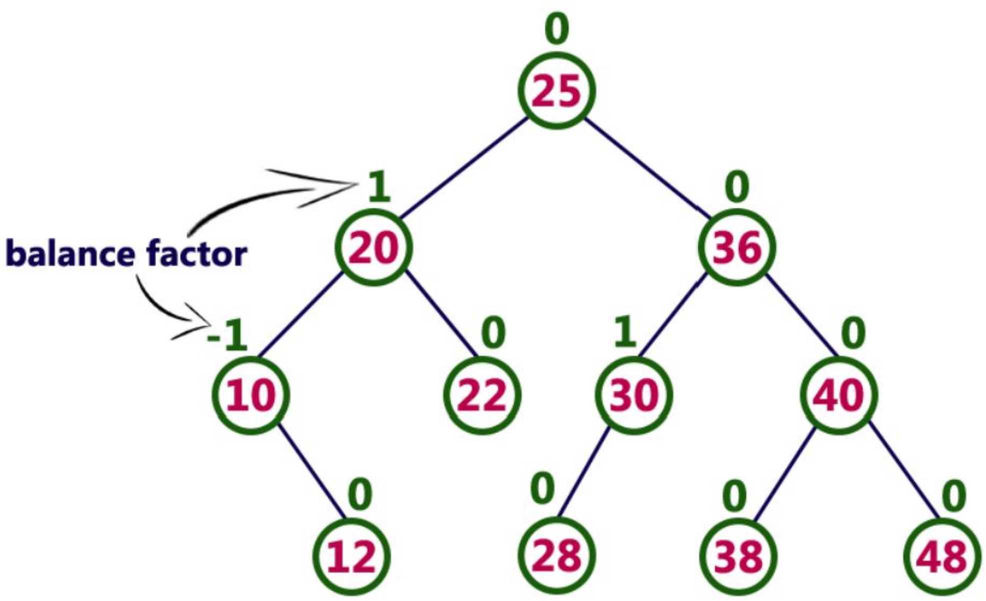
AVL tree

DATA STRUCTURES & ALGORITHMS

AVL tree

An AVL tree is a binary search tree that is height balanced: for each node x, the heights of the left and right subtrees of x differ by at most 1.

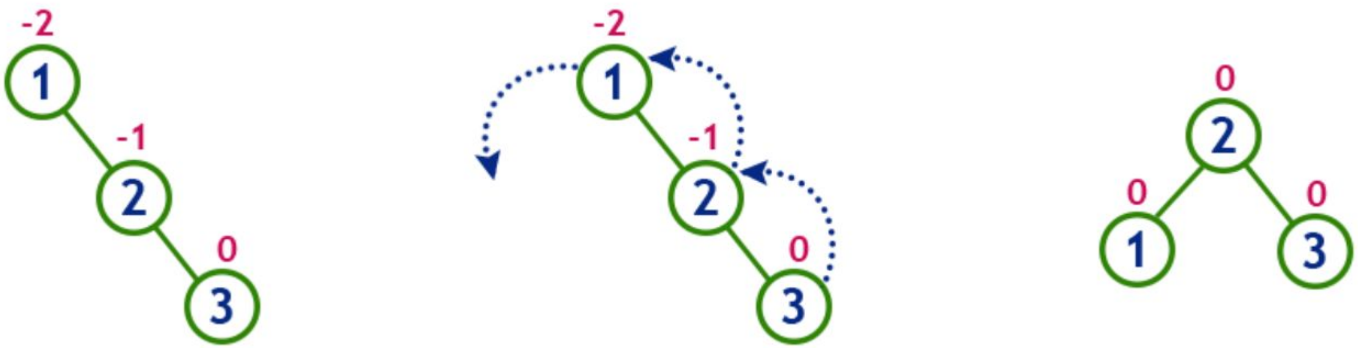


Insert

- standard BST insert for **x**
- Starting from **x**, travel up and find the first unbalanced node. Let **z** be the first unbalanced node, **y** be the child of z that comes on the path from **w** to **z** and **t** be the grandchild of **z** that comes on the path from **x** to **z**.
- 4 possible cases:
  - **y** is the left child of **z** and t is the left child of **y** (**Left Left** Case)
  - **y** is the left child of **z** and t is the right child of **y** (**Left Right** Case)
  - **y** is the right child of **z** and t is the right child of **y** (**Right Right** Case)
  - **y** is the right child of **z** and t is the left child of **y** (**Right Left** Case)

Left left case

insert 1, 2 and 3

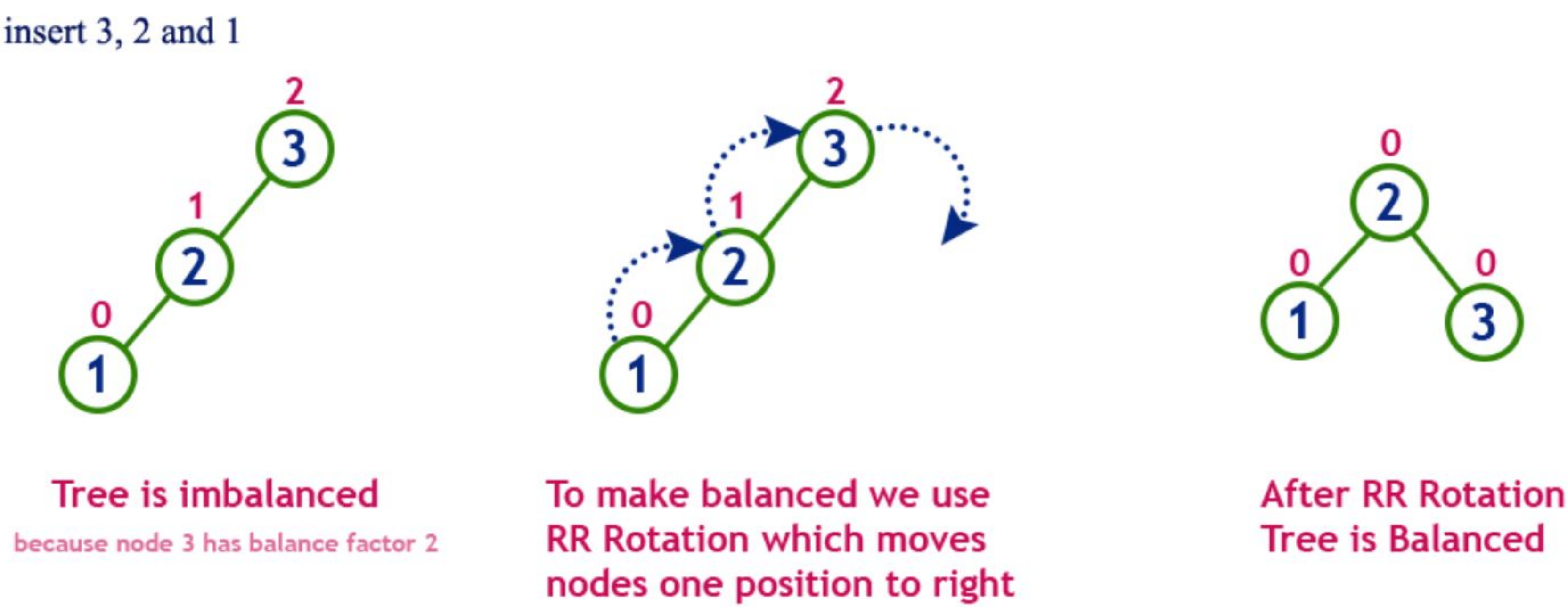


Tree is imbalanced

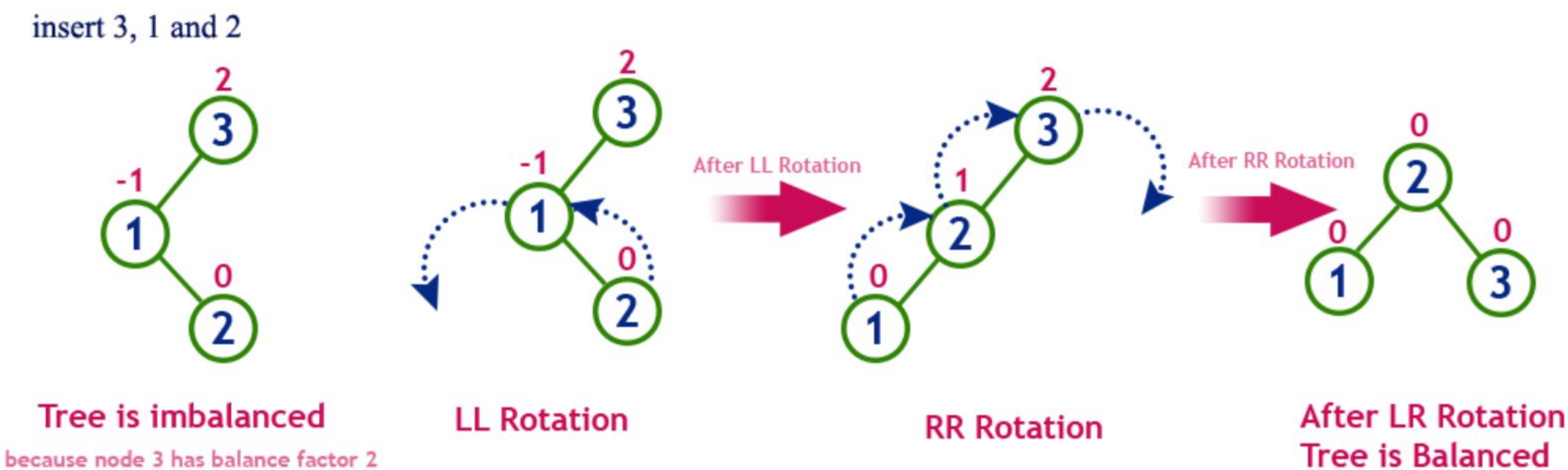
To make balanced we use LL Rotation which moves nodes one position to left

After LL Rotation Tree is Balanced

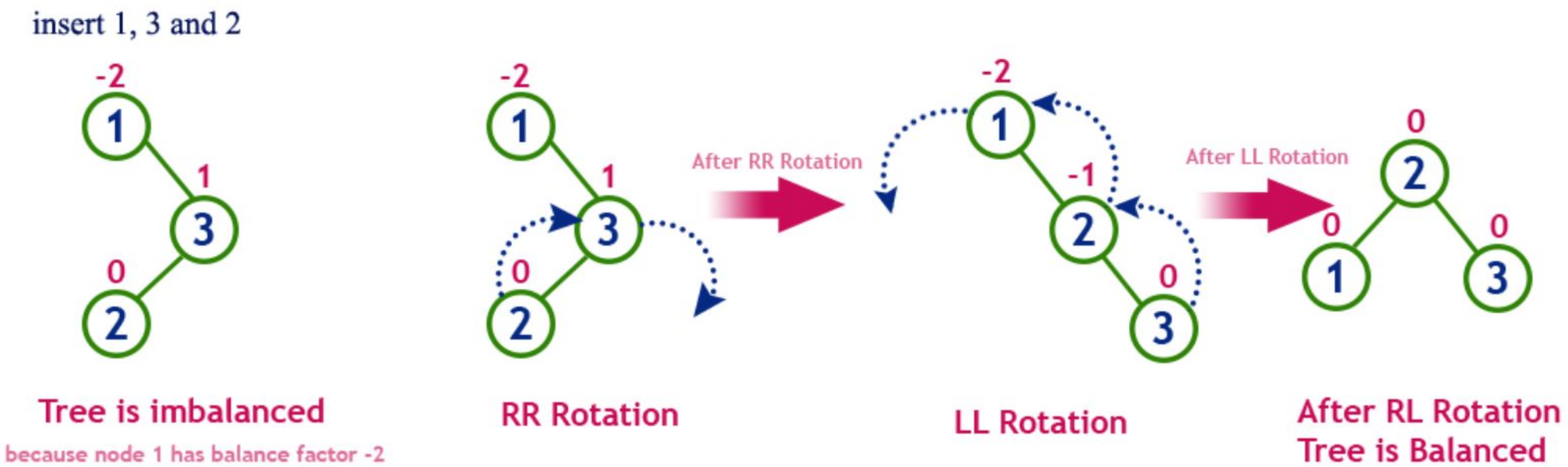
Right right case



Left right case



Right left case



AVL tree height

- $N(h-1)$, the minimum number of nodes in the left subtree of r
- $N(h-2)$, the minimum number of nodes in the right subtree of r.

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

We assumed that $N(h-1) > N(h-2)$, so we can say that

$$N(h) > 1 + N(h-2) + N(h-2) = 1 + 2 \cdot N(h-2) > 2 \cdot N(h-2)$$

So we have:

$$N(h) > 2 \cdot N(h-2)$$

We can try to solve this as a recurrence (note that $N(0) = 1$):

$$N(h) > 2 \cdot N(h-2) > 2 \cdot 2 \cdot N(h-4) > 2 \cdot 2 \cdot 2 \cdot N(h-6) > \cdots > 2^{h/2}$$

You can see it's $2^{h/2}$ by checking for a particular $h = 6$:

$$N(6) > 2 \cdot N(6-2) > 2 \cdot 2 \cdot N(4-2) > 2 \cdot 2 \cdot 2 \cdot N(2-2) > 2^3$$

Now, we can try and bound h:

$$N(h) > 2^{h/2} \overset{Take \log}{\Longleftrightarrow} \log N(h) > \log 2^{h/2} \Leftrightarrow h < 2 \log N_h$$