Algorithms and Data Structures for Data Science
AVL Trees 2

CS 277
Brad Solomon

April 5, 2023

University of Illinois Urbana-Champaign
Department of Computer Science

Mini-Project 2: Sketching

Average: 88%　　Standard Dev: 17.7%　　Median: 96%

Based on grades, things look like they went well

Most justification was reasonable (though occasionally unrealistic)

If you received below a 70% on part 3, consider coming to office hours!

Learning Objectives

Review AVL rotations

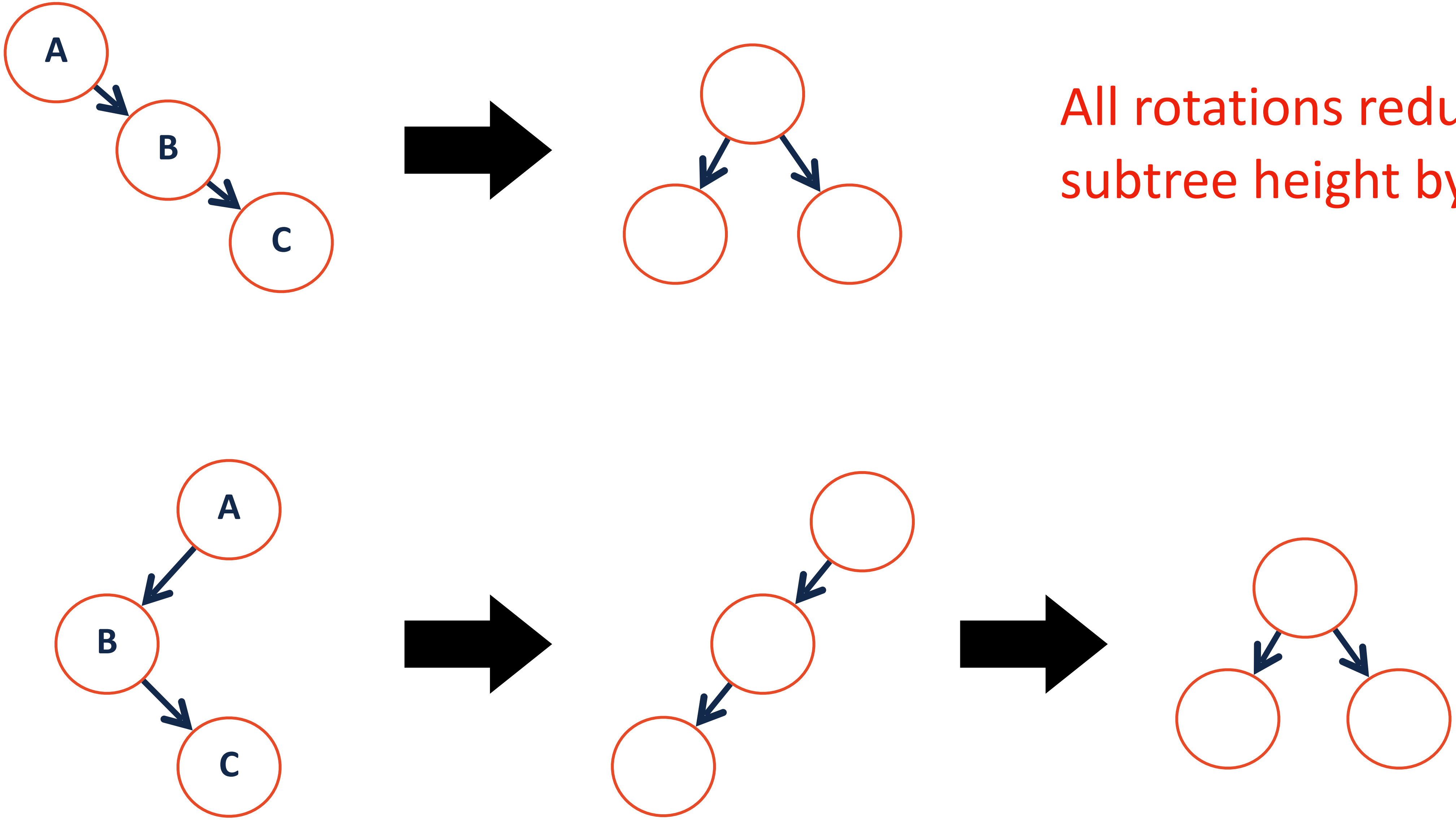
Review discussing AVL functions (remove)

Prove that the AVL tree's height is bounded

AVL Tree Rotations

All rotations are O(1)

All rotations reduce subtree height by one



AVL Insertion

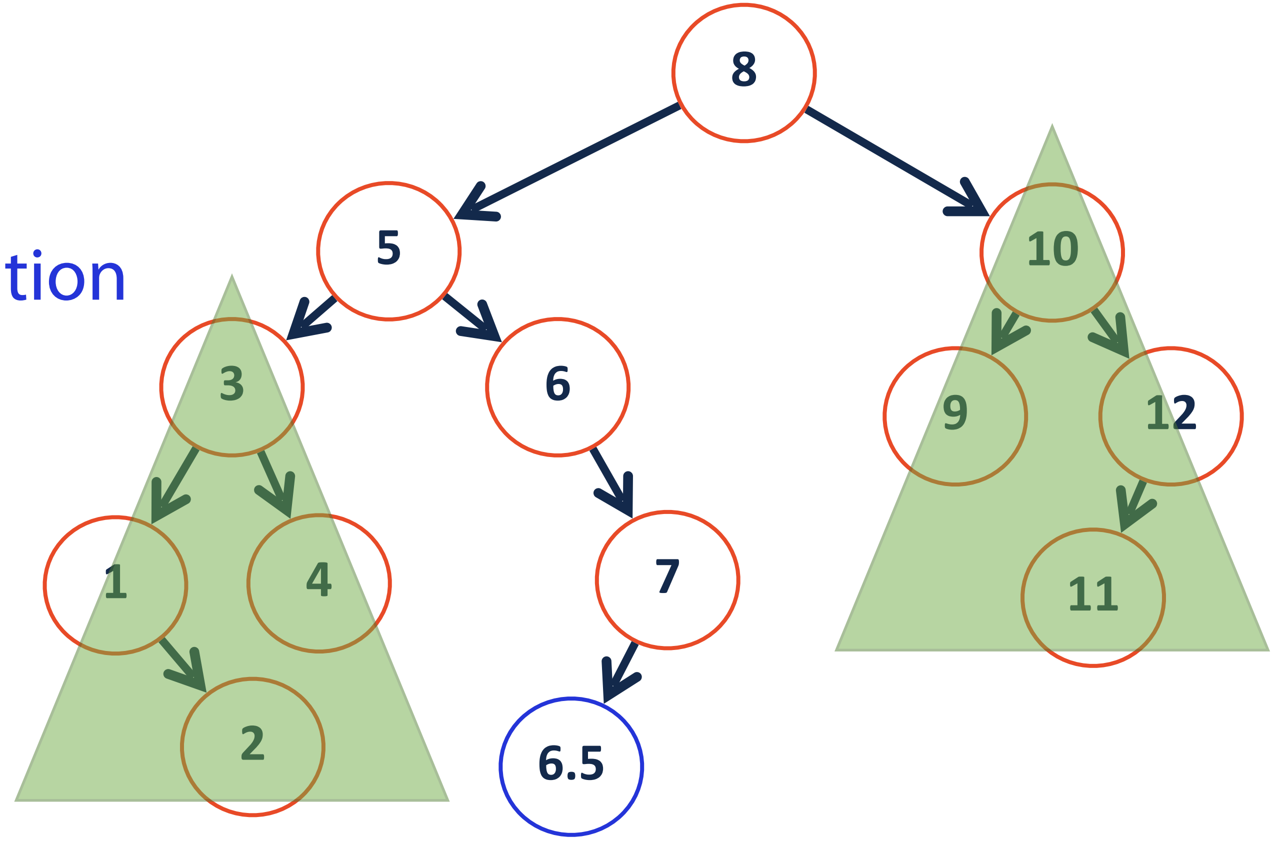
`_insert(6.5)`

Rebalance Function:

1) Checks balance at node

2) If node is unbalanced, pick rotation

3) Perform rotation

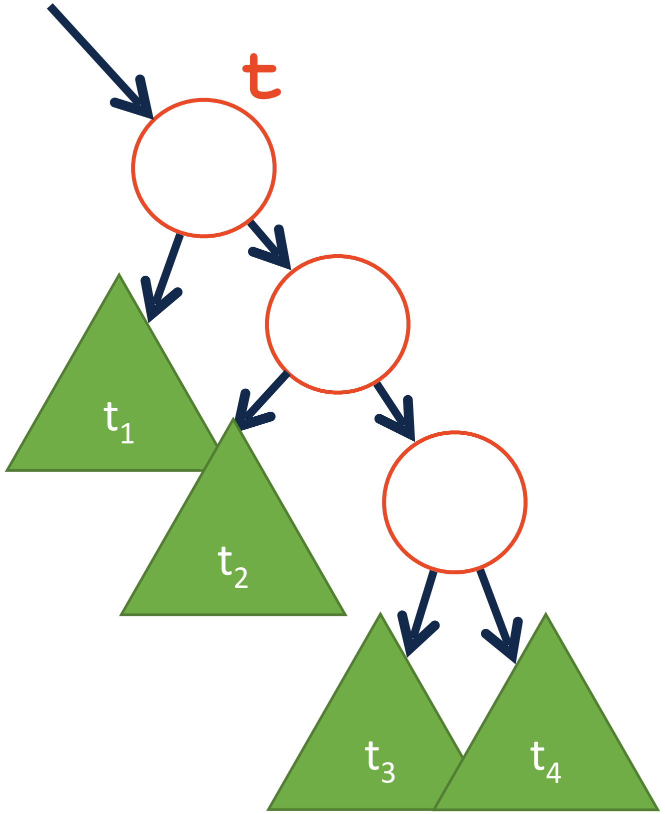



```
def insert_helper(node, key, val):

    ...

    return rebalance(node)
```

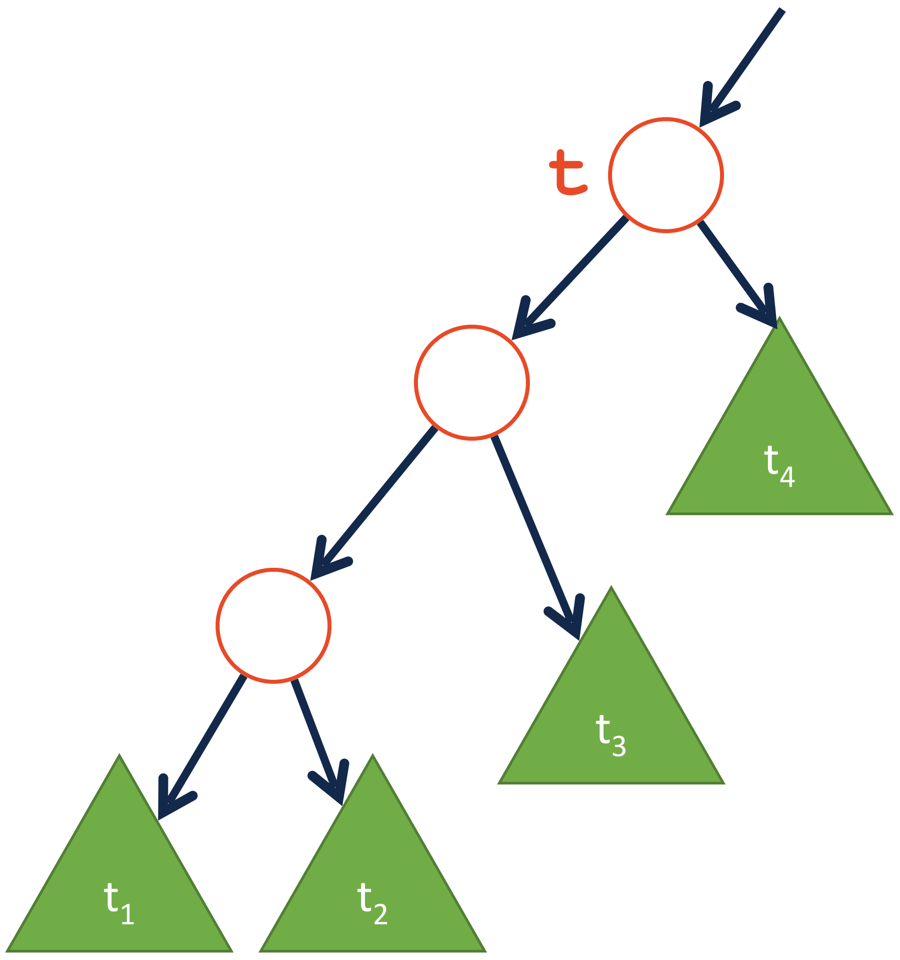
Picking the correct rotation (insert)



Theorem:
If an insertion occurred in subtrees $\mathbf{t_3}$ or $\mathbf{t_4}$ and an imbalance was first detected at **t**, then a _______________ rotation about **t** restores the balance of the tree.

We gauge this by noting the balance factor of **t is** _______ and the balance factor of **t->right** is _______.

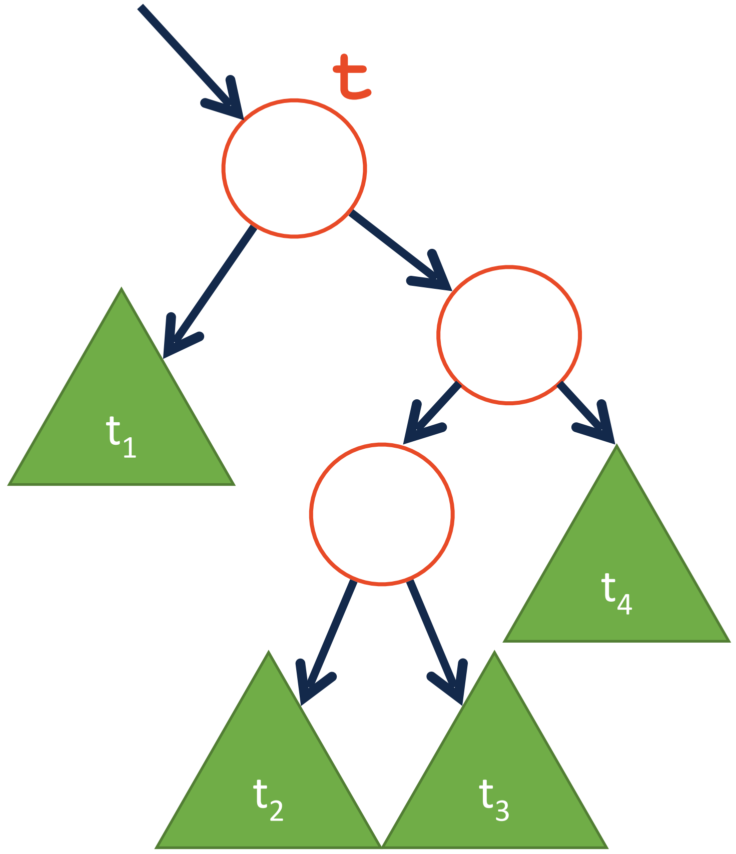
Picking the correct rotation (insert)



Theorem:
If an insertion occurred in subtrees $\mathbf{t_1}$ or $\mathbf{t_2}$ and an imbalance was first detected at **t**, then a _______________ rotation about **t** restores the balance of the tree.

We gauge this by noting the balance factor of **t is** _______ and the balance factor of **t->left** is _______.

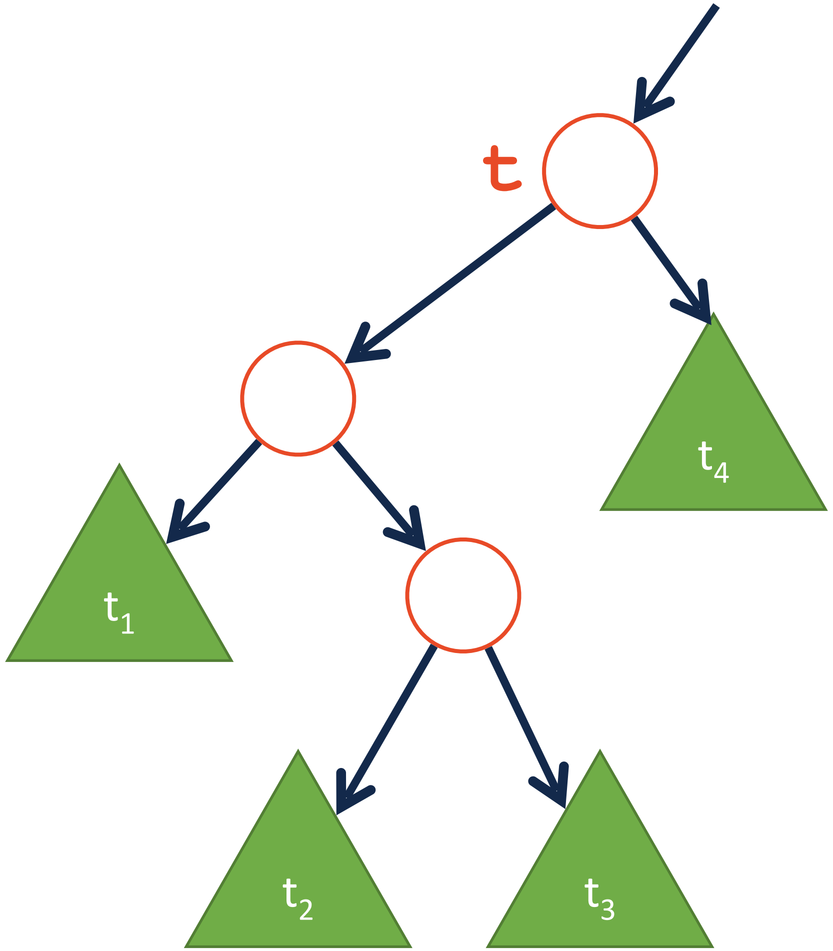
Picking the correct rotation (insert)



Theorem:
If an insertion occurred in subtrees $\mathbf{t_2}$ or $\mathbf{t_3}$ and an imbalance was first detected at **t**, then a ______________ rotation about **t** restores the balance of the tree.

We gauge this by noting the balance factor of **t is** _______ and the balance factor of **t->right** is _______.

Picking the correct rotation (insert)

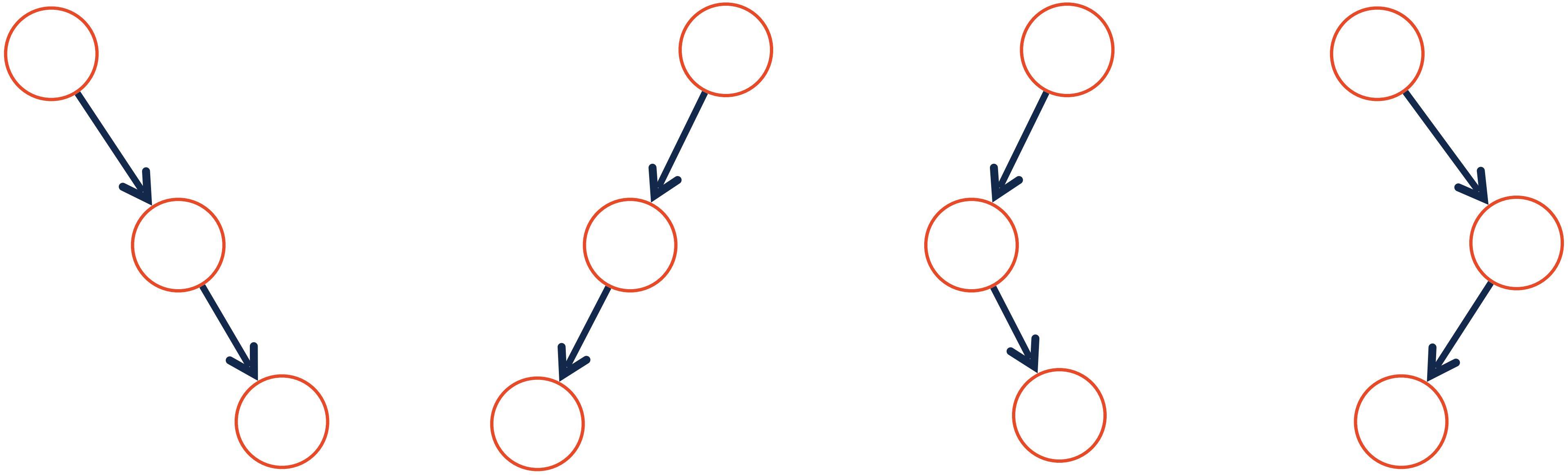


Theorem:
If an insertion occurred in subtrees $\mathbf{t_2}$ or $\mathbf{t_3}$ and an imbalance was first detected at **t**, then a ______________ rotation about **t** restores the balance of the tree.

We gauge this by noting the balance factor of **t is** _______ and the balance factor of **t->left** is _______.

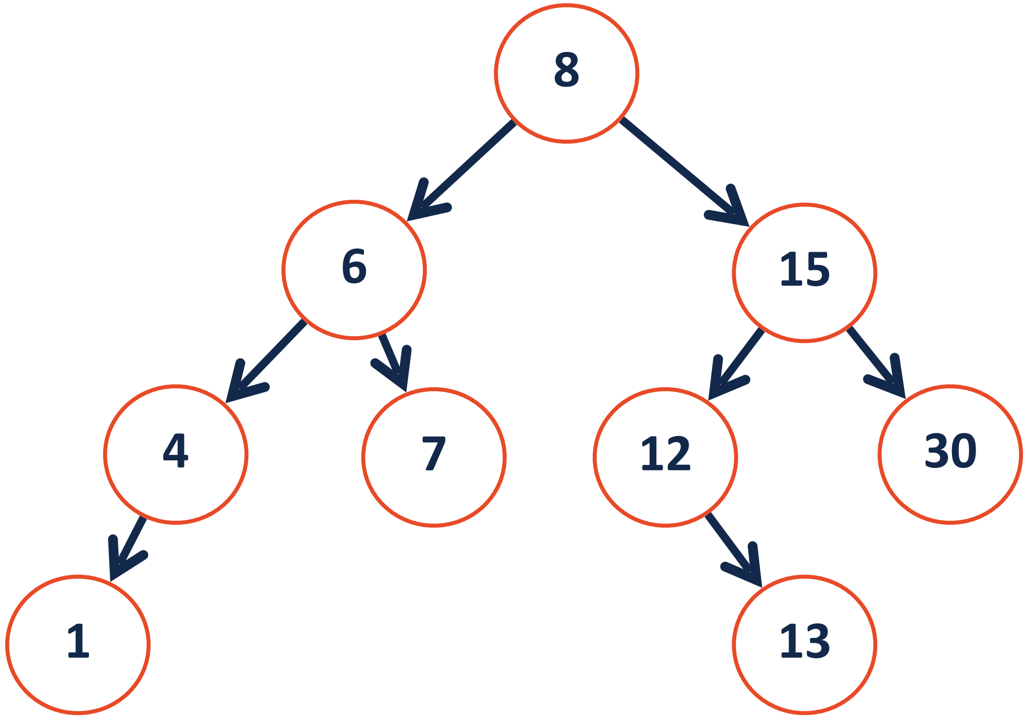
AVL Rotations

Left **Right** **LeftRight** **RightLeft**



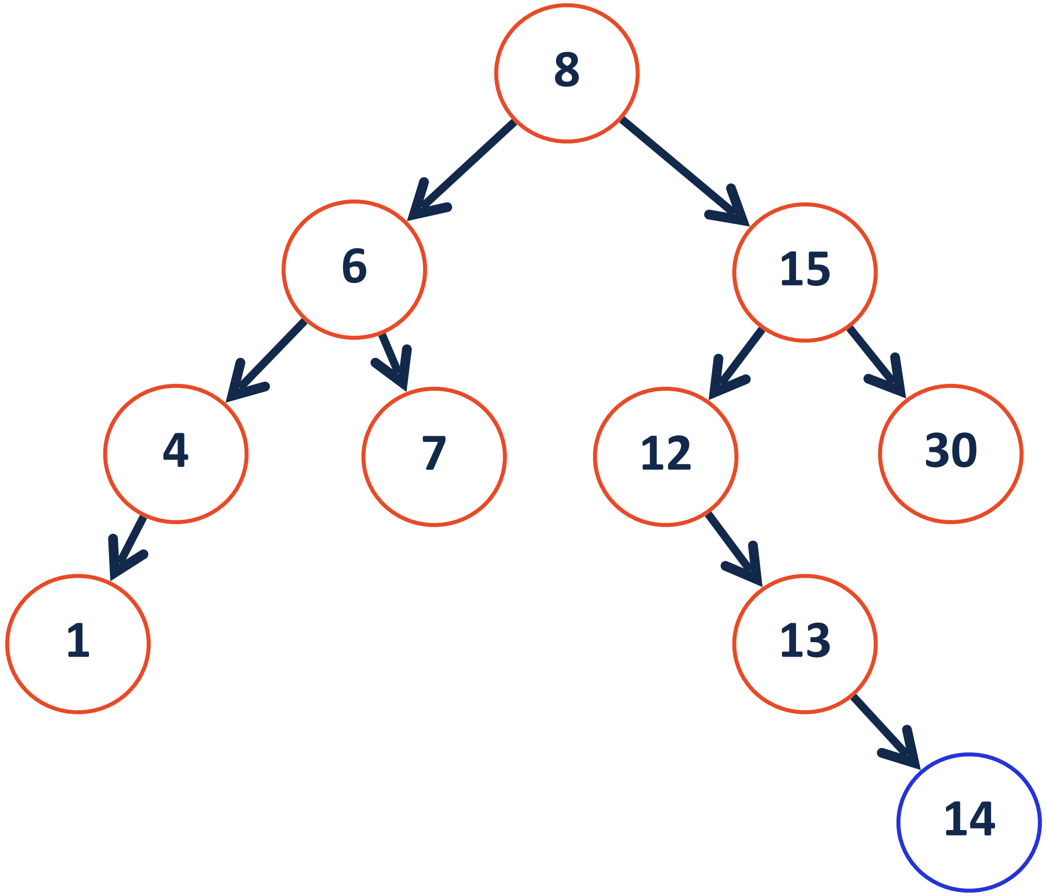
AVL Insertion Practice

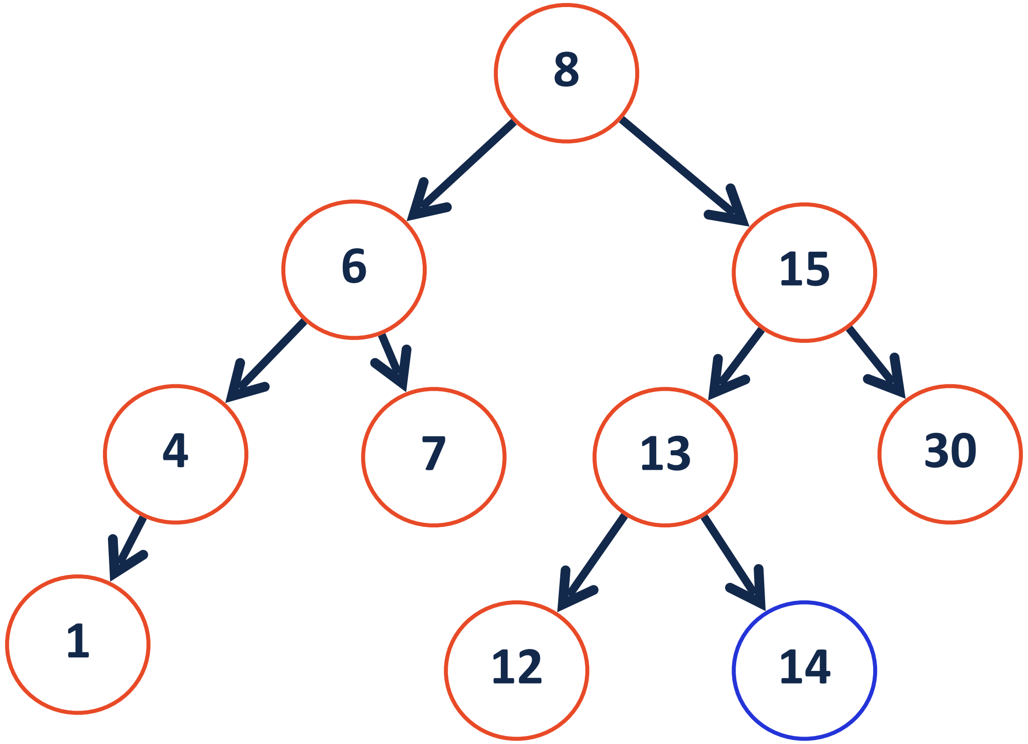
`_insert(14)`



AVL Insertion Practice

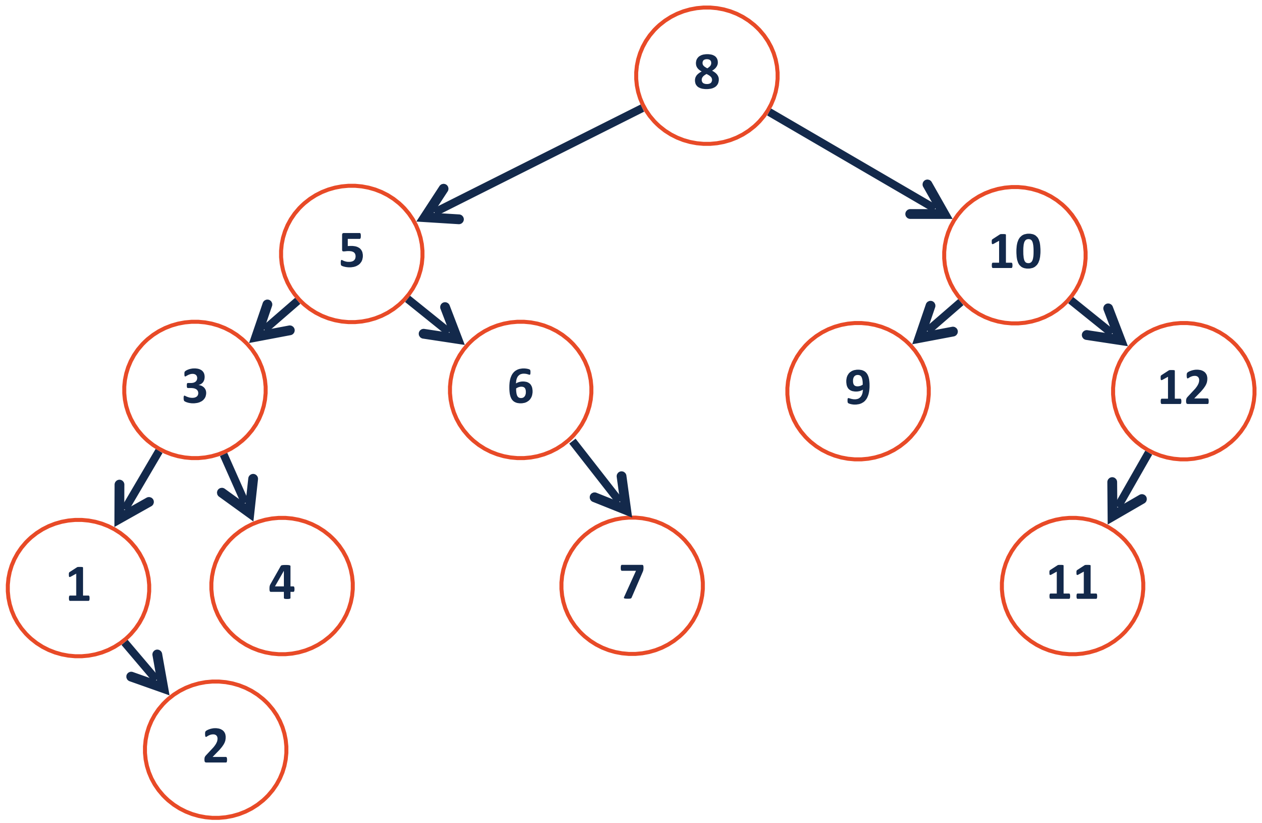
`_insert(14)`





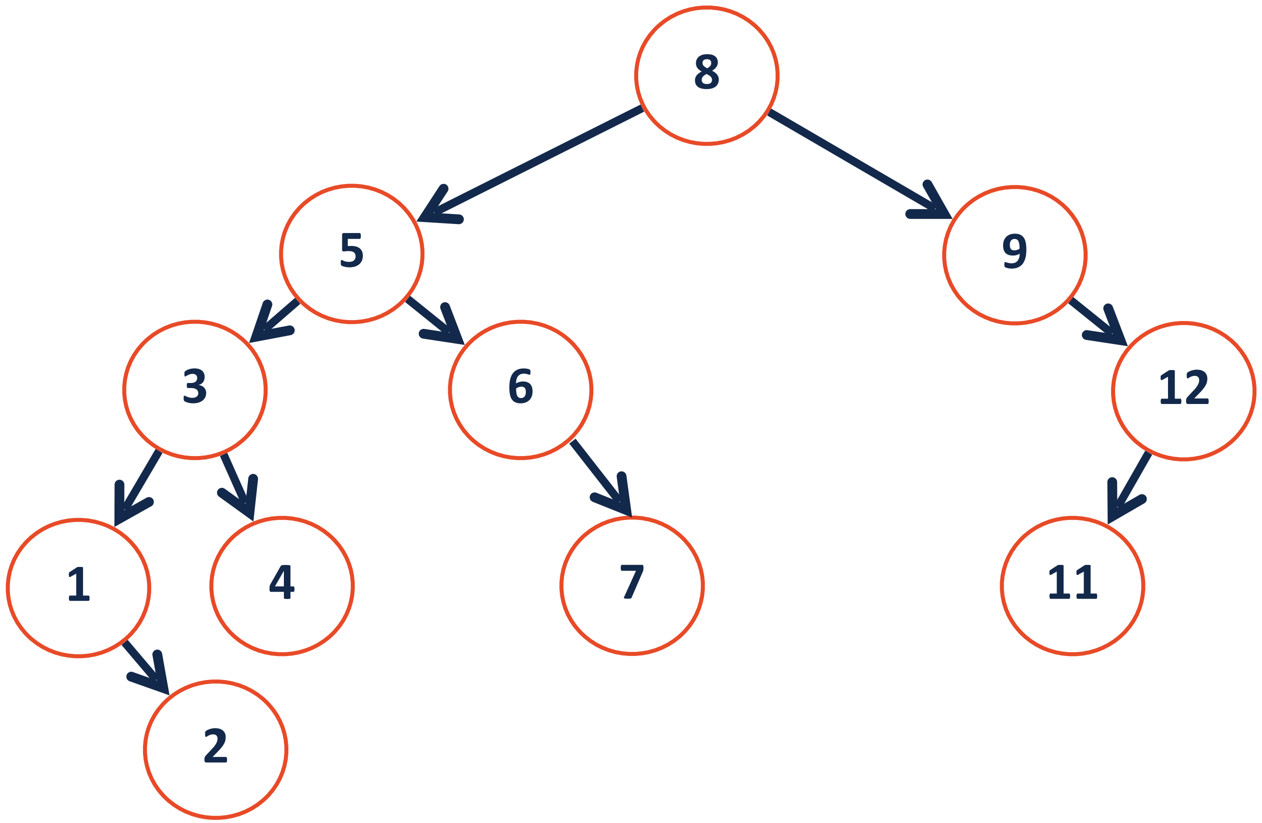
AVL Remove

`_remove(10)`



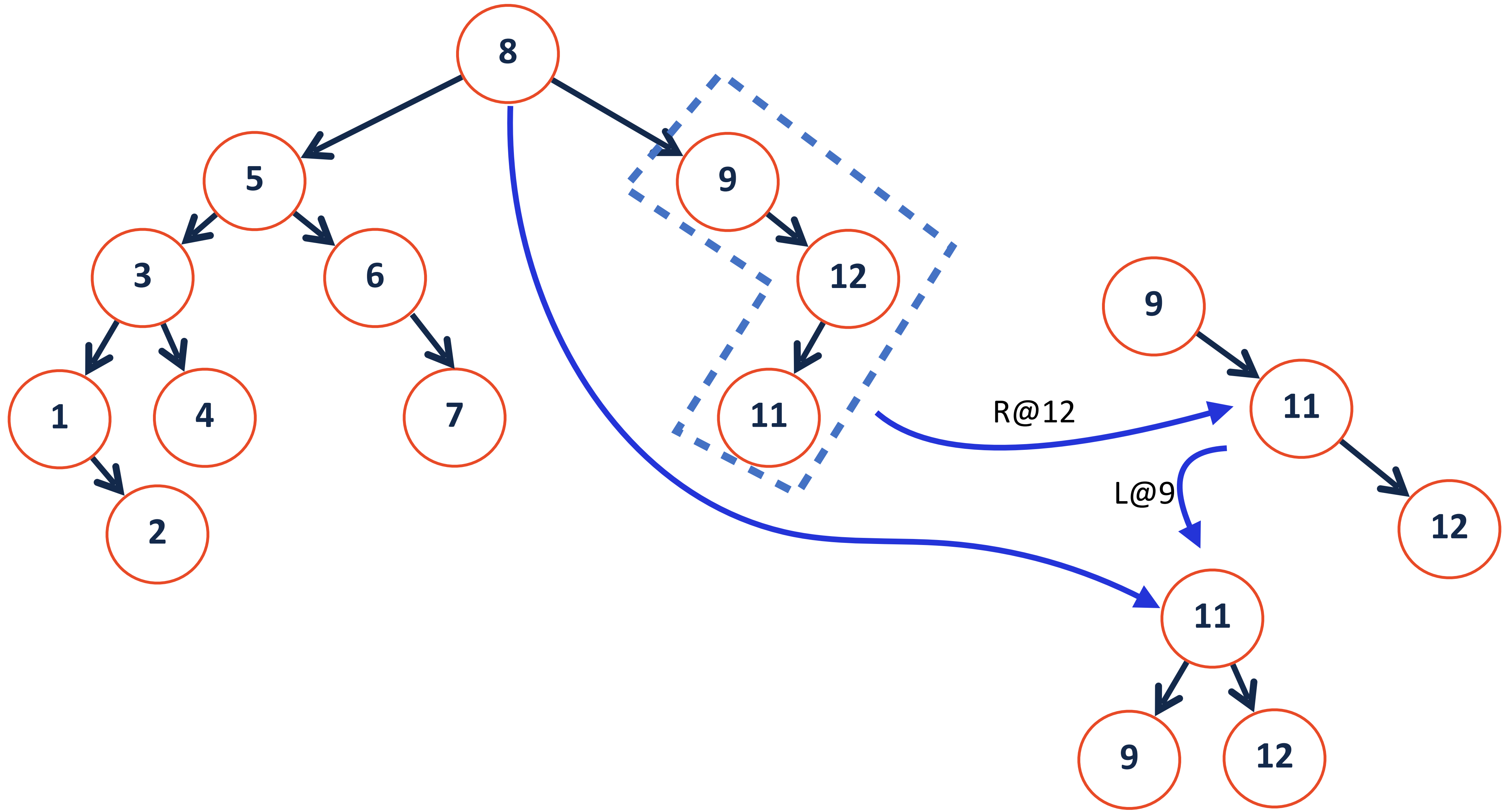
AVL Remove

`_remove(10)`



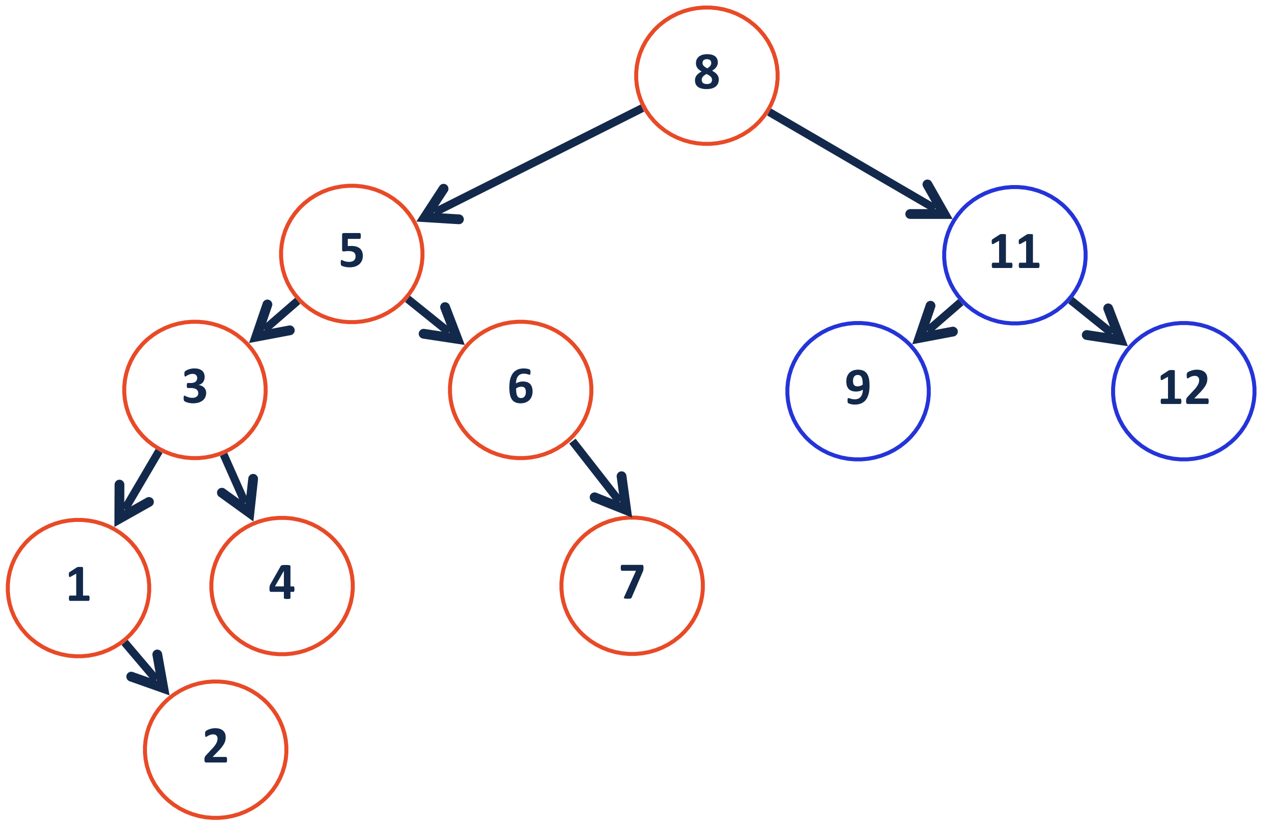
AVL Remove

`_remove(10)`



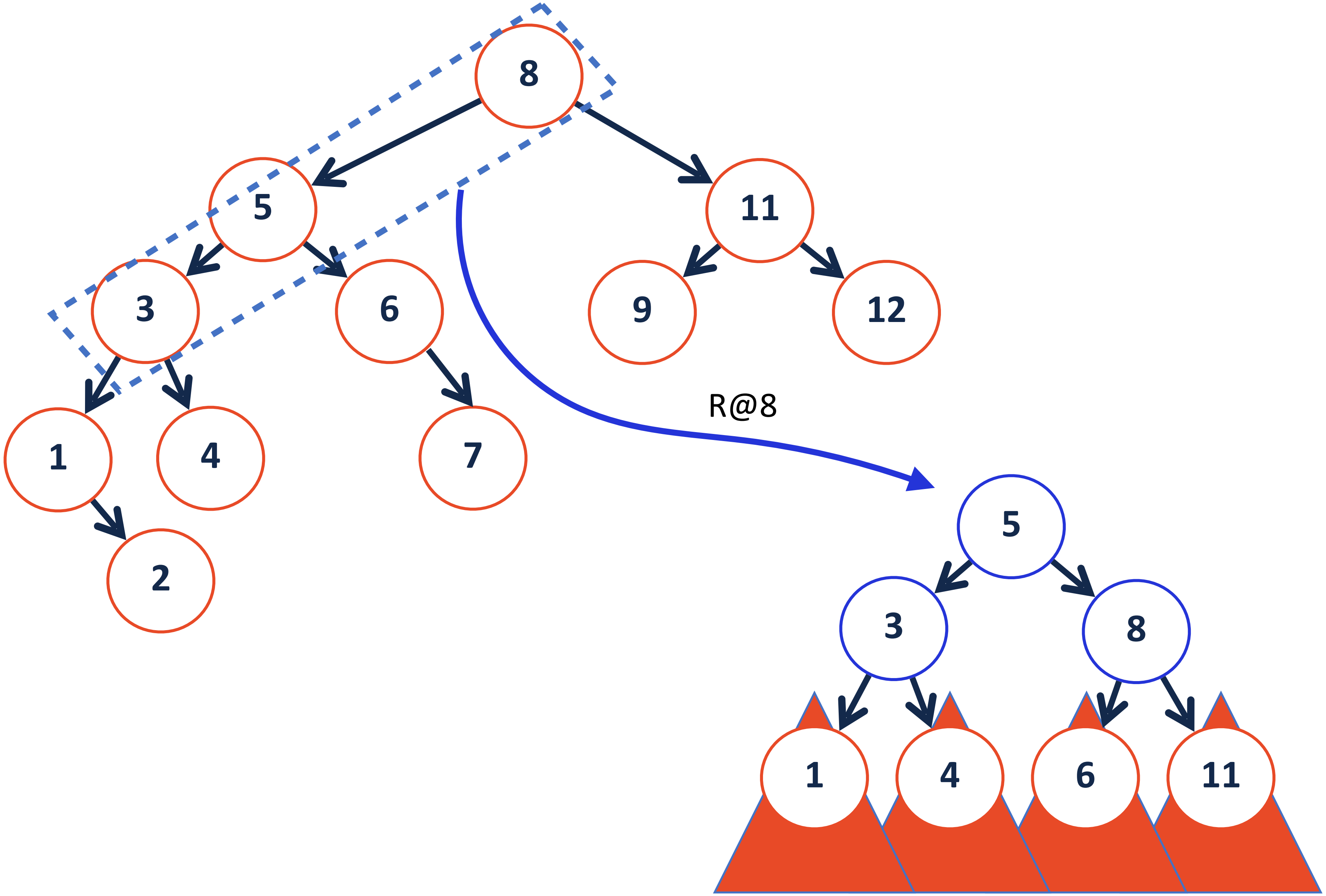
AVL Remove

`_remove(10)`



AVL Remove

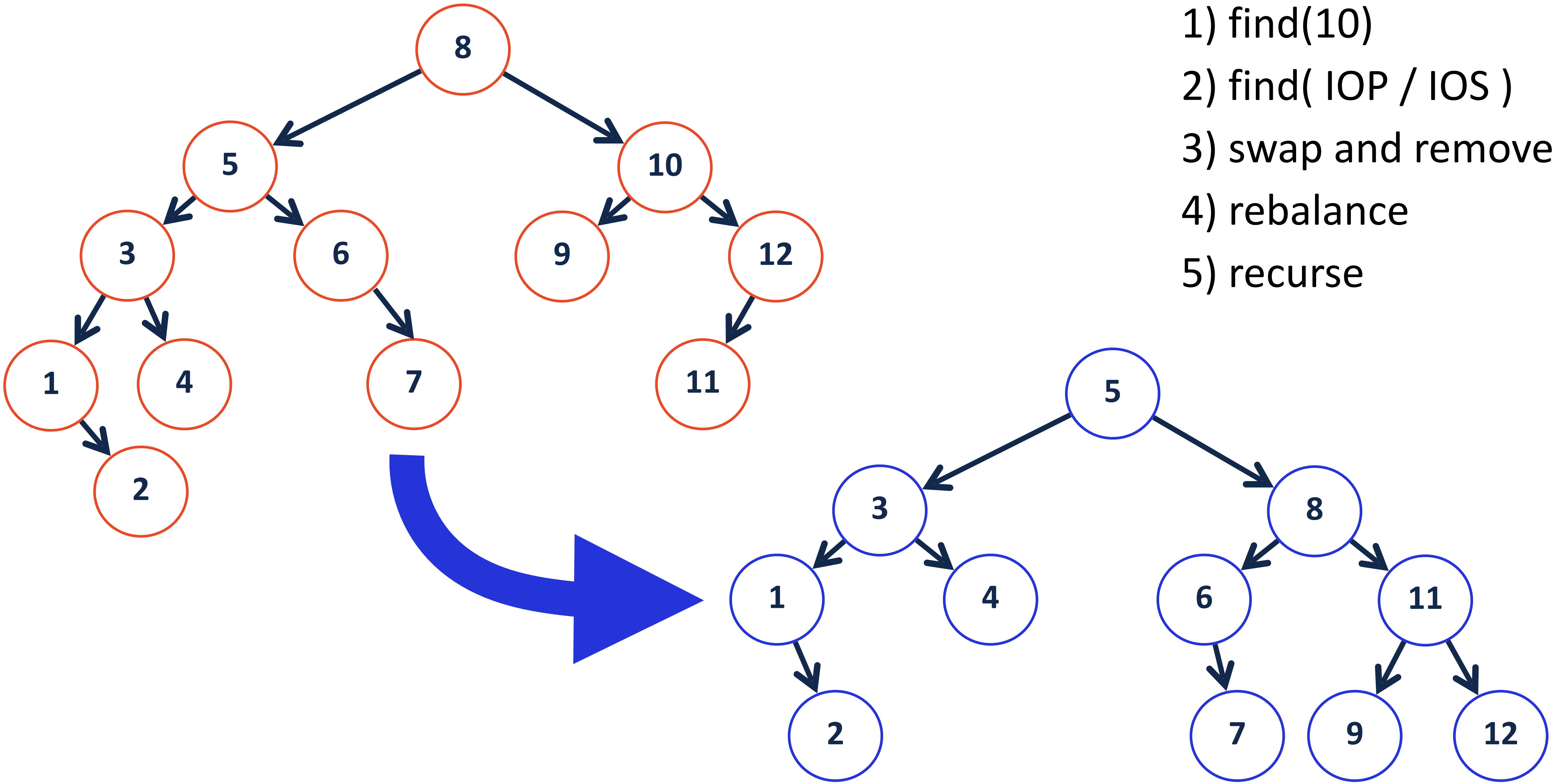
`_remove(10)`



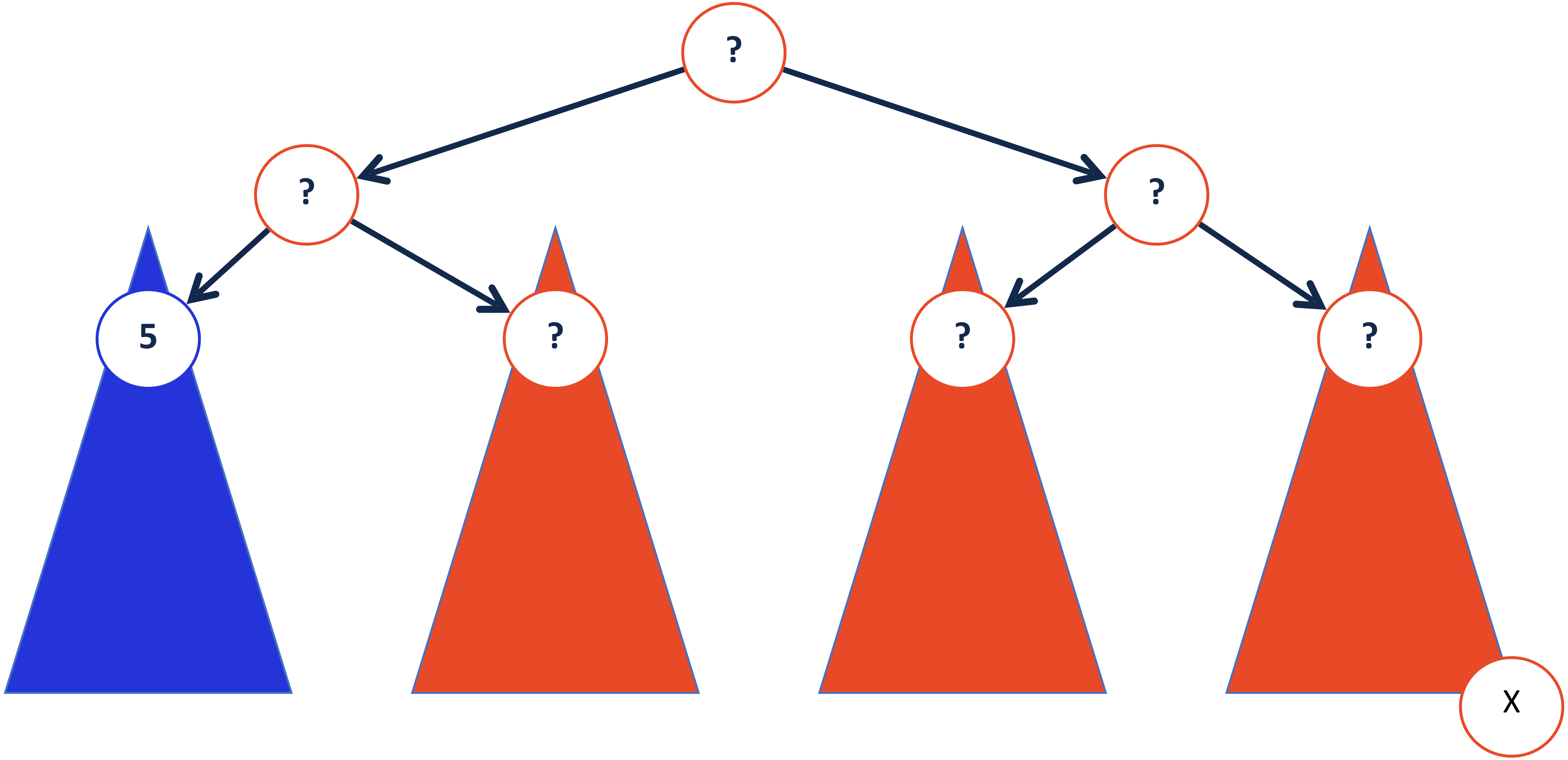
AVL Remove

_remove(10)

1) find(10)
2) find(IOP / IOS)
3) swap and remove
4) rebalance
5) recurse



AVL Remove



AVL Tree Analysis

For an AVL tree of height h:

Find runs in: ___________.

Insert runs in: ___________.

Remove runs in: ___________.

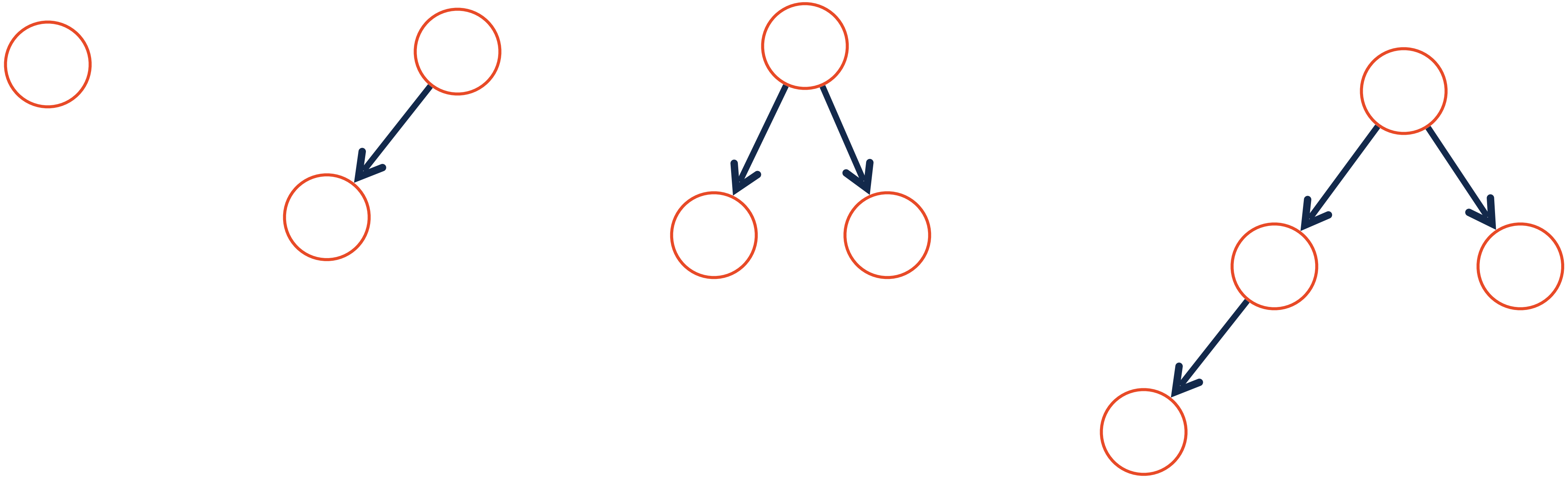
Claim: The height of the AVL tree with n nodes is: ___________.

AVL Tree Height

Claim: The height of an AVL tree with n nodes is bounded by $O(\log n)$

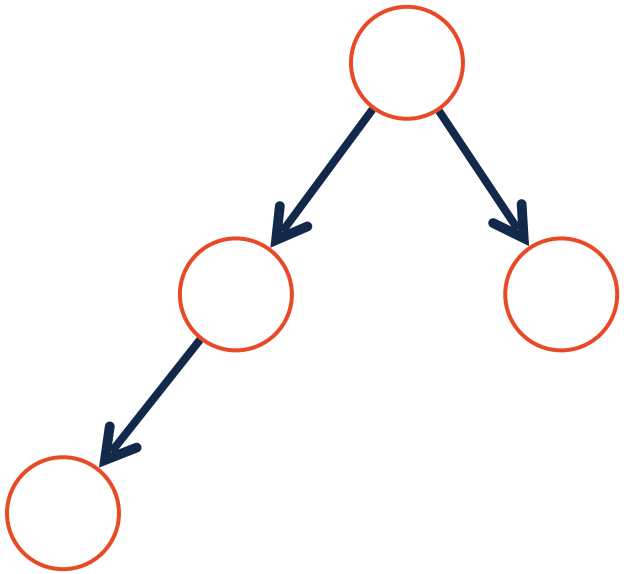
AVL Tree Height

Claim: The height of an AVL tree with n nodes is bounded by $O(\log n)$



AVL Tree Height

If we assume a balanced tree is $O(\log n)$, does insertion break this?

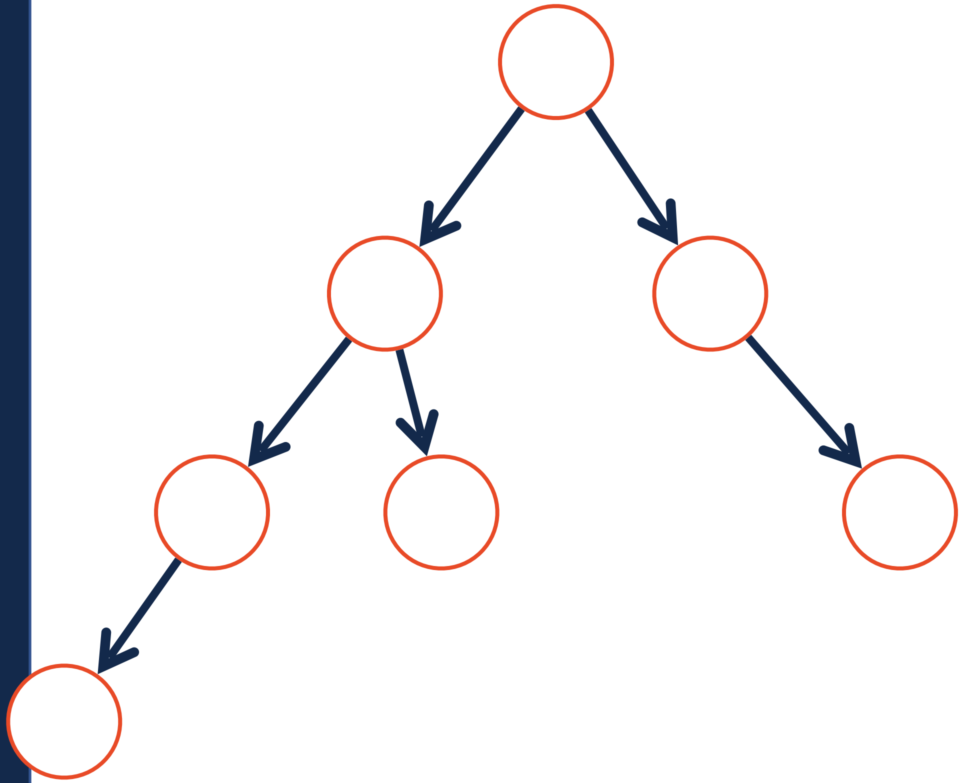


Insertion increases height by ___________.

How many rotations performed: ___________.

AVL Tree Height

If we assume a balanced tree is $O(\log n)$, does remove break this?

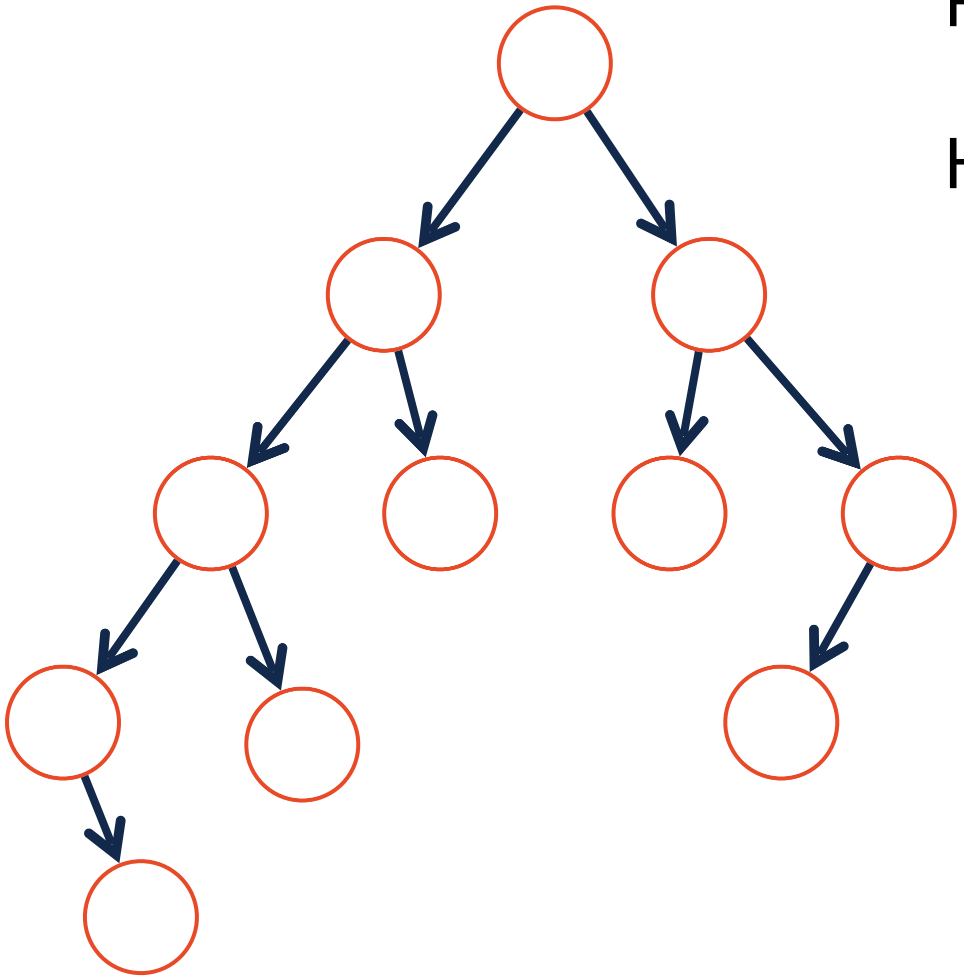


Remove decreases height by ____________.

How many rotations performed: ____________.

AVL Tree Height

If we assume a balanced tree is $O(\log n)$, does remove break this?



Remove decreases height by ___________.

How many rotations performed: ___________.

Summary of Balanced BST

Max Height: 1.44 * log(n). **[O(log n)]**

Rotations:

Zero rotations on **find**

One rotation on **insert**

O(h) == O(log(n)) rotations on **remove**

Summary of Trees

The shape of a **binary trees** can be directly meaningful

An unbalanced **binary search tree** can still be useful in the real world

An balanced **binary search tree** is guaranteed to take O(log n)

Whats next?

A non-linear data structure defined recursively as a collection of nodes where each node contains a value and zero or more connected nodes.

(In CS 277) a tree is also:

1) Acyclic — contains no cycles

2) Rooted — root node connected to all nodes