Data Structures

Balanced Binary Search Trees

CS 225
Brad Solomon

September 25, 2024

University of Illinois Urbana-Champaign
Department of Computer Science

Additional Extra Credit / Research Opportunity

Research Survey by Morgan Fong, PhD student studying CS Education

Study meant to measure sense of belonging in CS courses

You are asked to complete surveys periodically

Completing survey will award +2 bonus points

Points are awarded individually!

Research permission not necessary!

Learning Objectives

Briefly review BST in the context of height

Discuss the big picture problem with BSTs

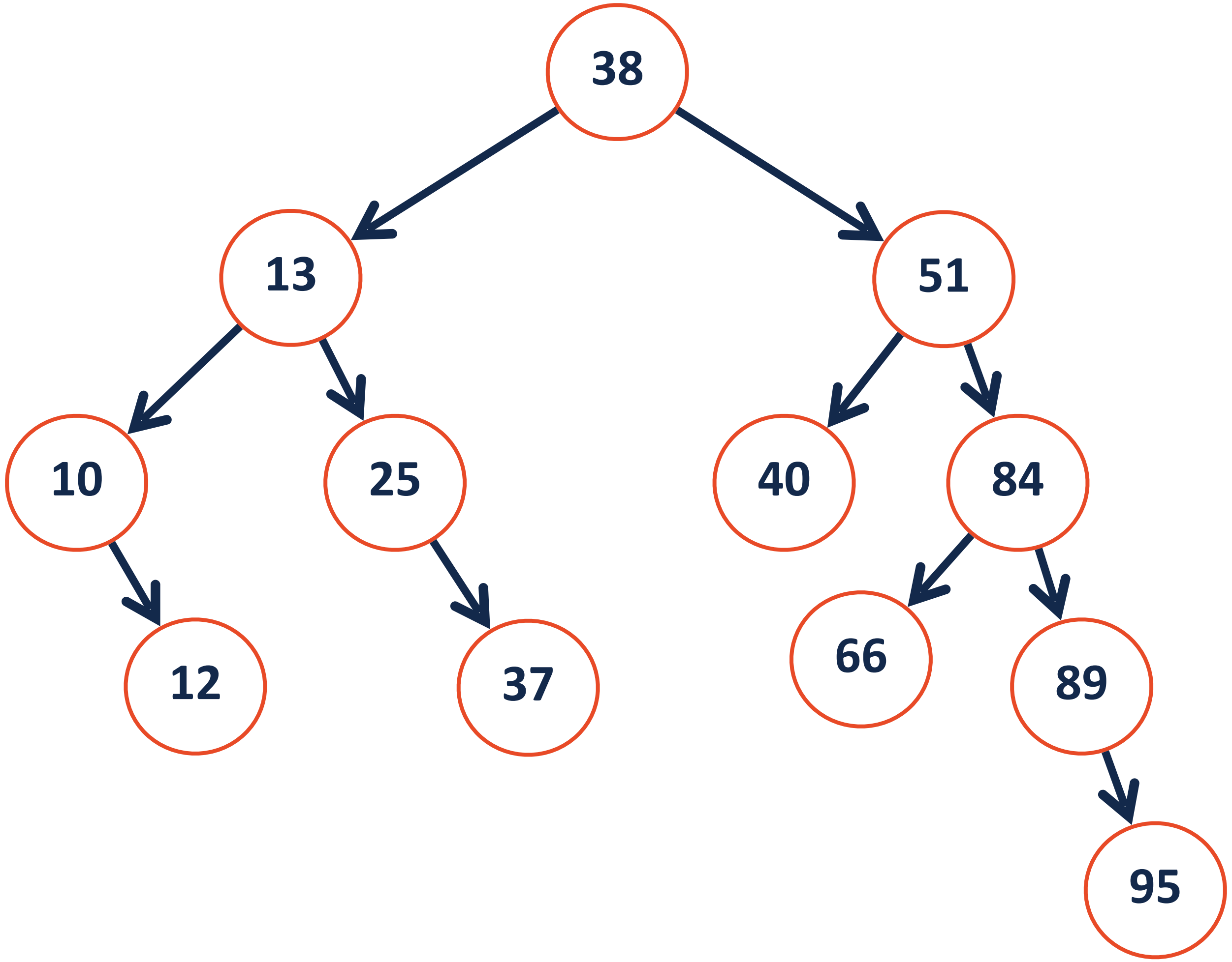
Introduce the self-balancing BST

BST Analysis – Running Time

Operation	BST Worst Case
find	O(h)
insert	O(h)
remove	O(h)
traverse	O(n)

BST Analysis – Running Time

Operation	BST Worst Case
find	O(h)
insert	O(h)
remove	O(h)
traverse	O(n)



BST Analysis

Every operation on a BST depends on the **height** of the tree.

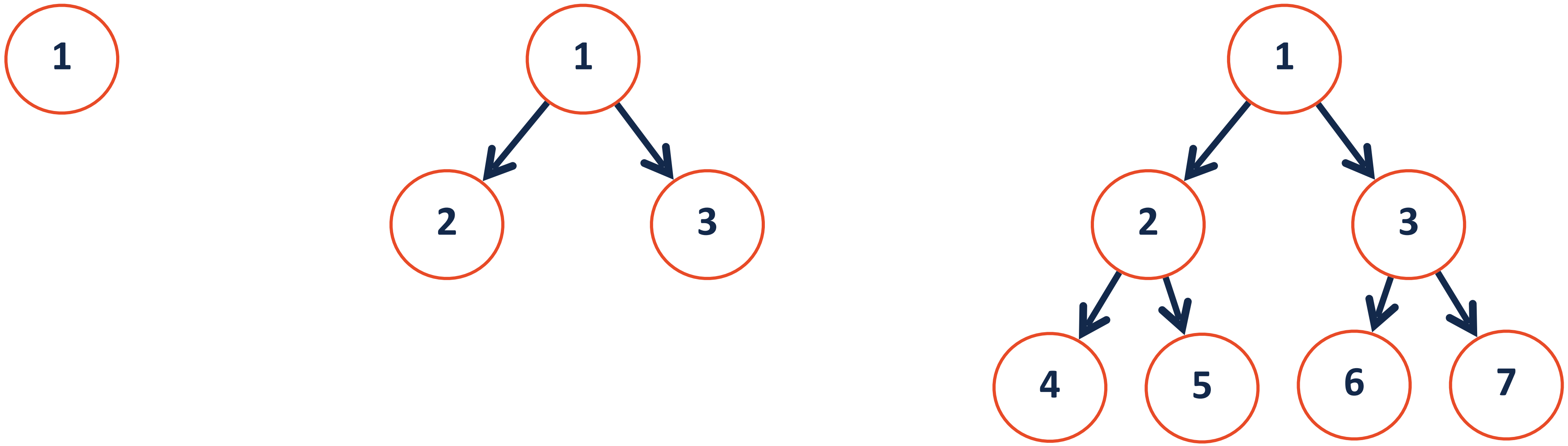
… how do we relate $O(h)$ to n, the size of our dataset?

BST Analysis

What is the **max** number of nodes in a tree of height h ?

BST Analysis

What is the **max** number of nodes in a tree of height h ?

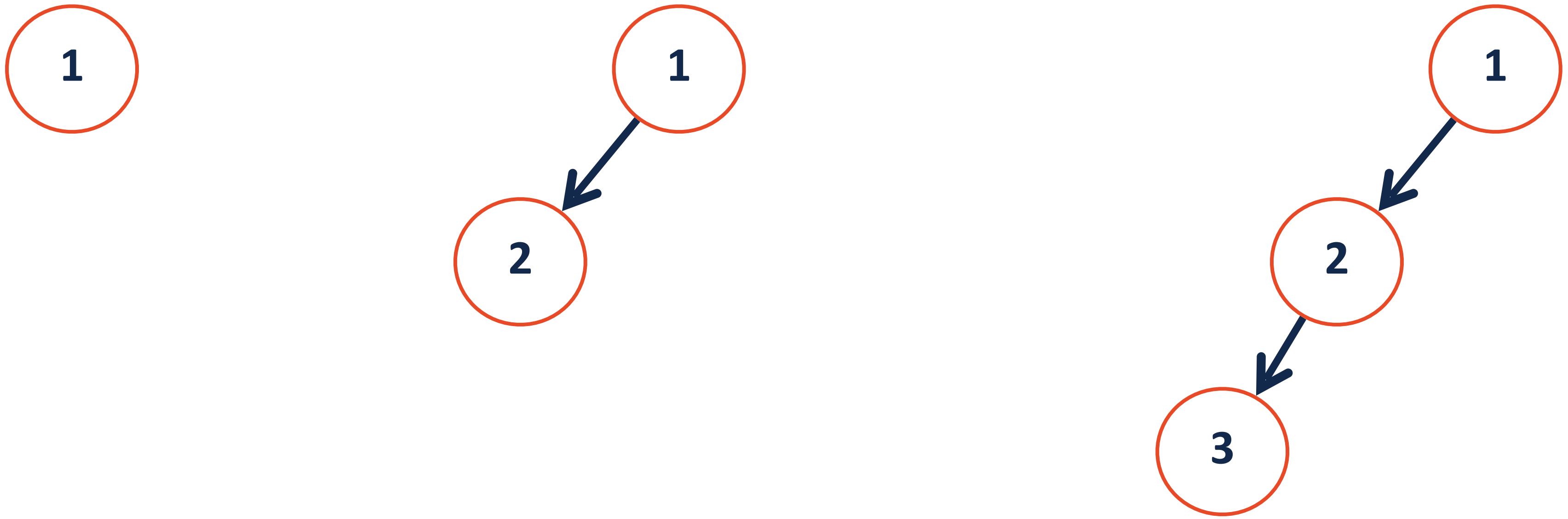


BST Analysis

What is the **min** number of nodes in a tree of height h ?

BST Analysis

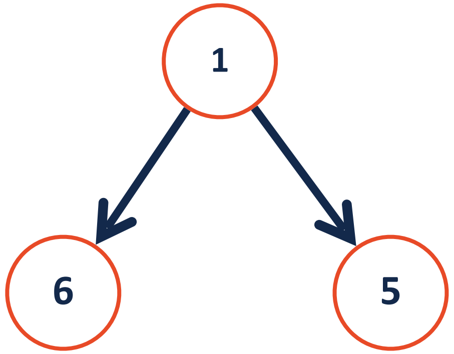
What is the **min** number of nodes in a tree of height h ?



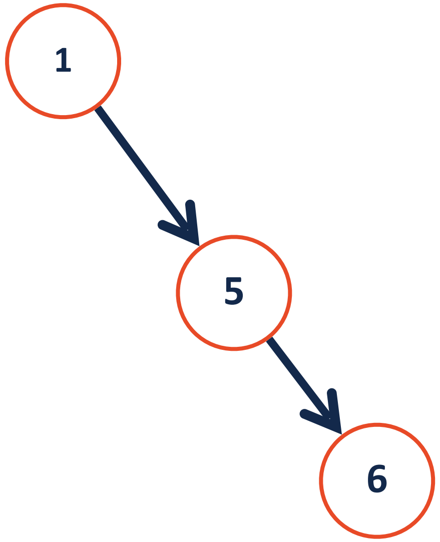
BST Analysis

A BST of n nodes has a height between:

Lower-bound: $O(\log n)$

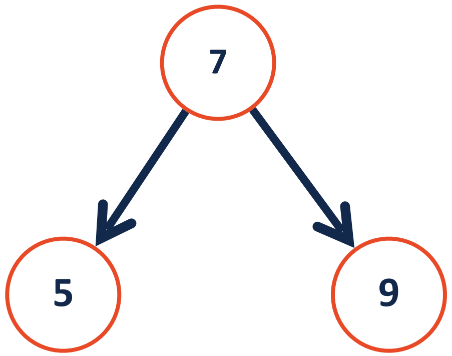


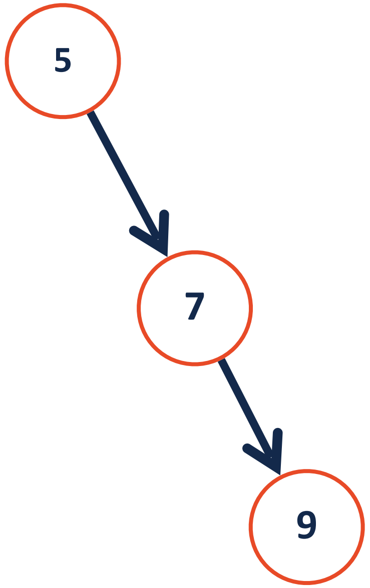
Upper-bound: $O(n)$



Height-Balanced Tree

What tree is better?





Height balance: $b = height(T_R) - height(T_L)$

A tree is "balanced" if:

Option A: Correcting bad insert order

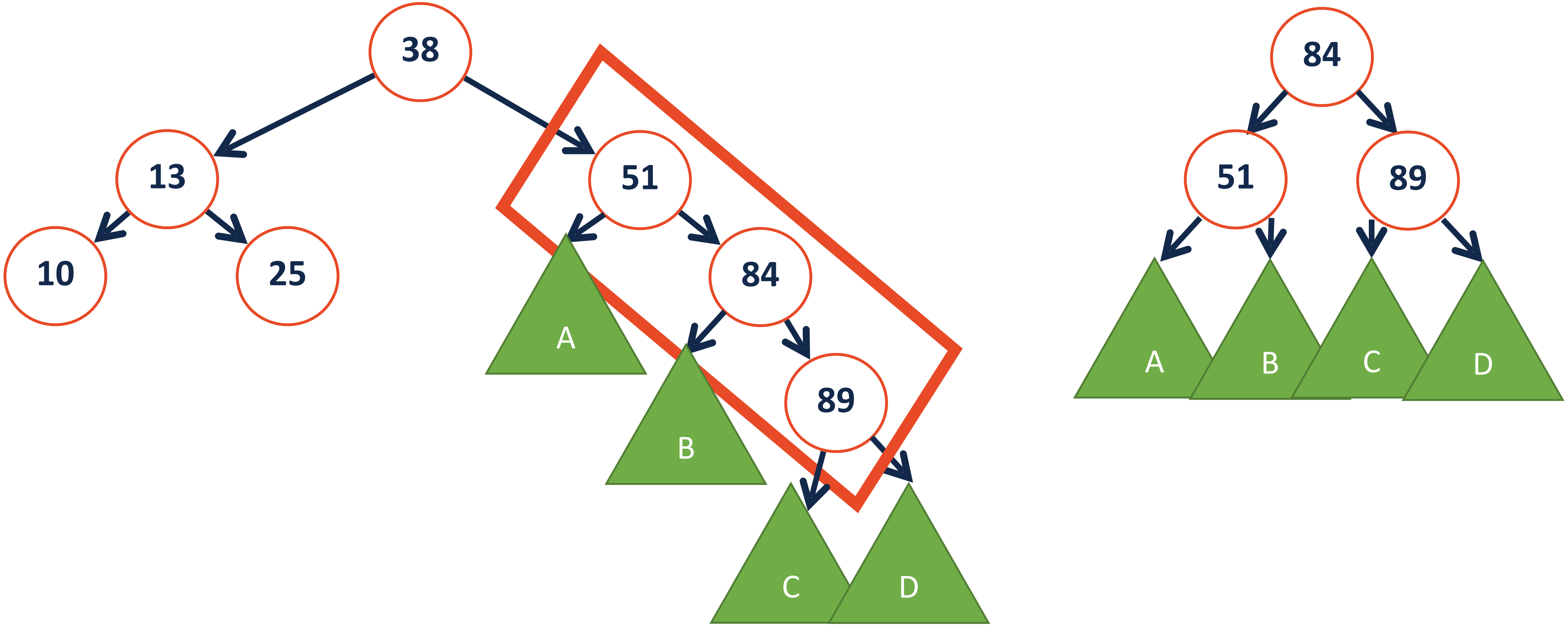
The height of a BST depends on the order in which the data was inserted

Insert Order: [1, 3, 2, 4, 5, 6, 7]

Insert Order: [4, 2, 3, 6, 7, 1, 5]

AVL-Tree: A self-balancing binary search tree

Rather than fixing an insertion order, just correct the tree as needed!



BST Rotations (The AVL Tree)

We can adjust the BST structure by performing **rotations**.

These rotations:

1.

2.

BST Rotations (The AVL Tree)

We can adjust the BST structure by performing **rotations**.

These rotations:

1. Modify the arrangement of nodes while preserving BST property

2.

BST Rotations (The AVL Tree)

We can adjust the BST structure by performing **rotations**.

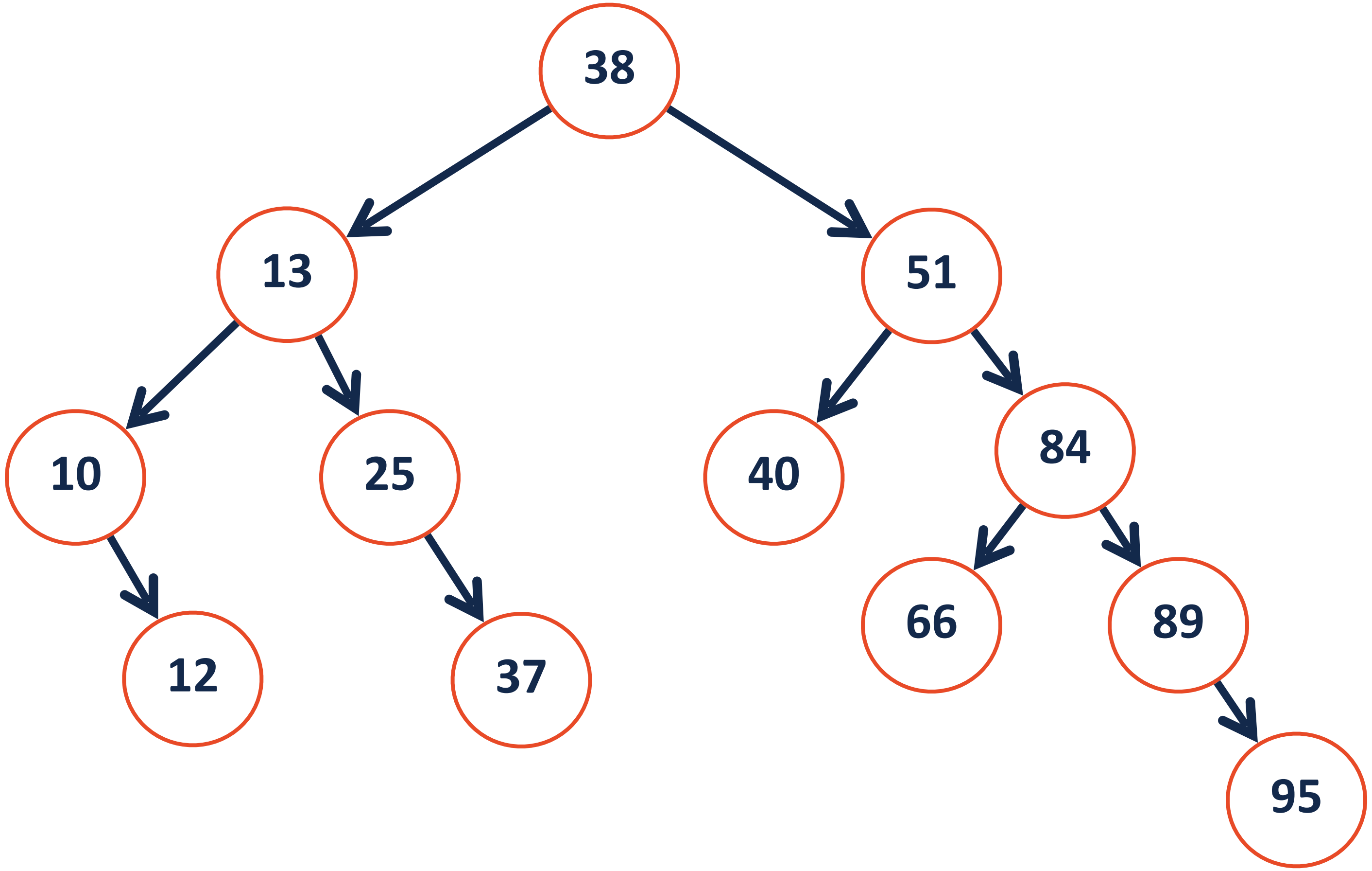
These rotations, when used correctly:

1. Modify the arrangement of nodes while preserving BST property

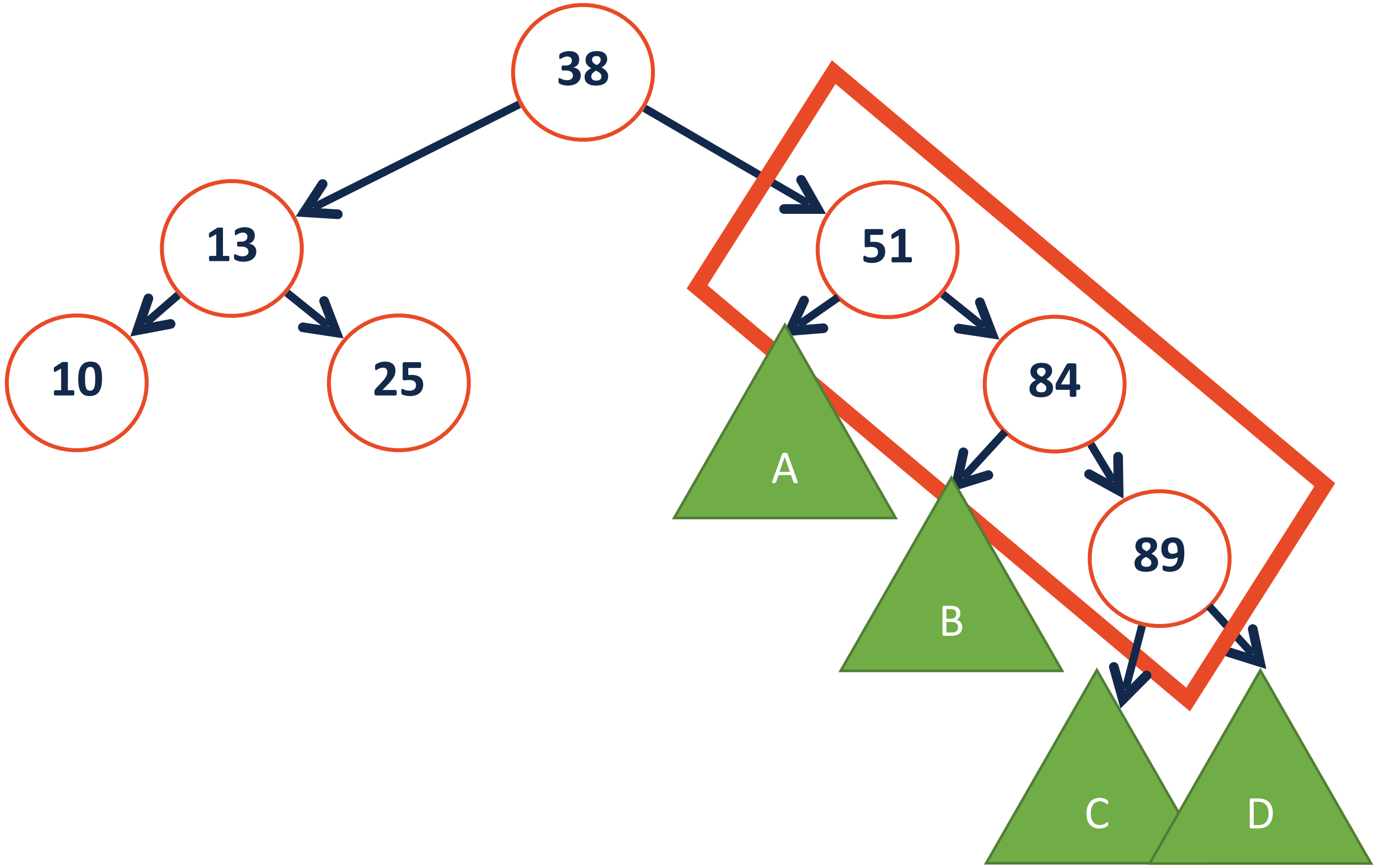
2. Reduce tree height by one

BST Rotations (The AVL Tree)

To begin, lets find the imbalance in the following tree:

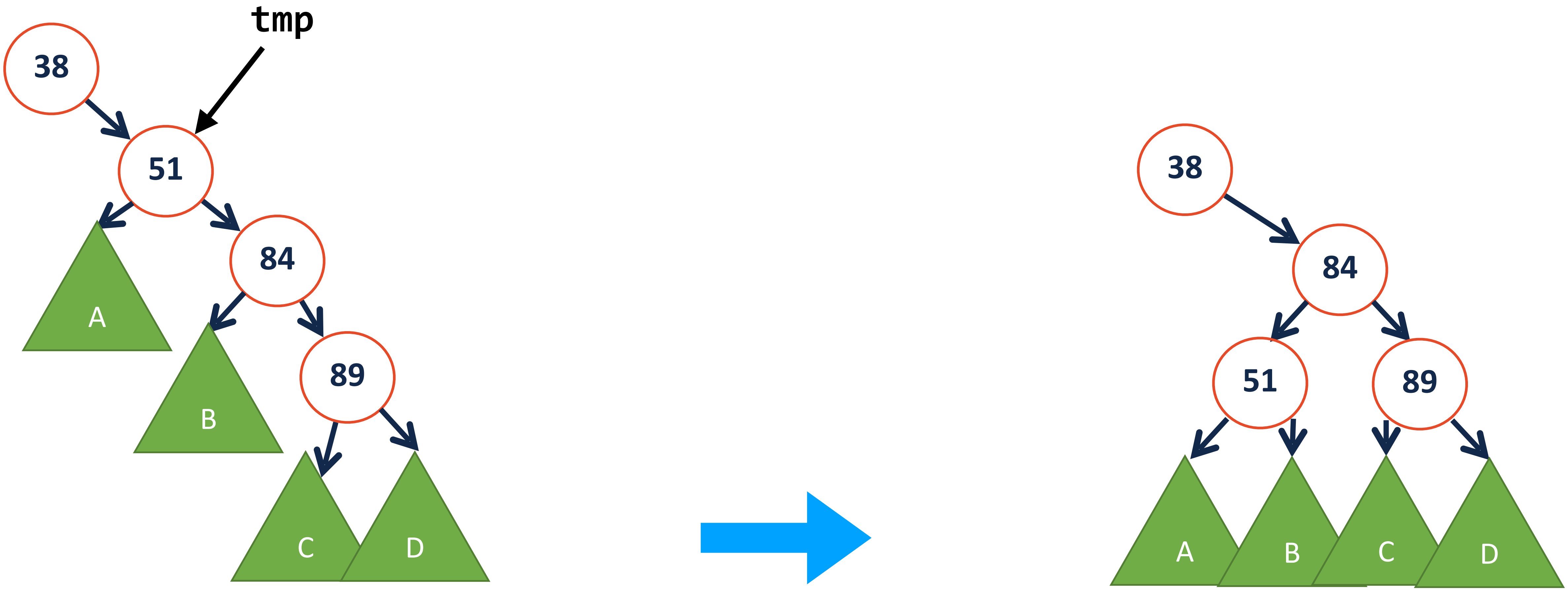


Left Rotation



Left Rotation

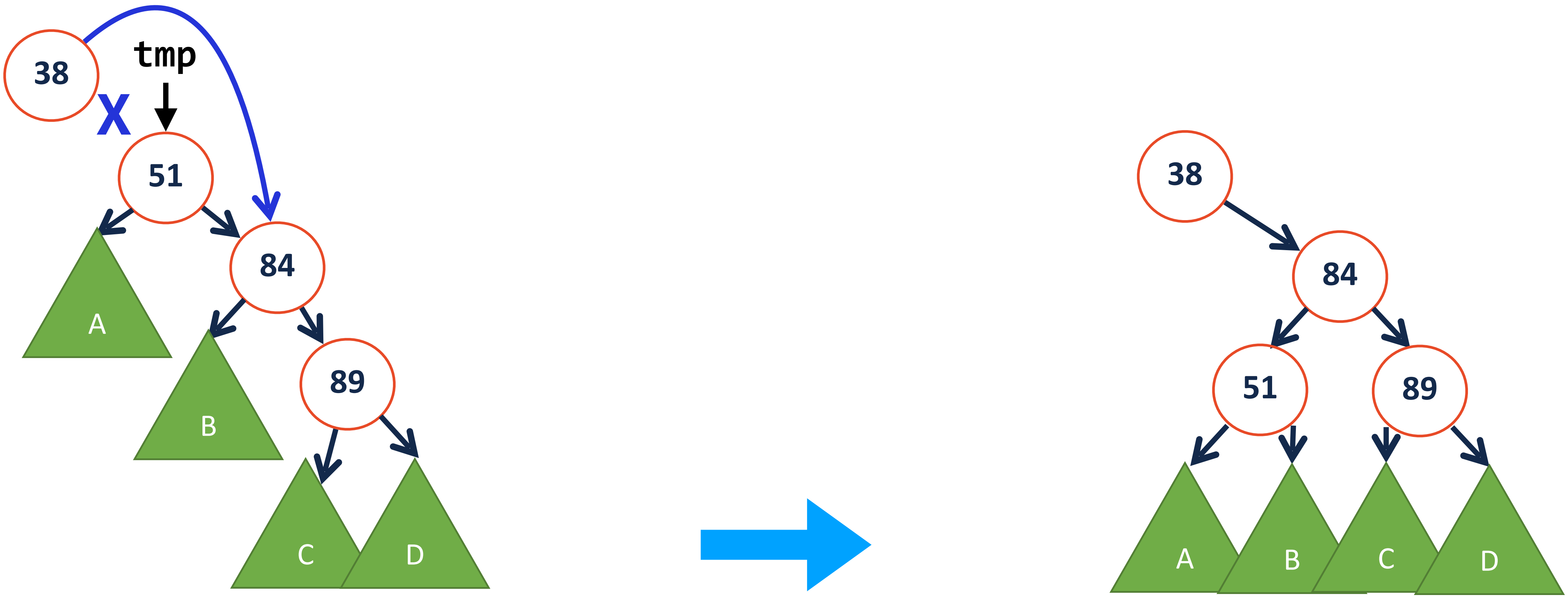
1) Create a tmp pointer to root



Left Rotation

1) Create a tmp pointer to root

2) Update root to point to mid

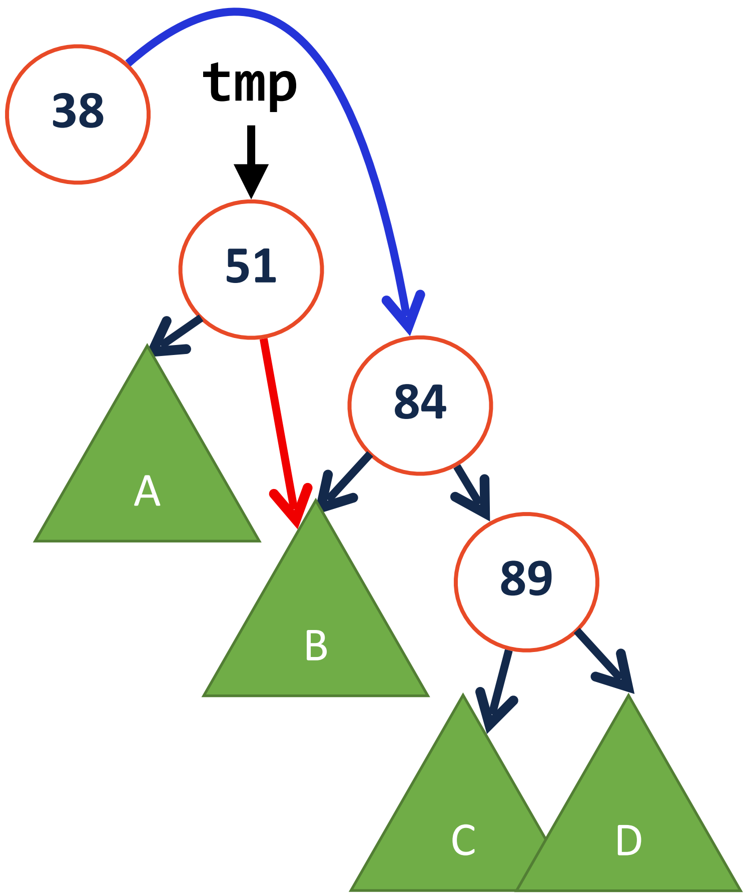


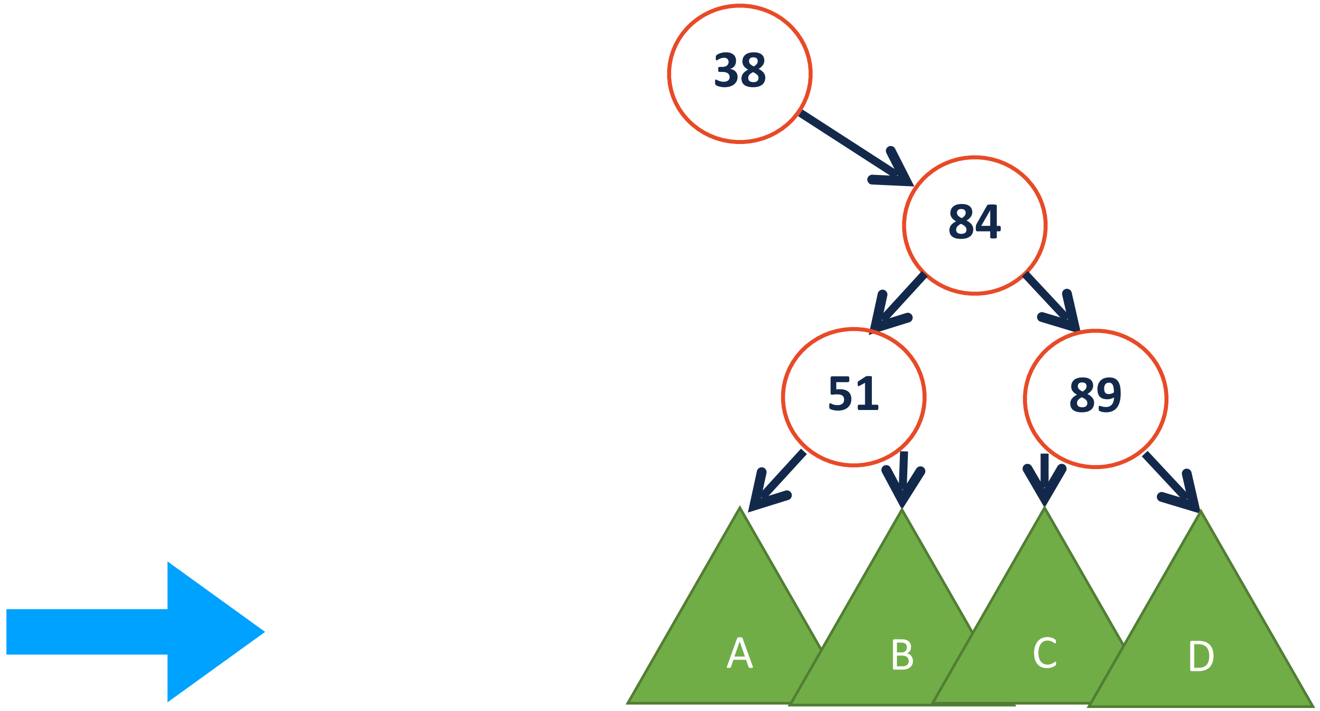
Left Rotation

1) Create a tmp pointer to root

2) Update root to point to mid

3) tmp->right = root->left





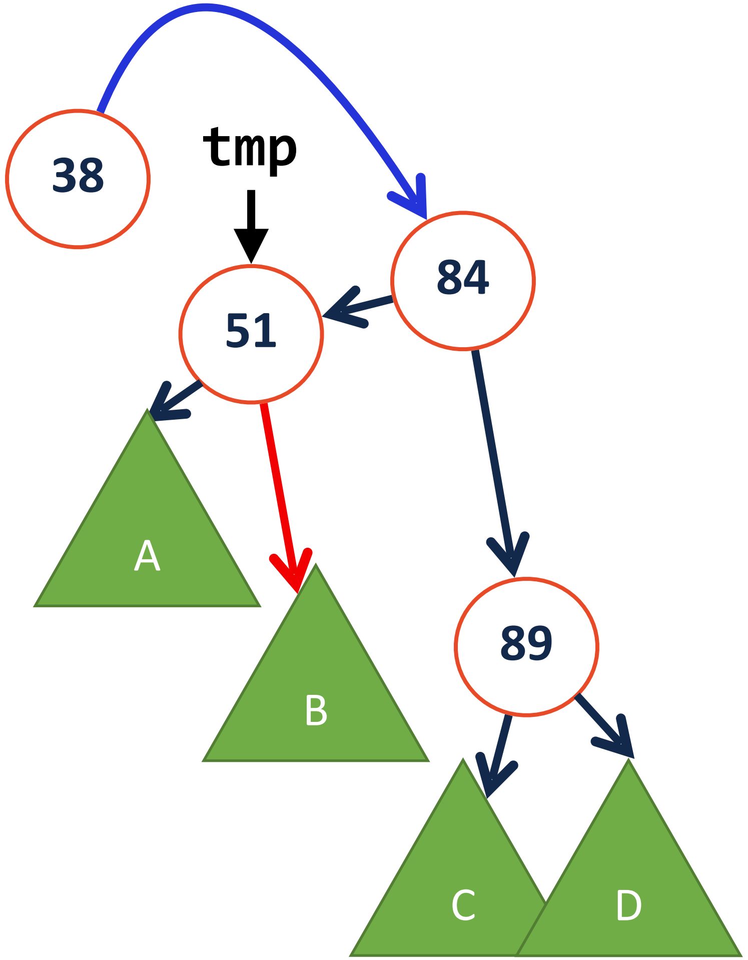
Left Rotation

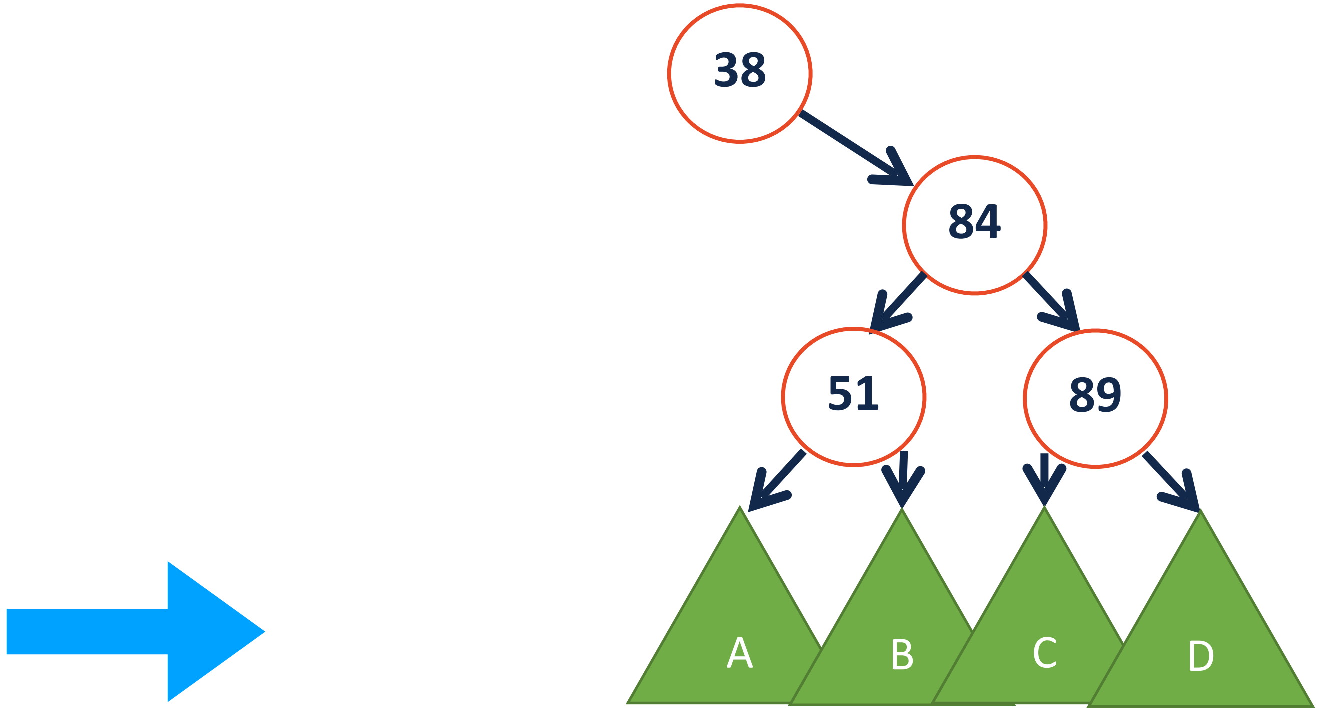
1) Create a tmp pointer to root

2) Update root to point to mid

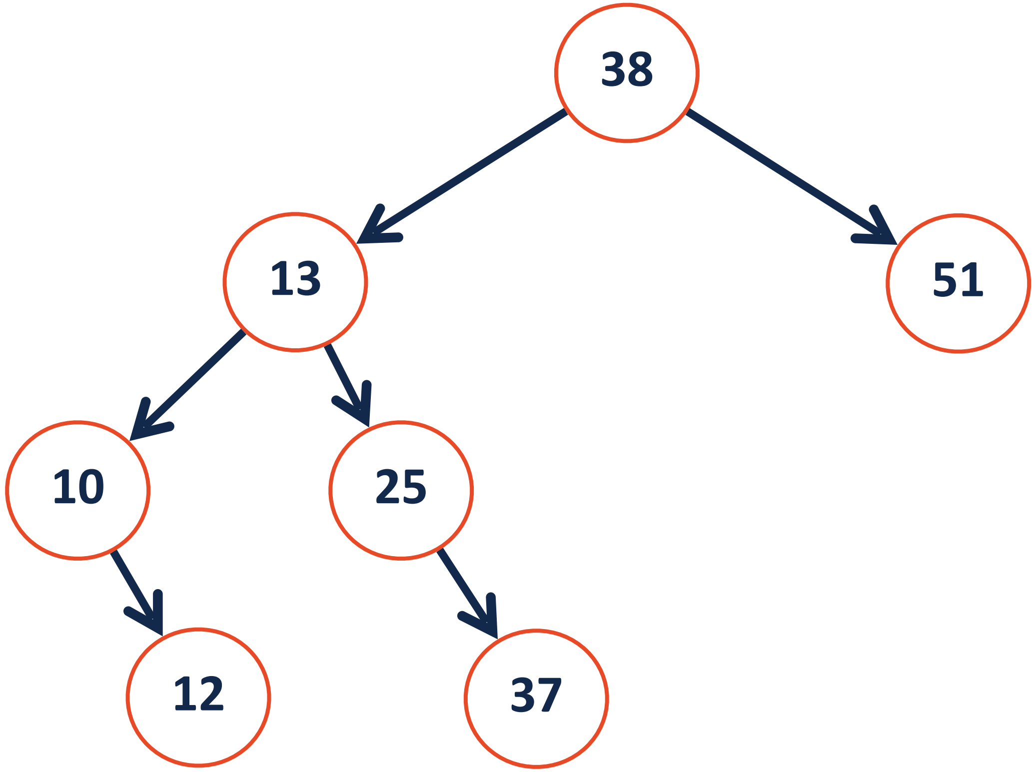
3) tmp->right = root->left

4) root->left = tmp

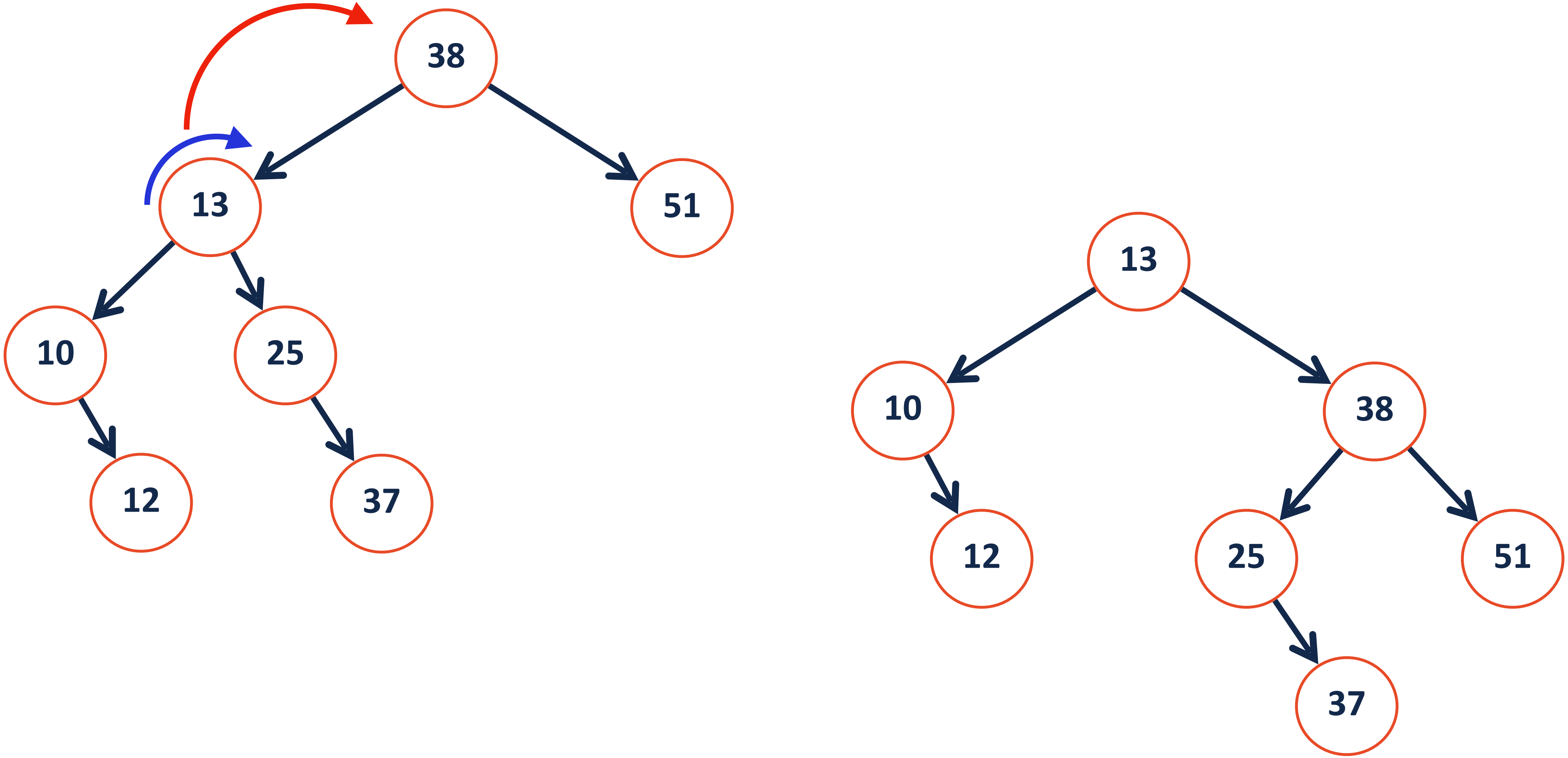




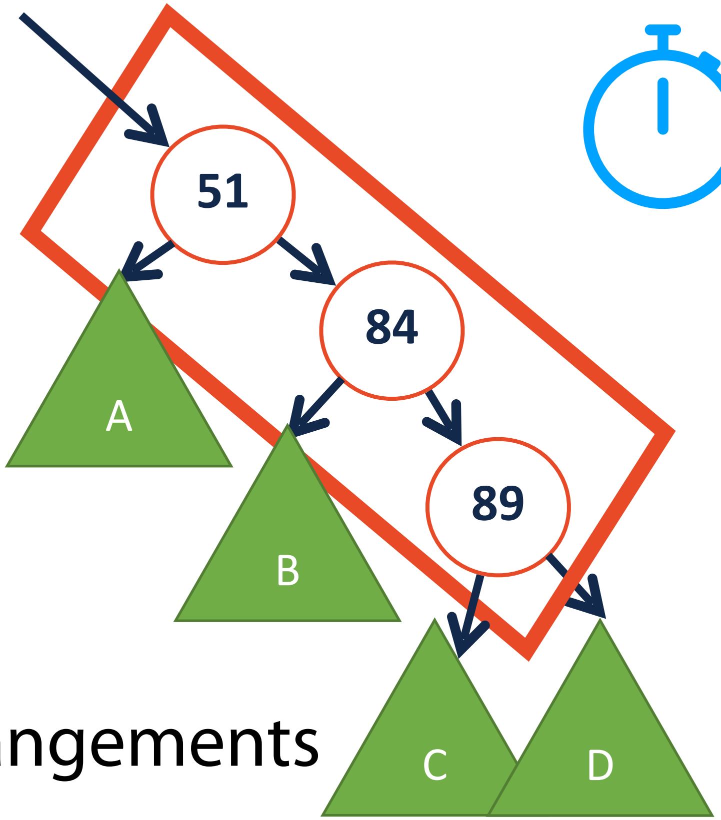
Right Rotation



Right Rotation



Coding AVL Rotations



Two ways of visualizing:

1) Think of an arrow 'rotating' around the center

2) Recognize that there's a concrete order for rearrangements

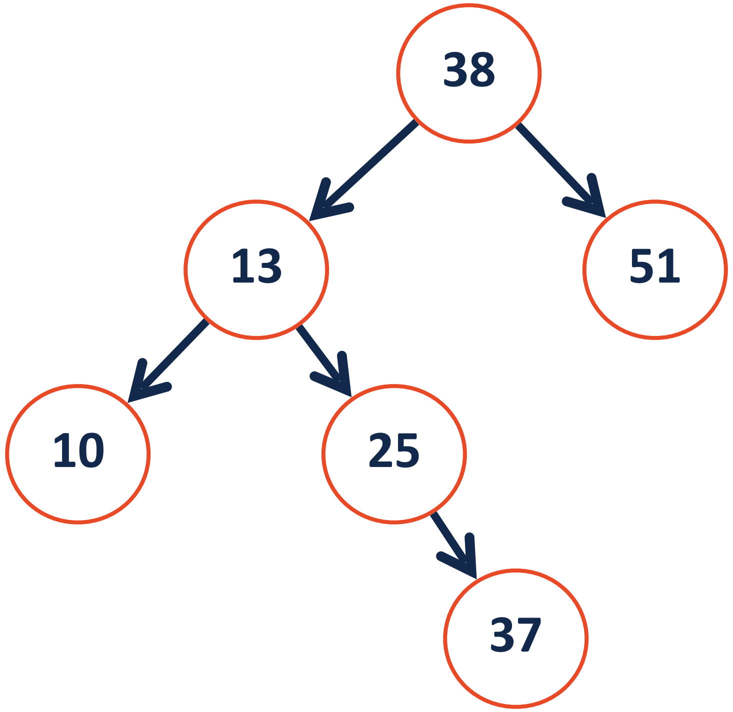
Ex: Unbalanced at current (root) node and need to *rotateLeft*?

Replace current (root) node with it's right child.

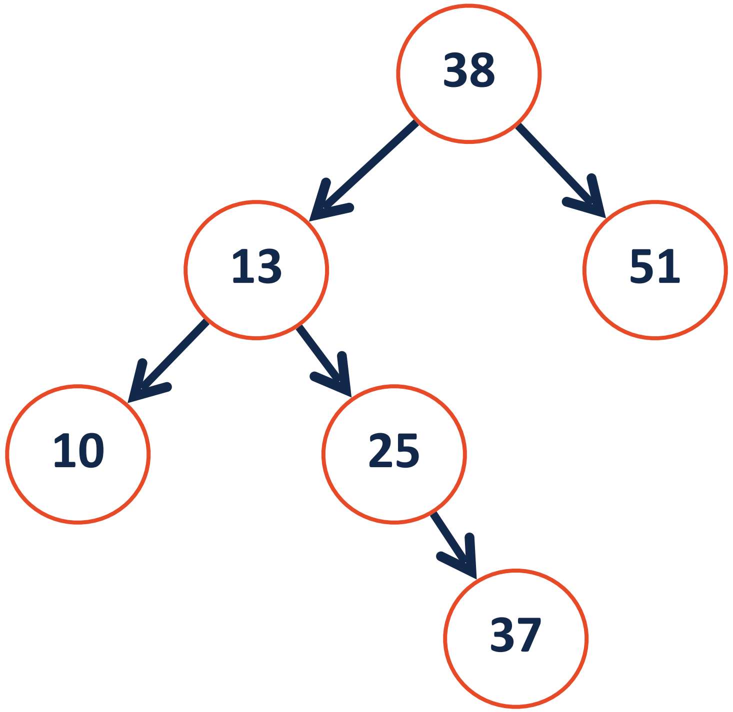
Set the right child's left child to be the current node's right

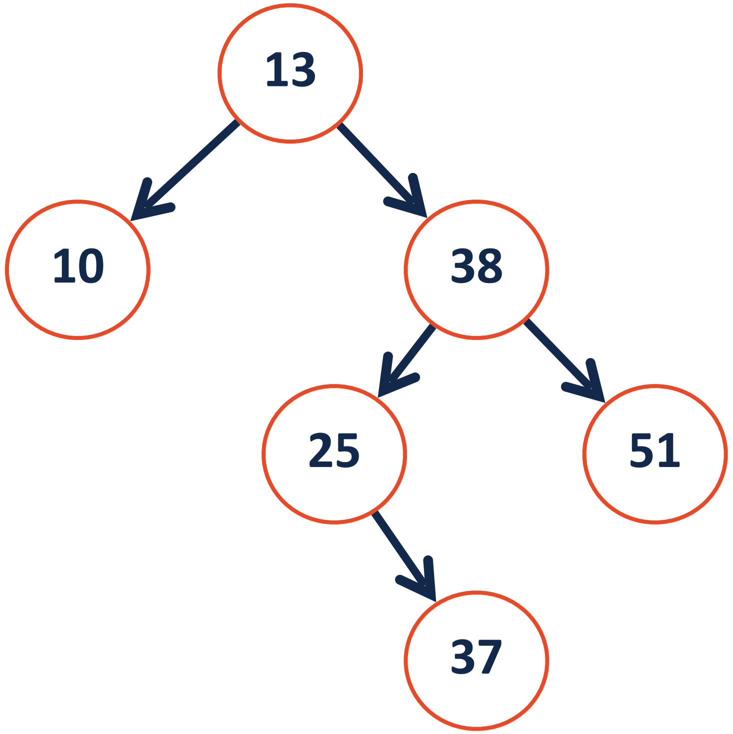
Make the current node the right child's left child

AVL Rotation Practice



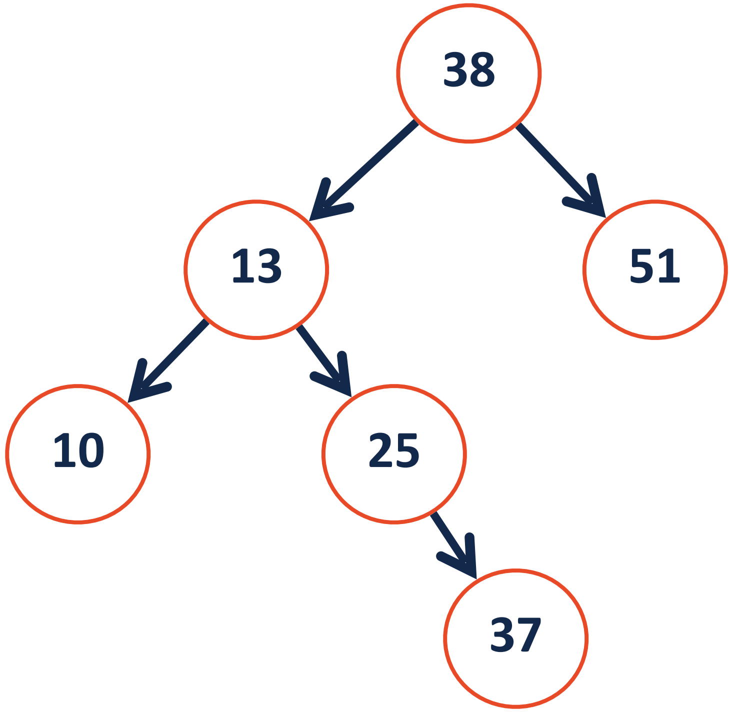
AVL Rotation Practice



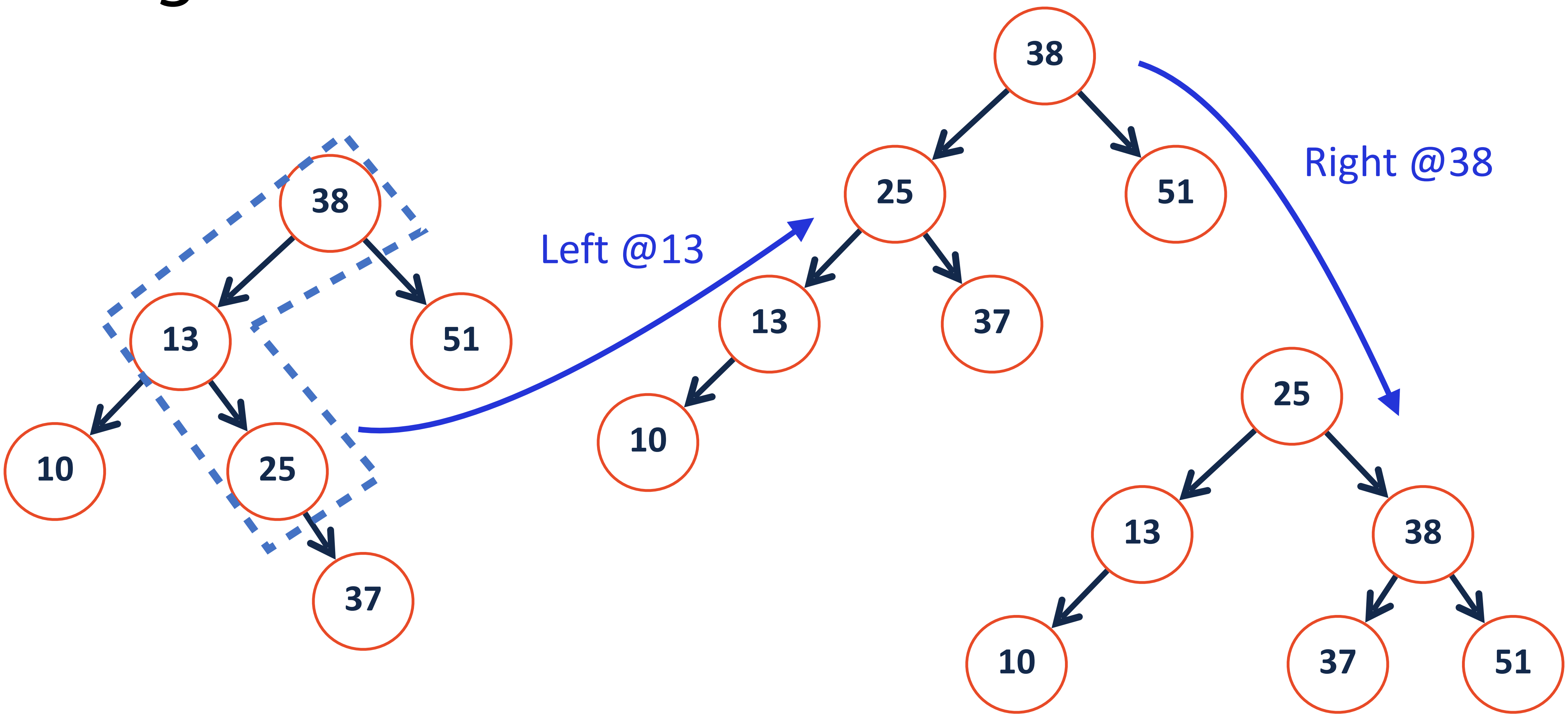


Somethings not quite right…

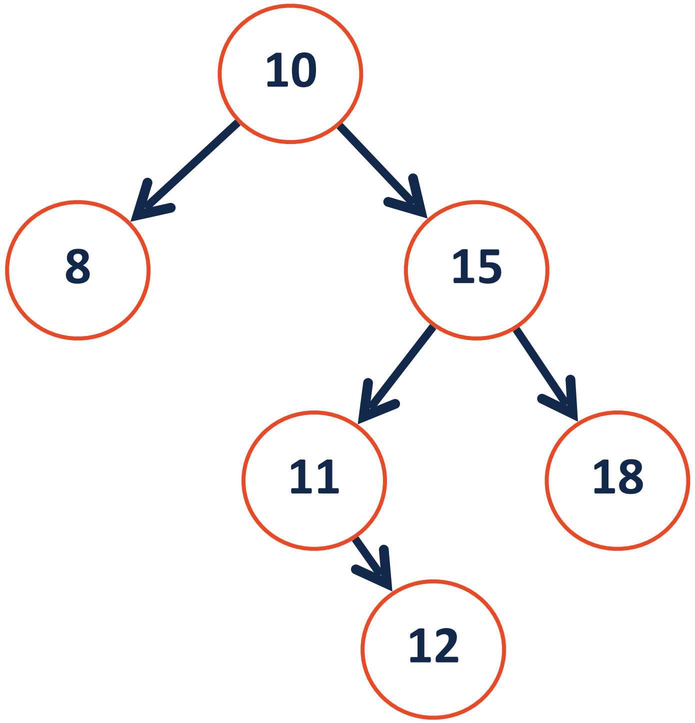
LeftRight Rotation



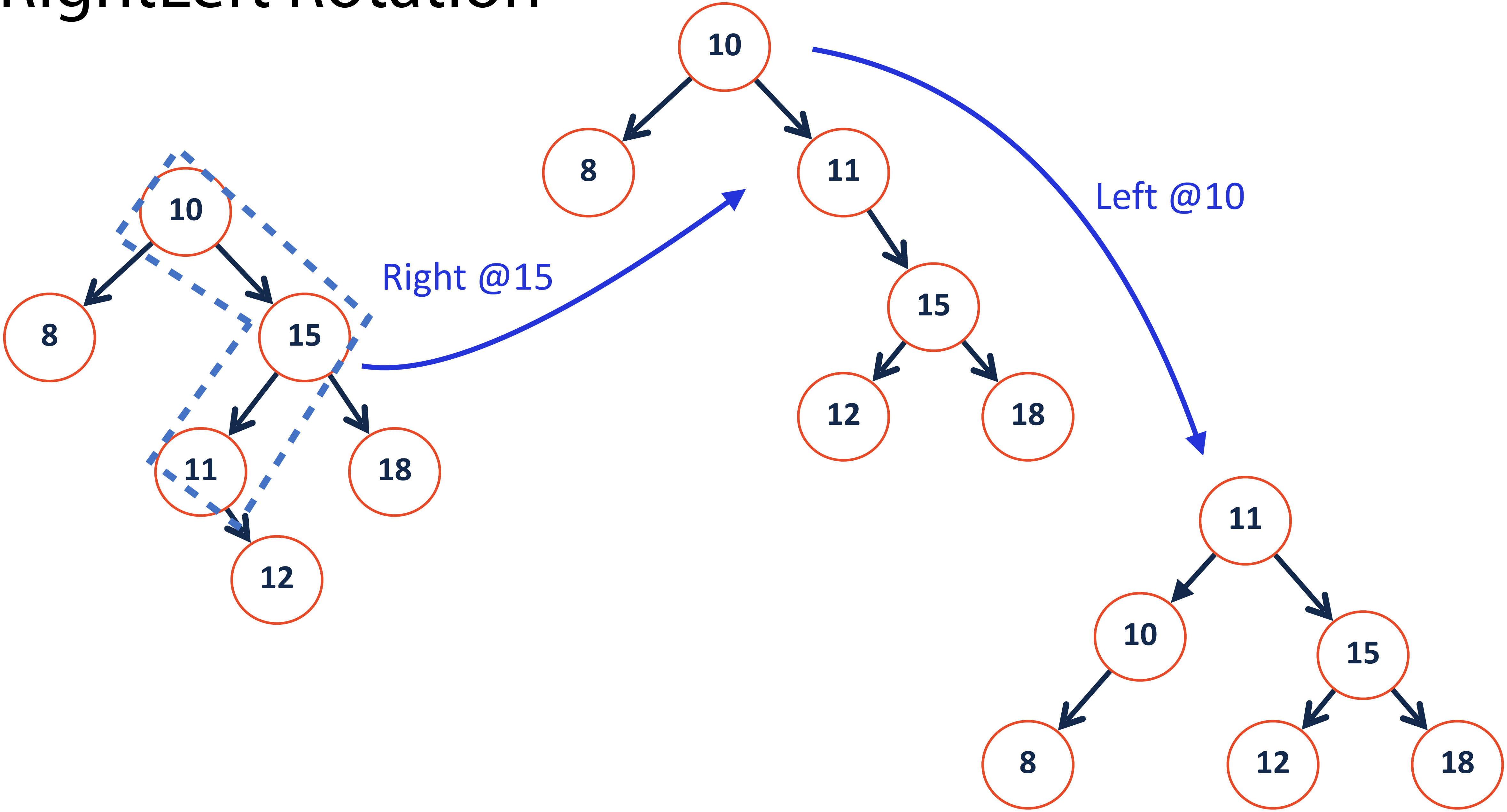
LeftRight Rotation



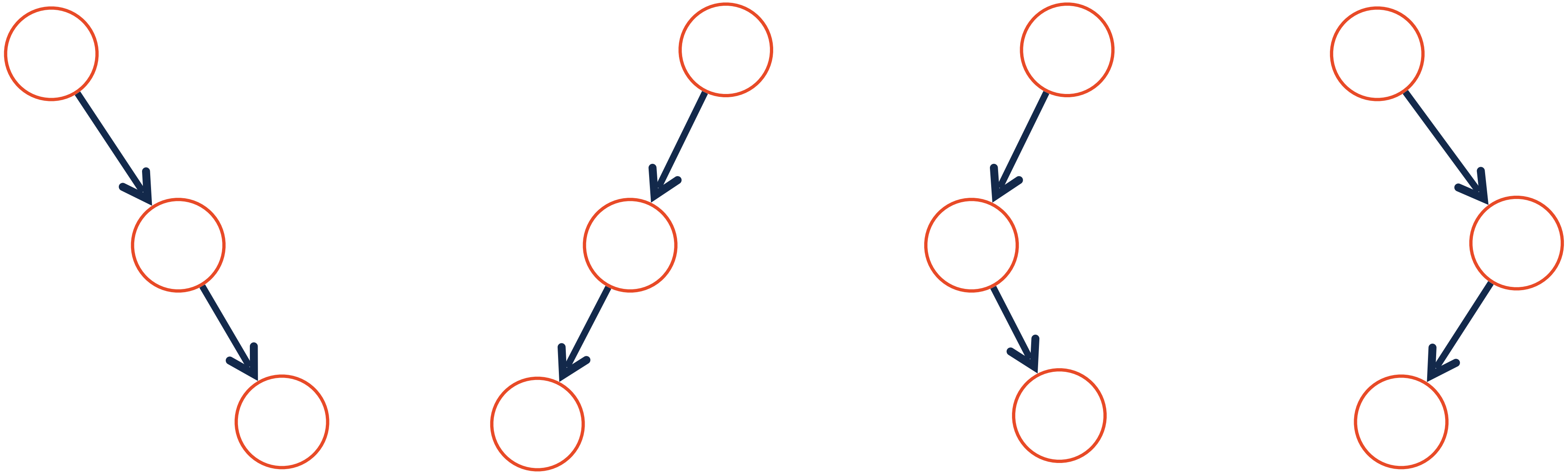
RightLeft Rotation



RightLeft Rotation



AVL Rotations



AVL Rotations

Four kinds of rotations: (L, R, LR, RL)

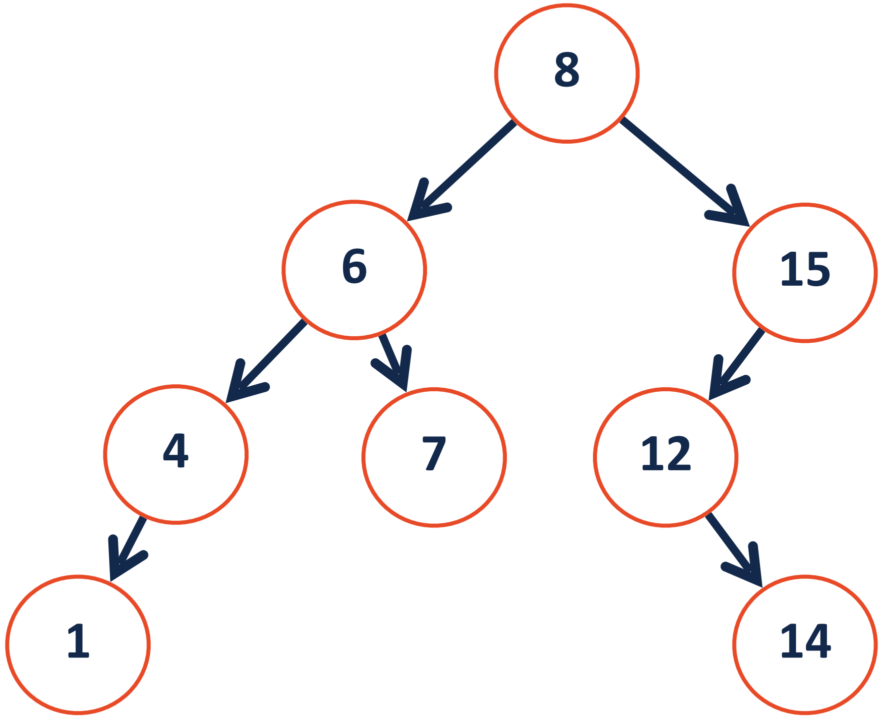
1. All rotations are local (subtrees are not impacted)

2. The running time of rotations are constant

3. The rotations maintain BST property

Goal:

AVL Rotation Practice



AVL vs BST ADT

The AVL tree is a modified binary search tree that rotates **when necessary**

How does the constraint on balance affect the core functions?

Find

Insert

Remove