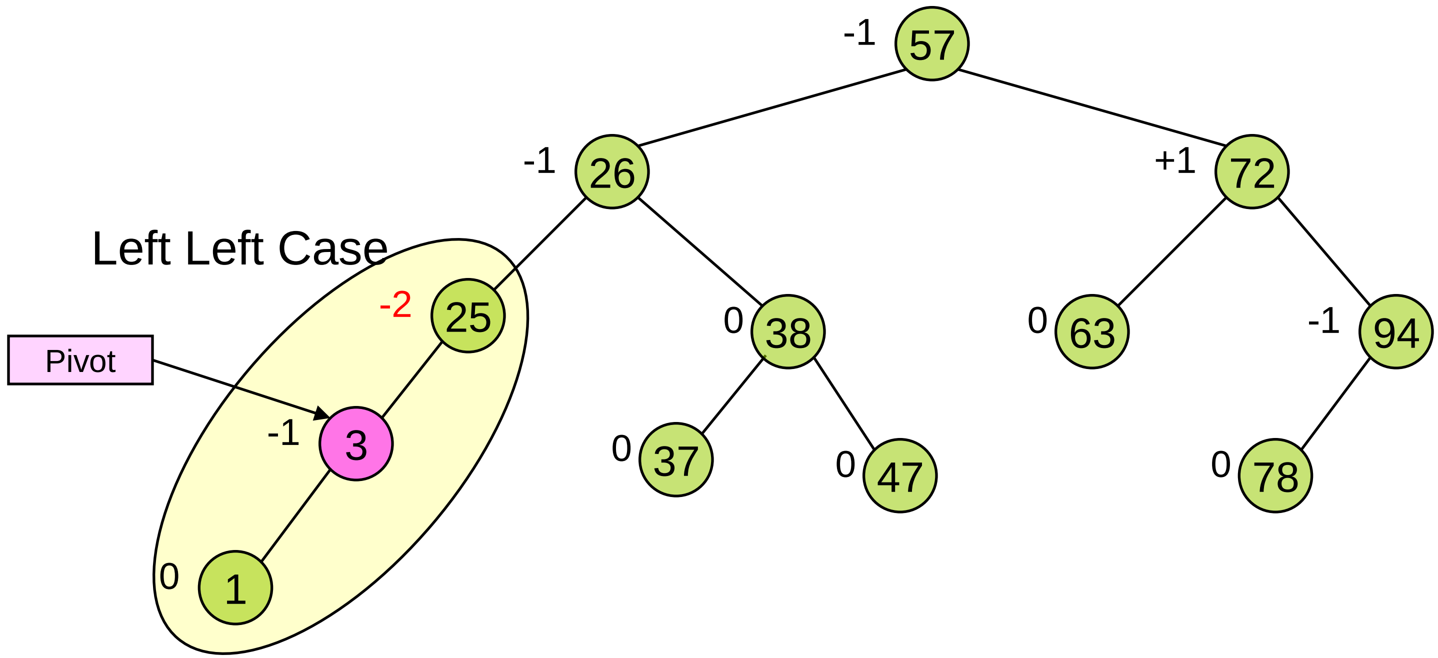
Balance ok

Insert 1 and recalculate balance factors

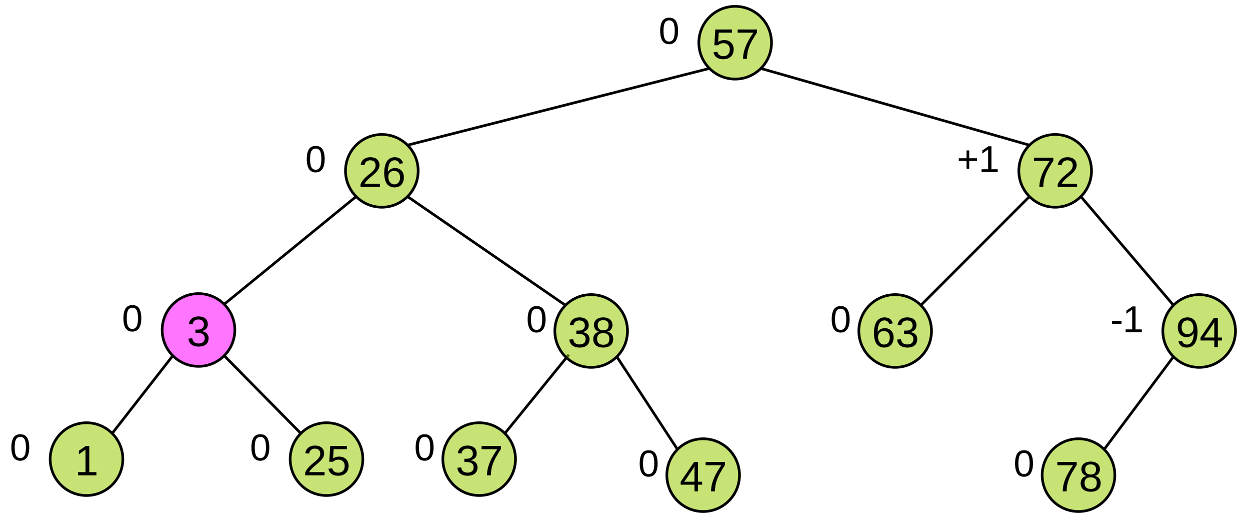


Balance not ok

(Balance factor of -2 is not allowed)

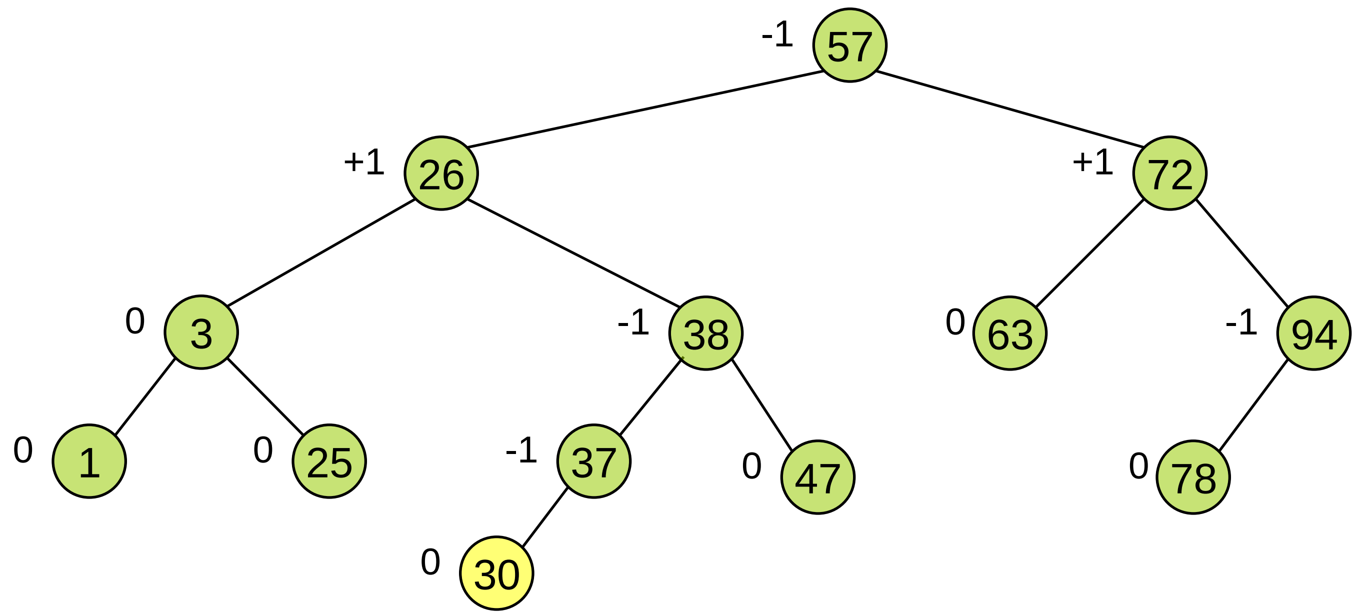


Balance not ok



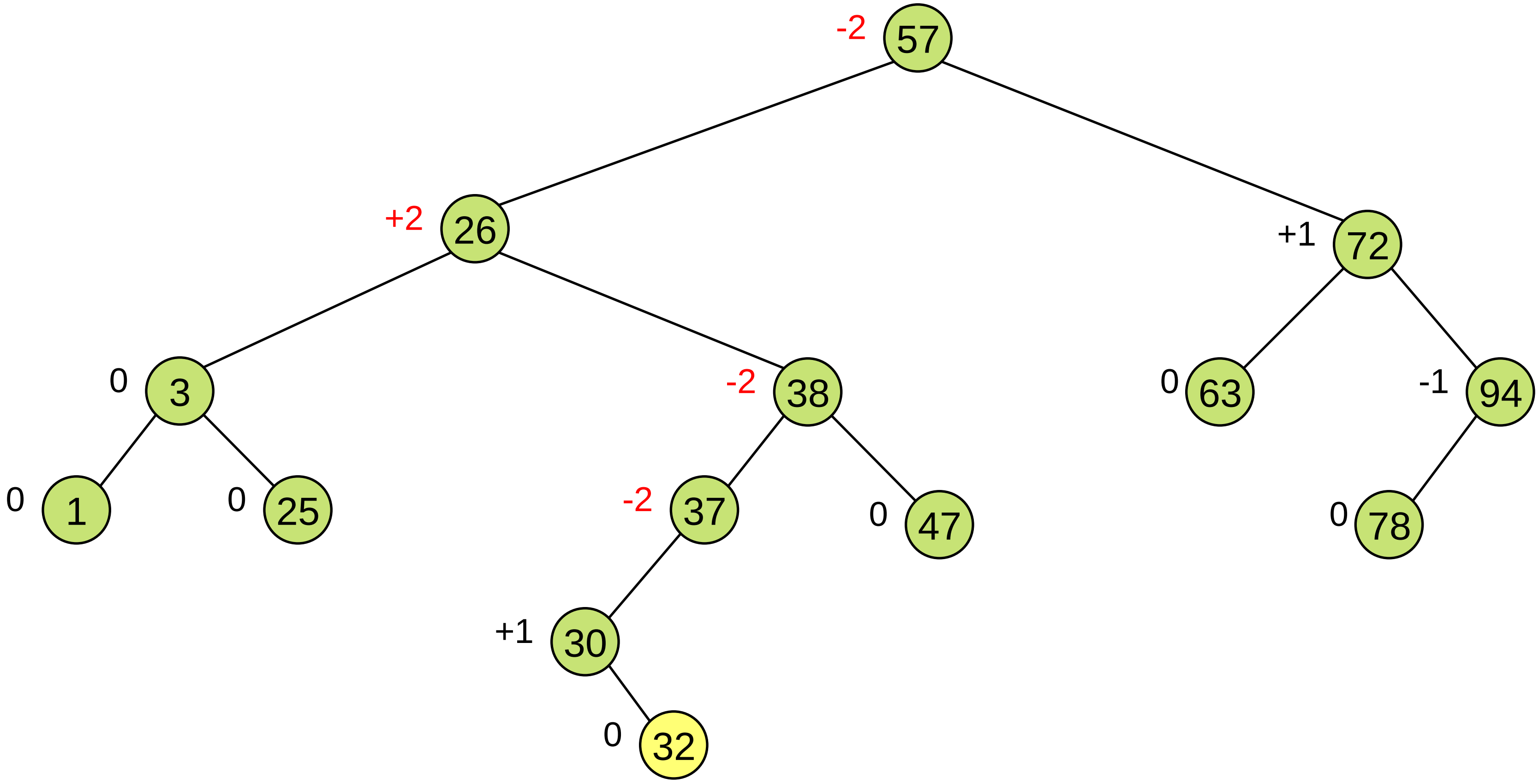
Balance ok

Insert 30 and recalculate balance factors

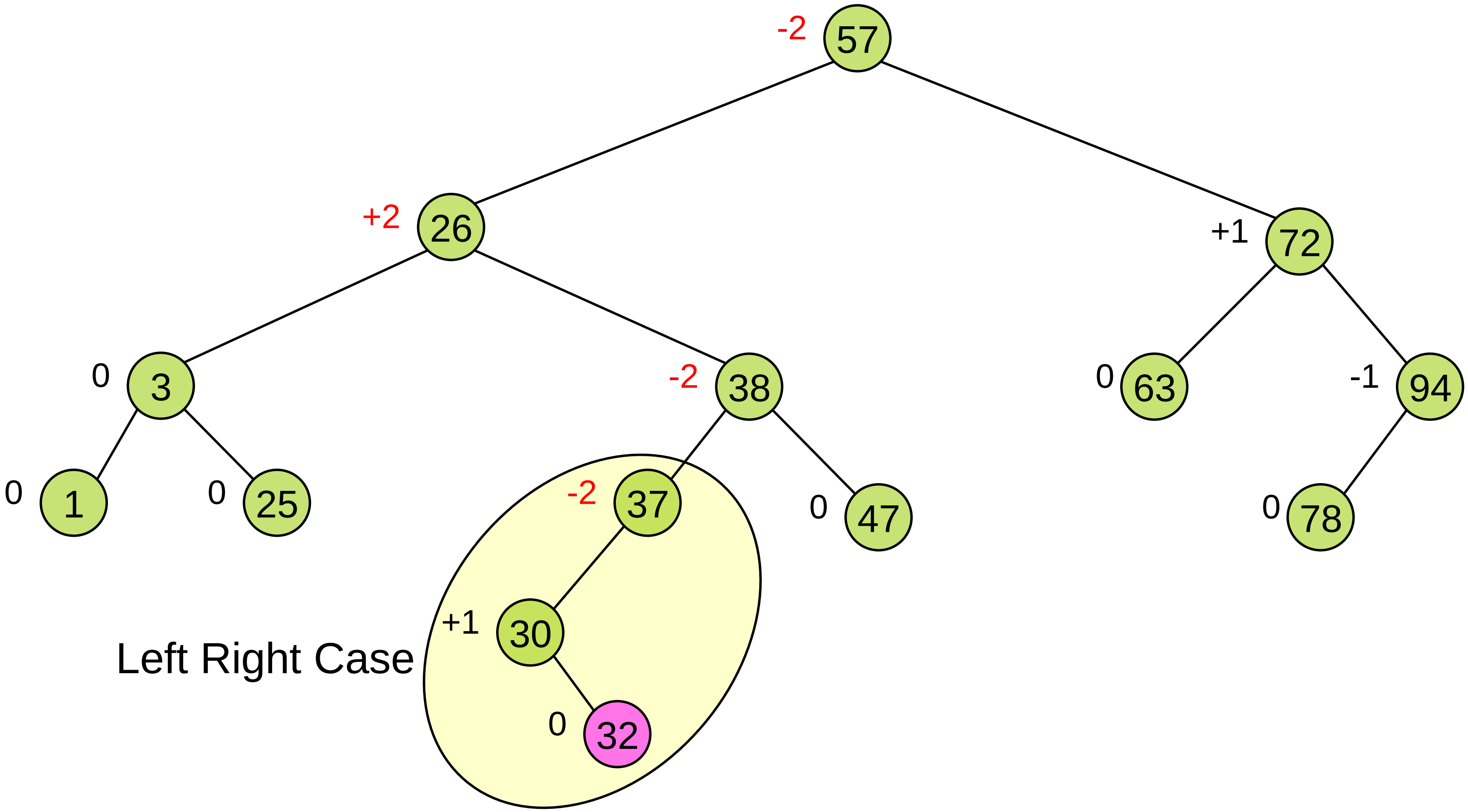


Balance ok

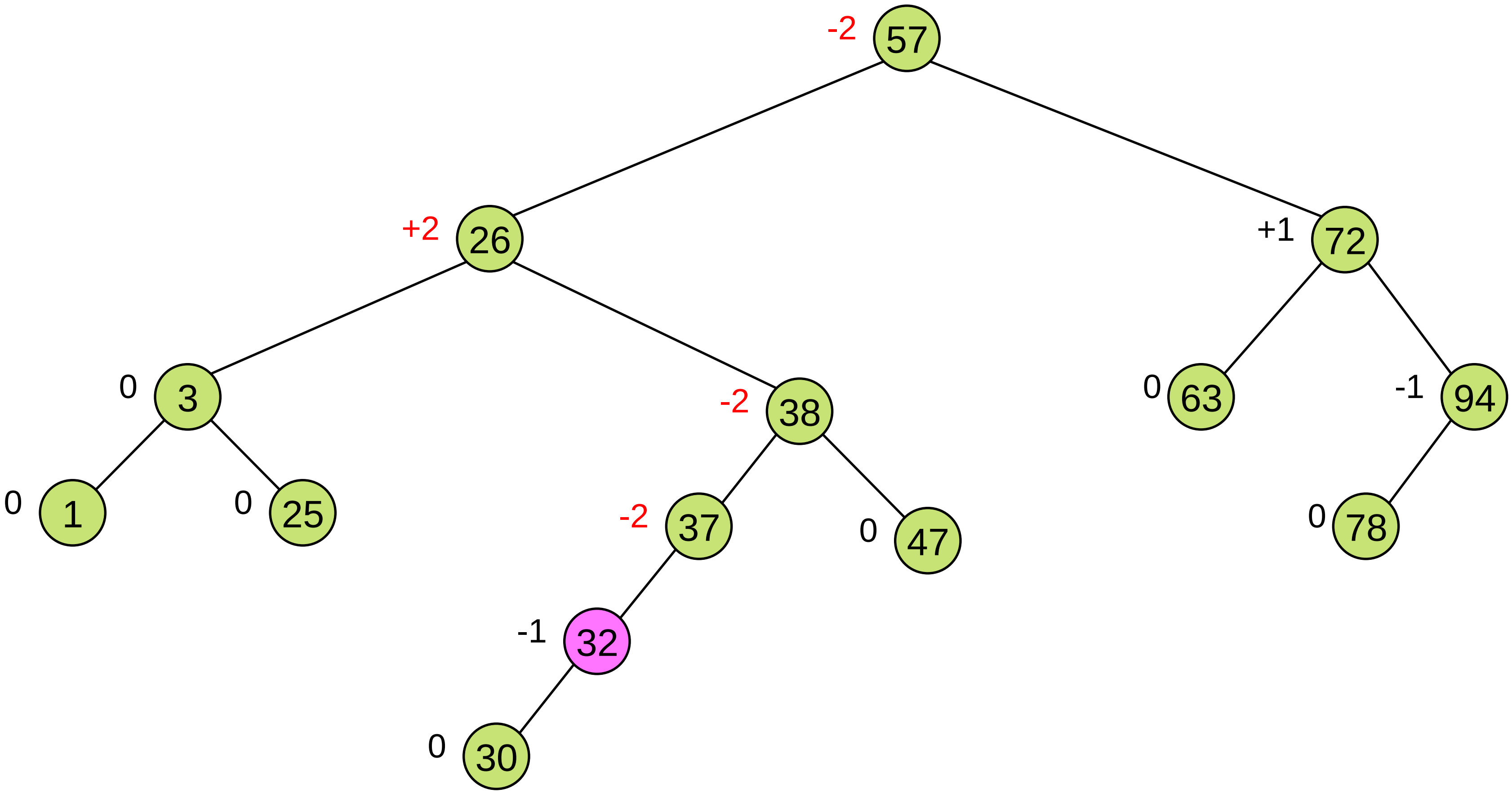
Insert 32 and recalculate balance factors



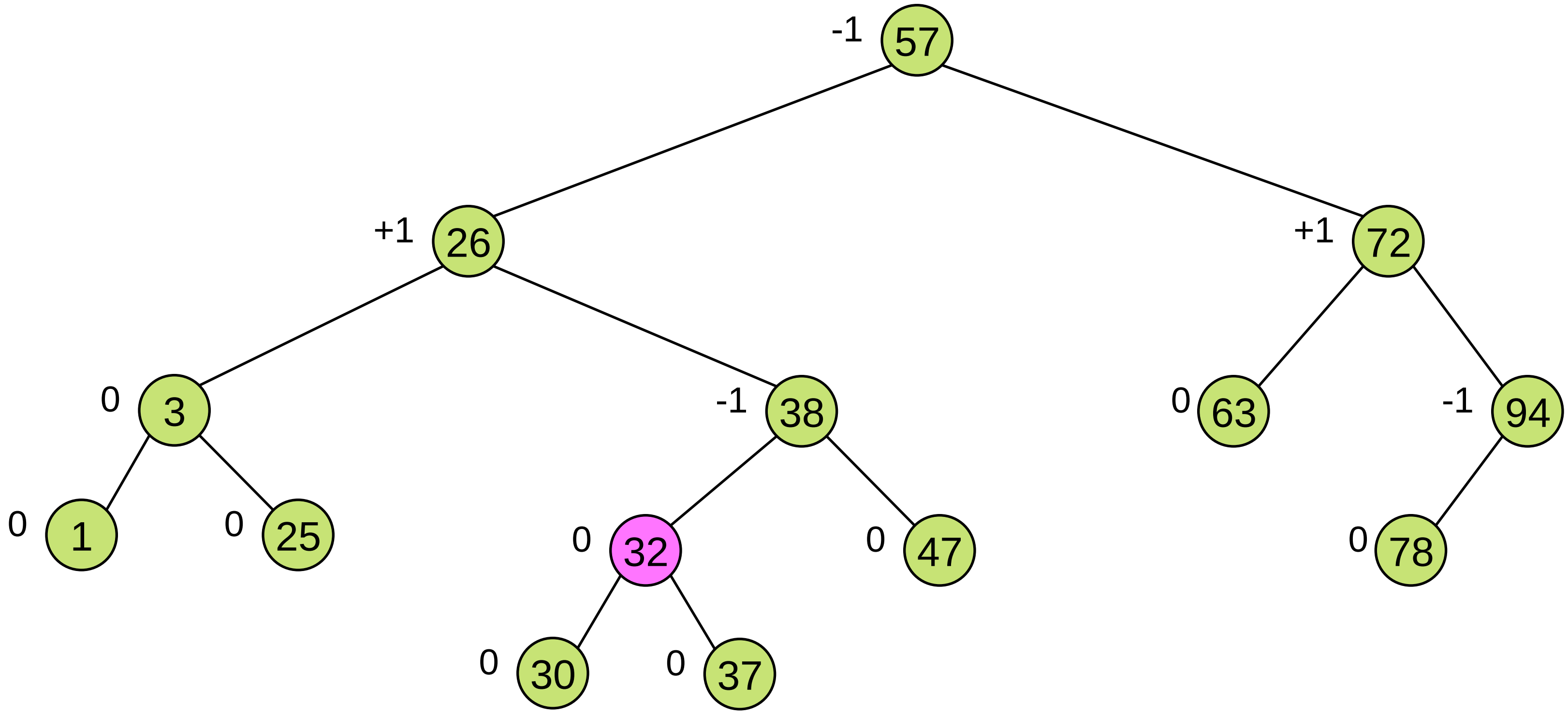
Balance not ok



Balance not ok

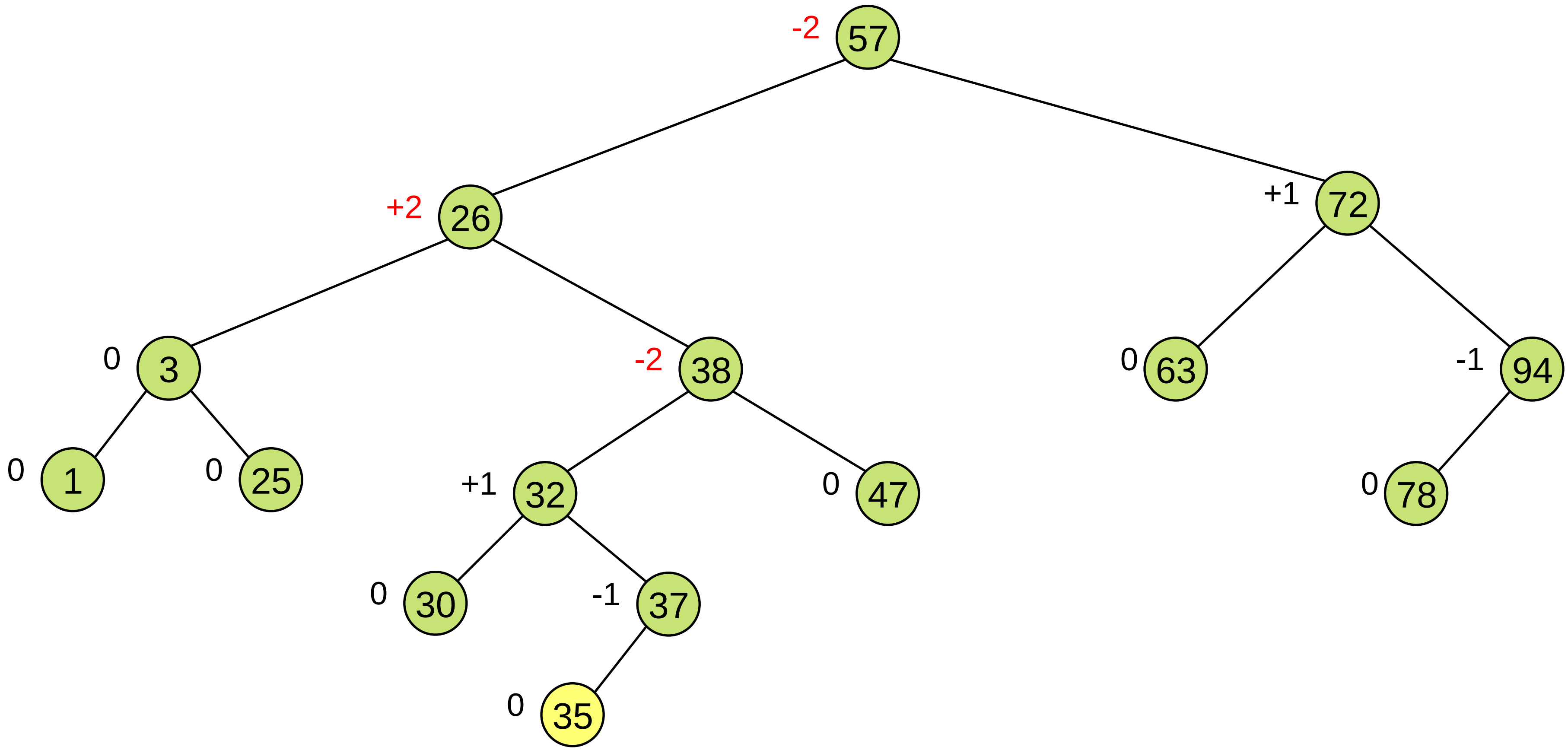


Balance not ok

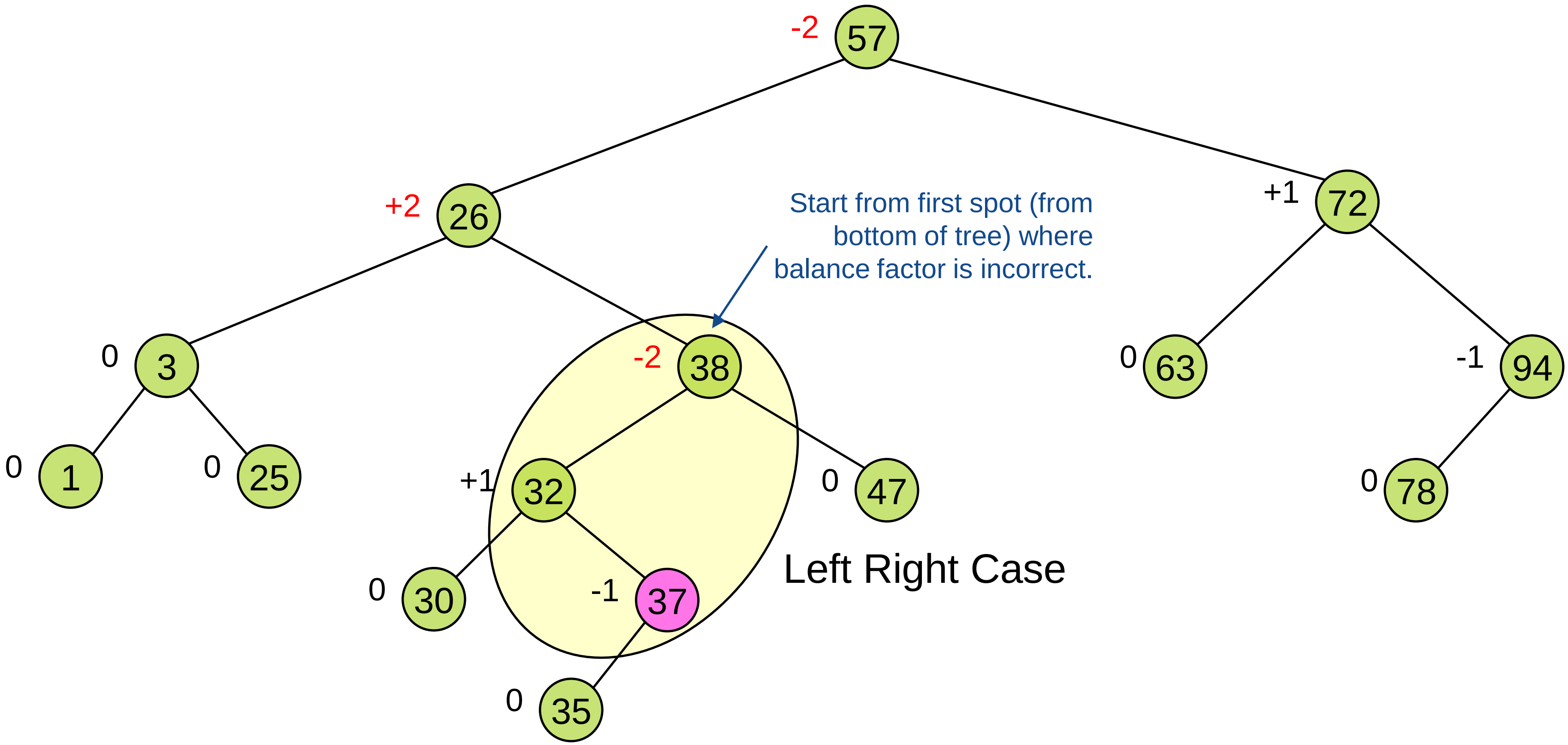


Balance ok

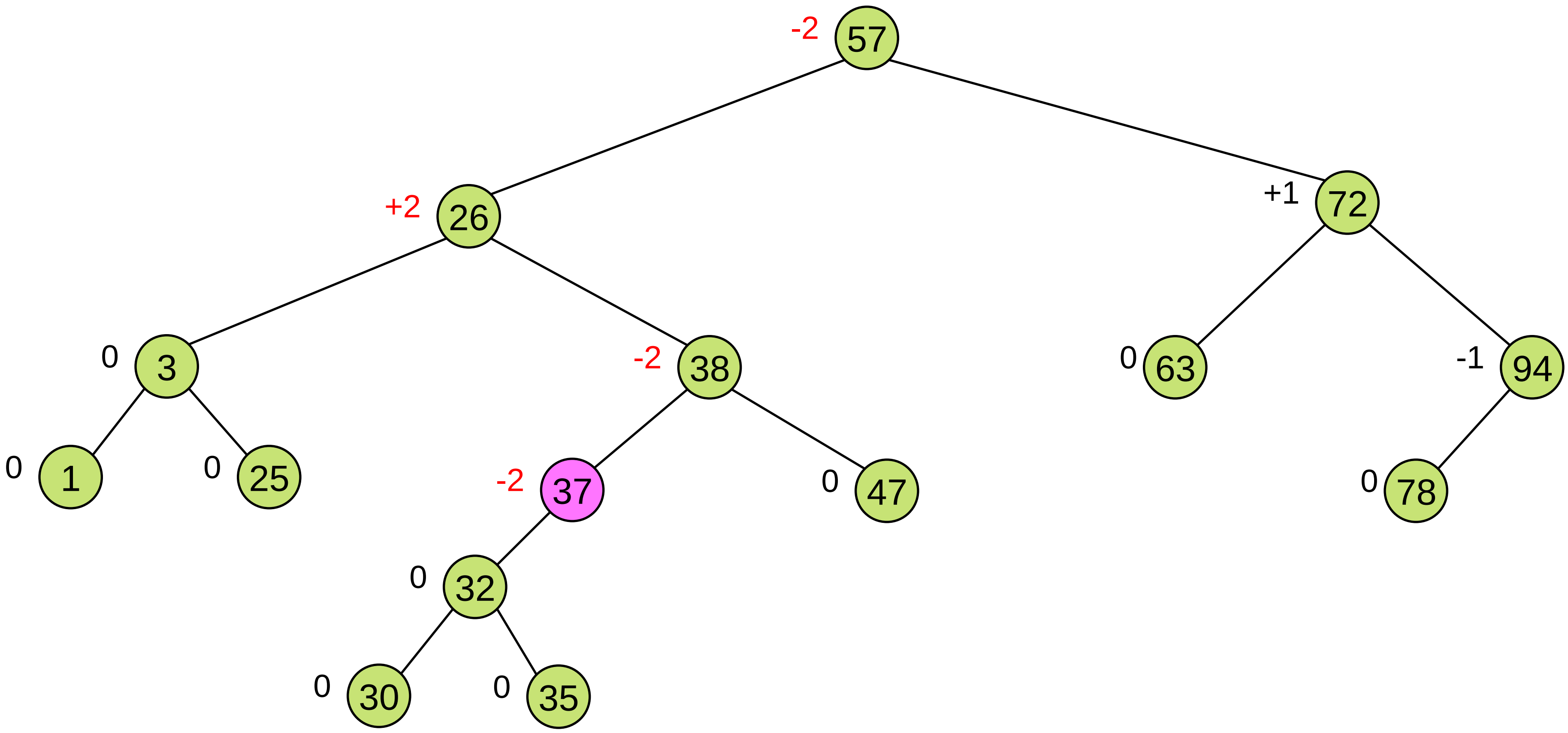
Insert 35 and recalculate balance factors



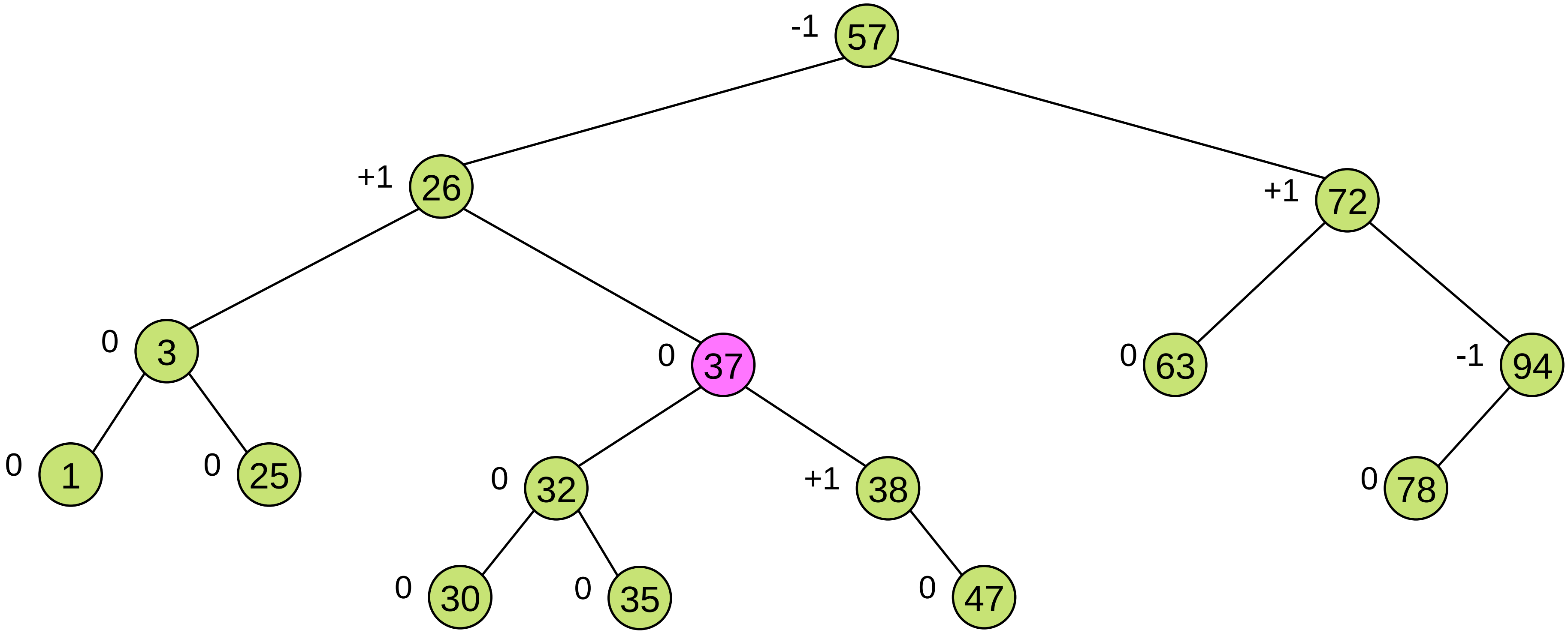
Balance not ok



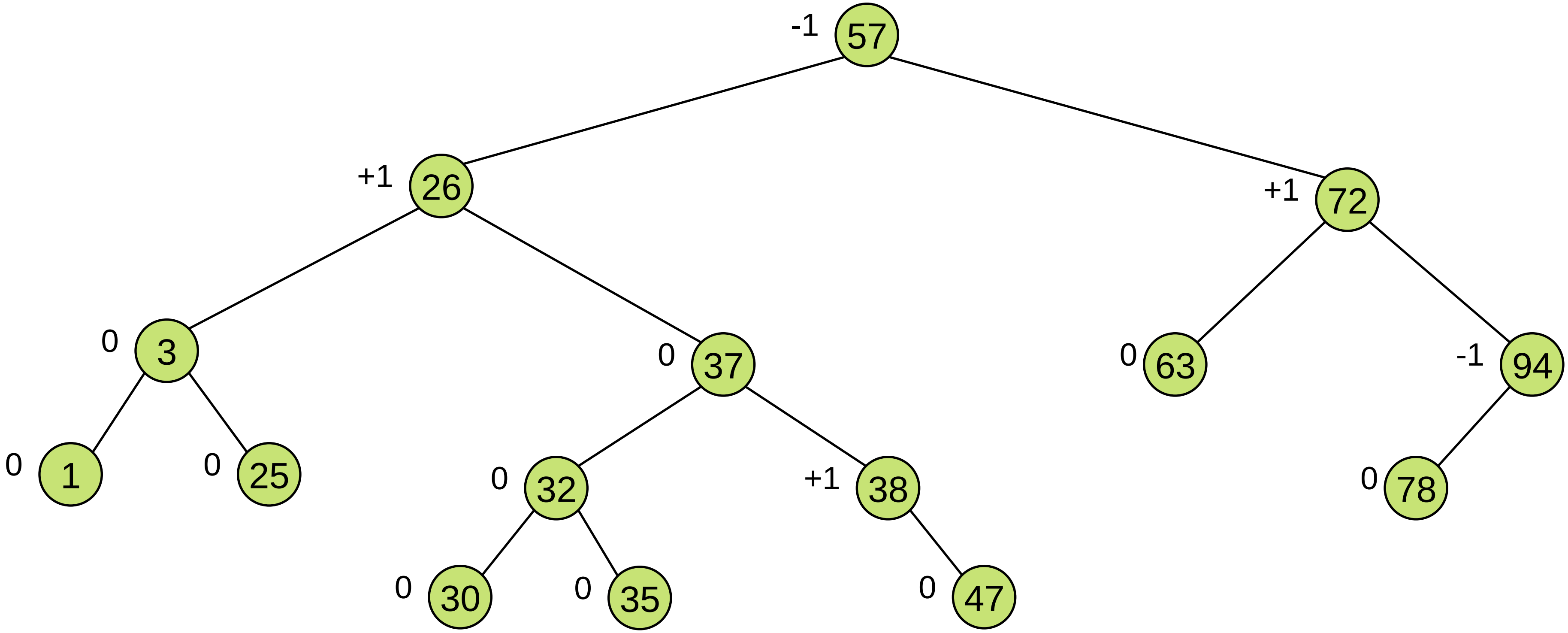
Balance not ok



Balance not ok



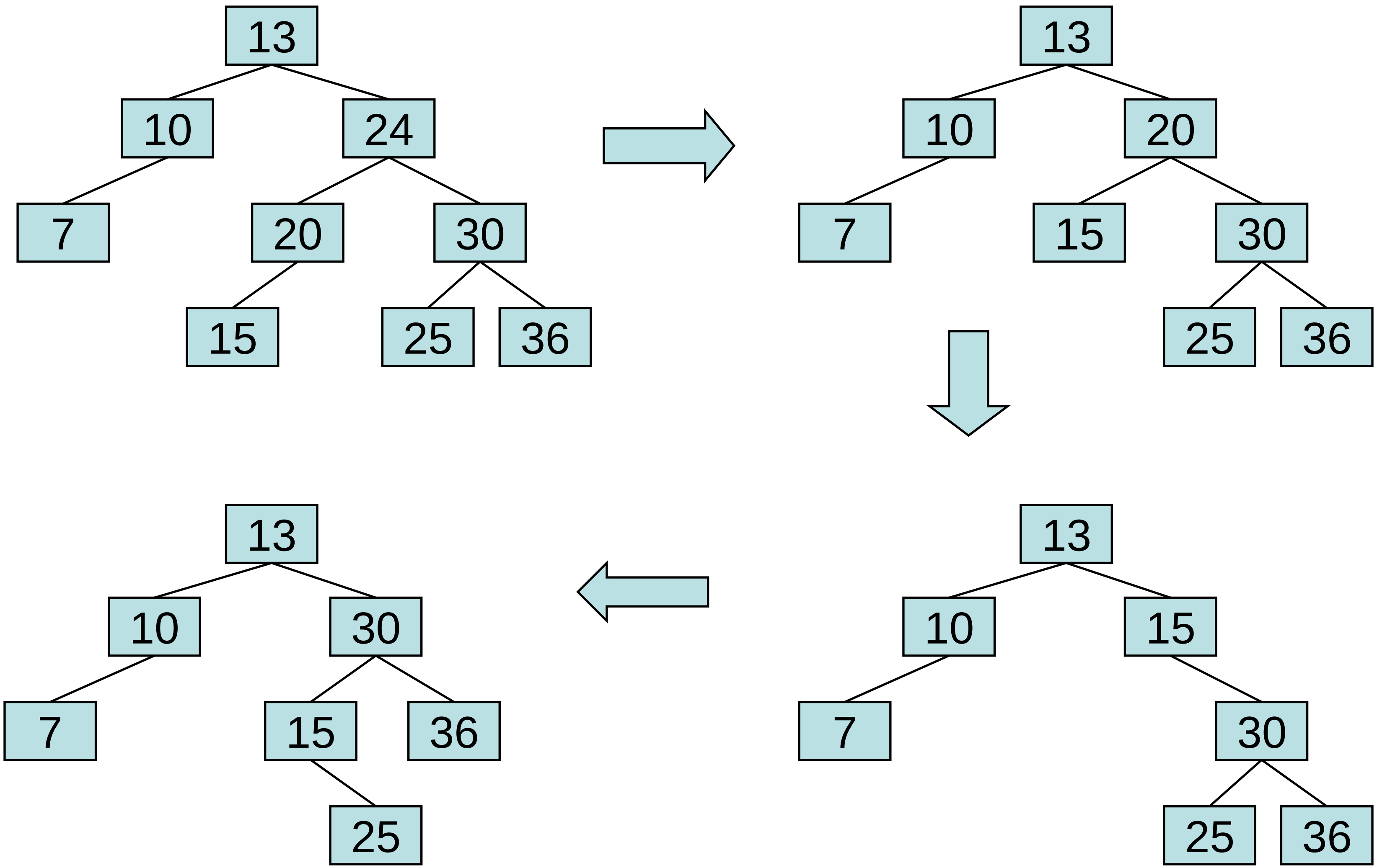
Balance ok



Balance ok

Exercise: insert 36

Remove 24 and 20 from the AVL tree.



END