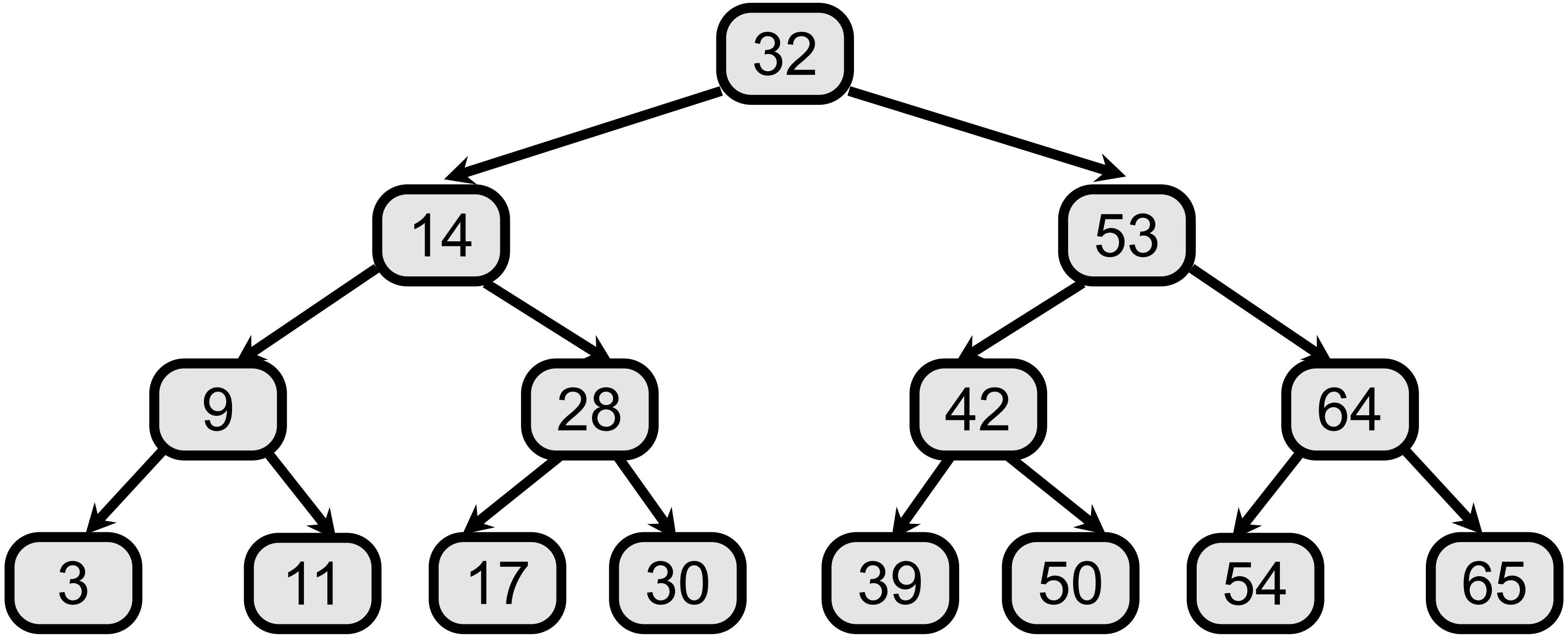
# Binary Search Trees: Exercises

Semester 2, 2020

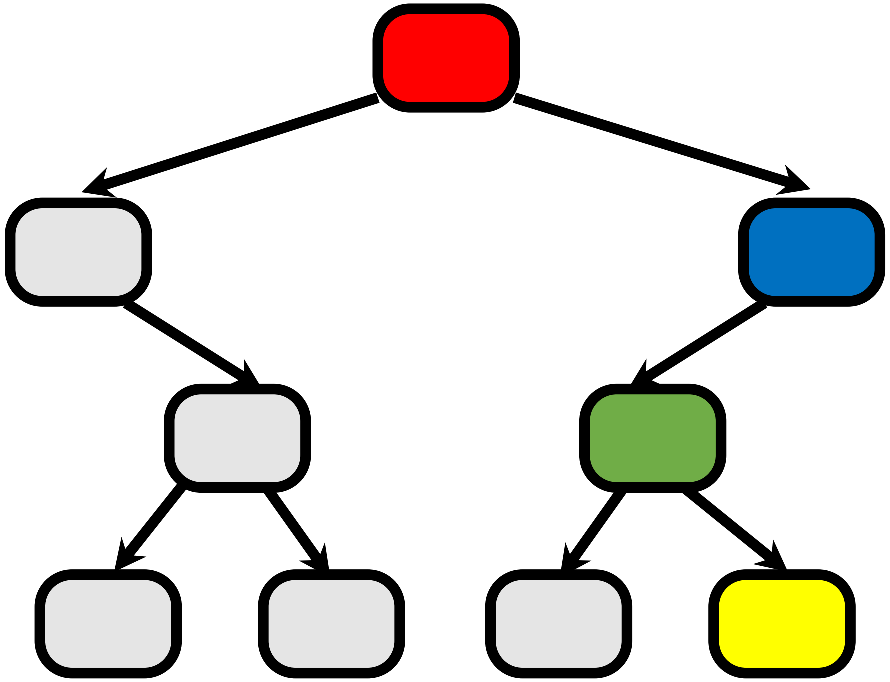
Kris Ehinger

# Review: Binary search tree



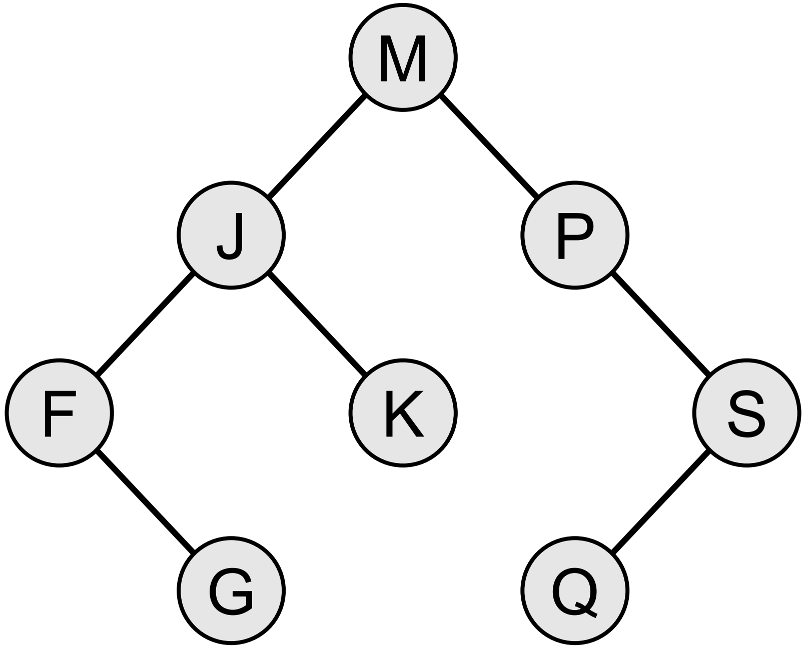
# Review: Tree traversal

- Which is the highest key (last item from in-order traversal) in this tree?



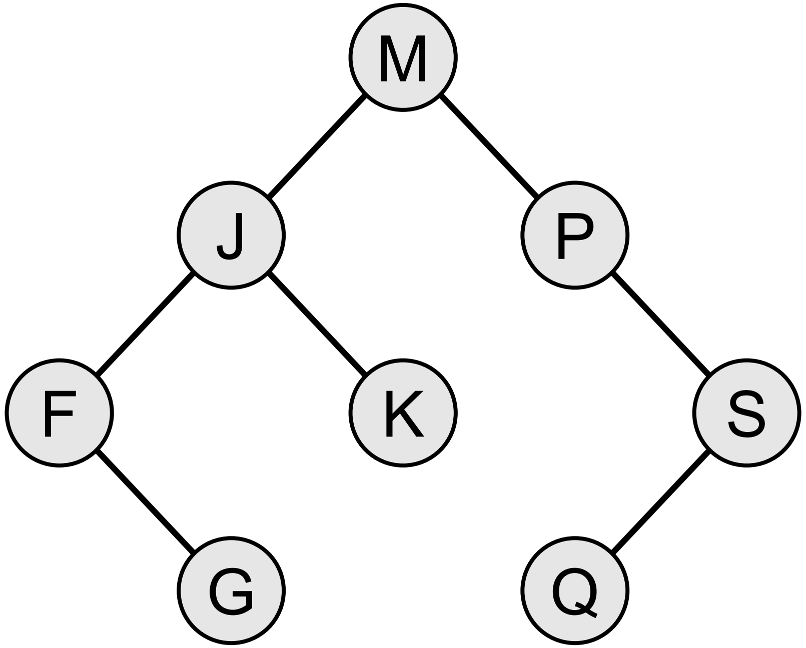
# Tree traversal

- What is the output of recursive post-order tree traversal? Assume that visit(t) prints the node’s key.

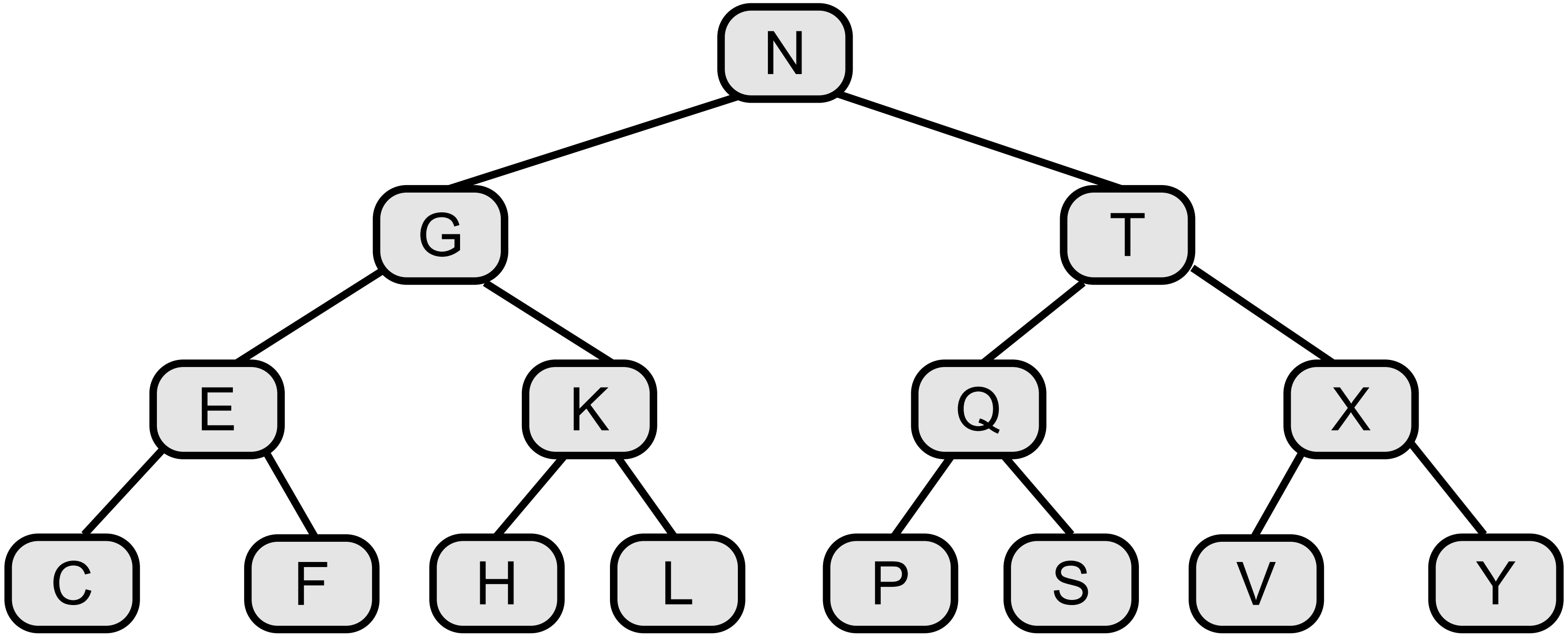


# Tree traversal

- What is the output of recursive pre-order tree traversal? Assume that visit(t) prints the node's key.

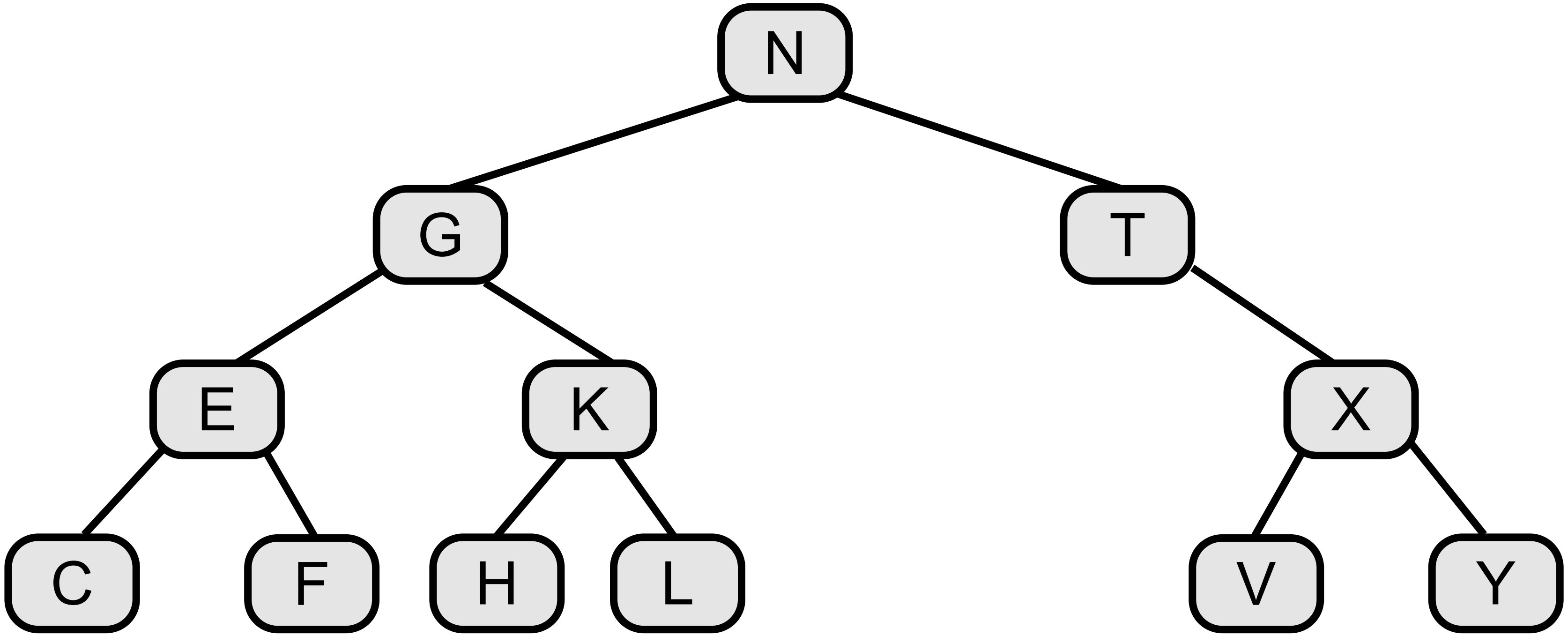


# In-order predecessor / successor



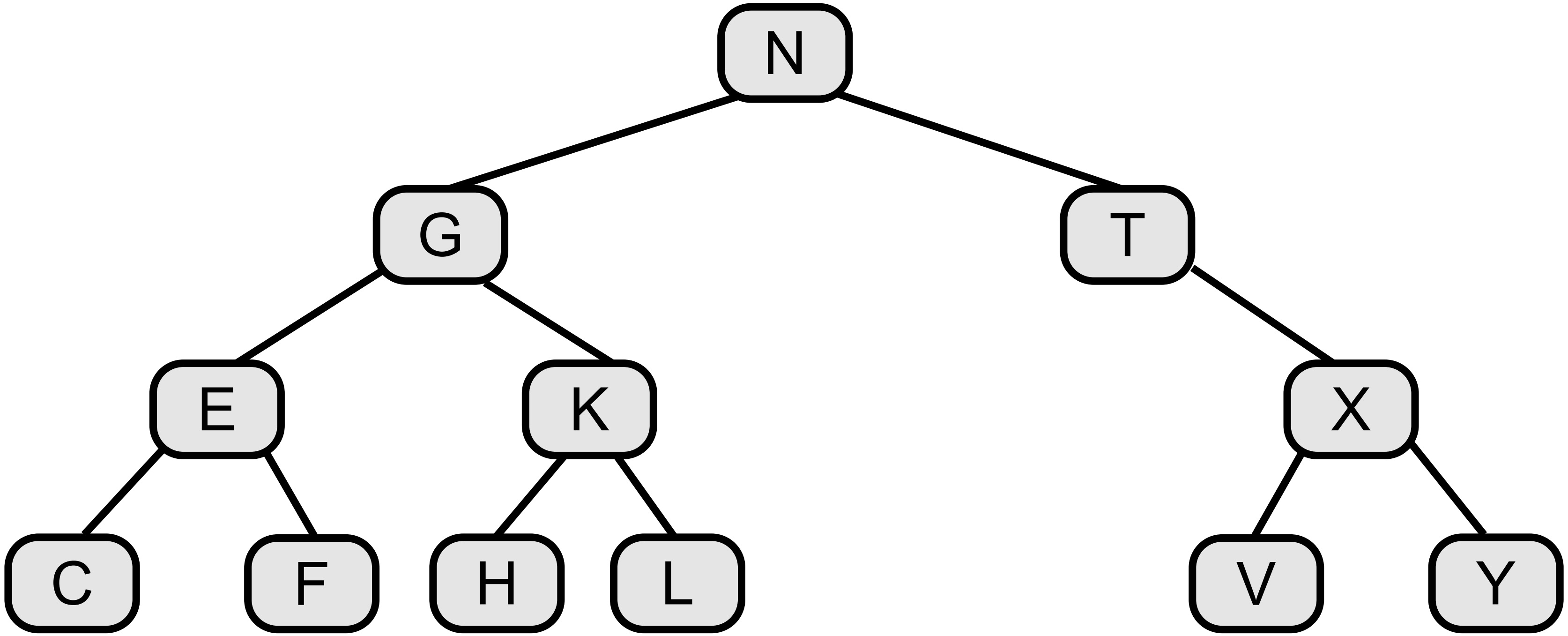
# Node deletion

- How to delete?



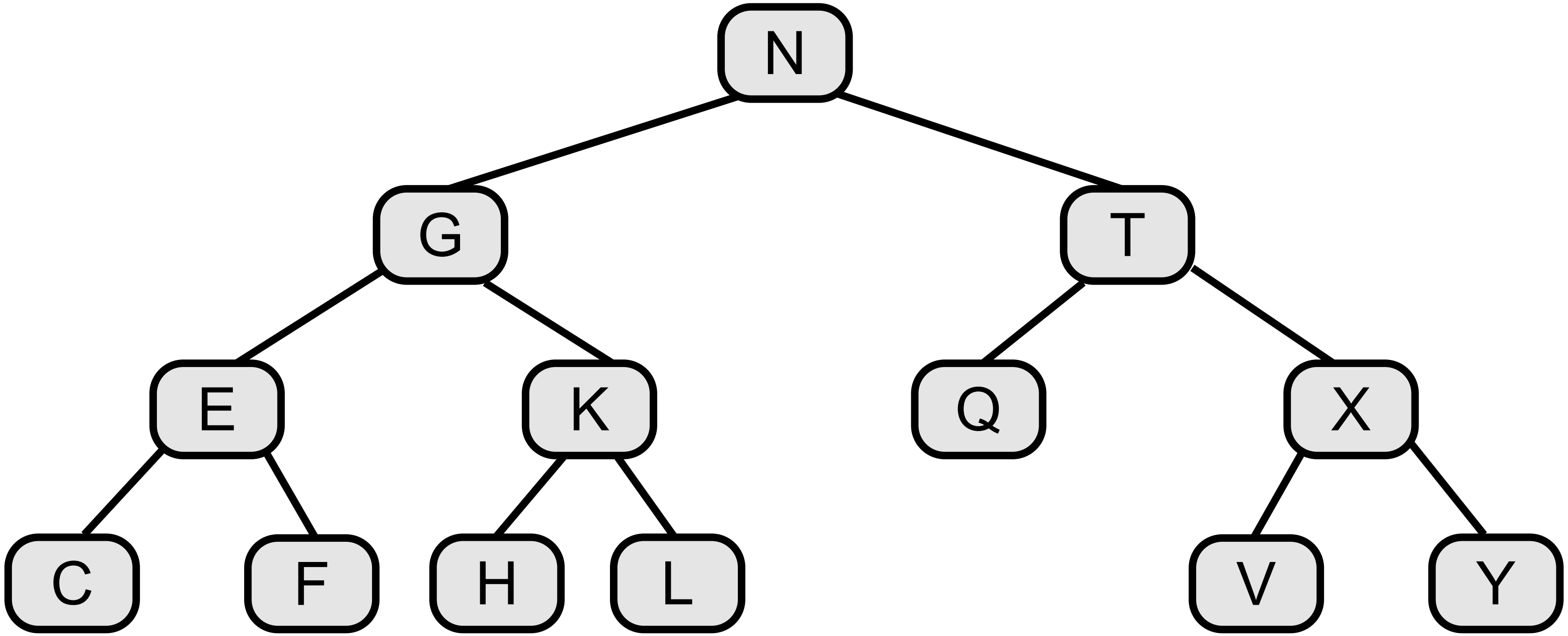
# Node deletion

- How to delete?



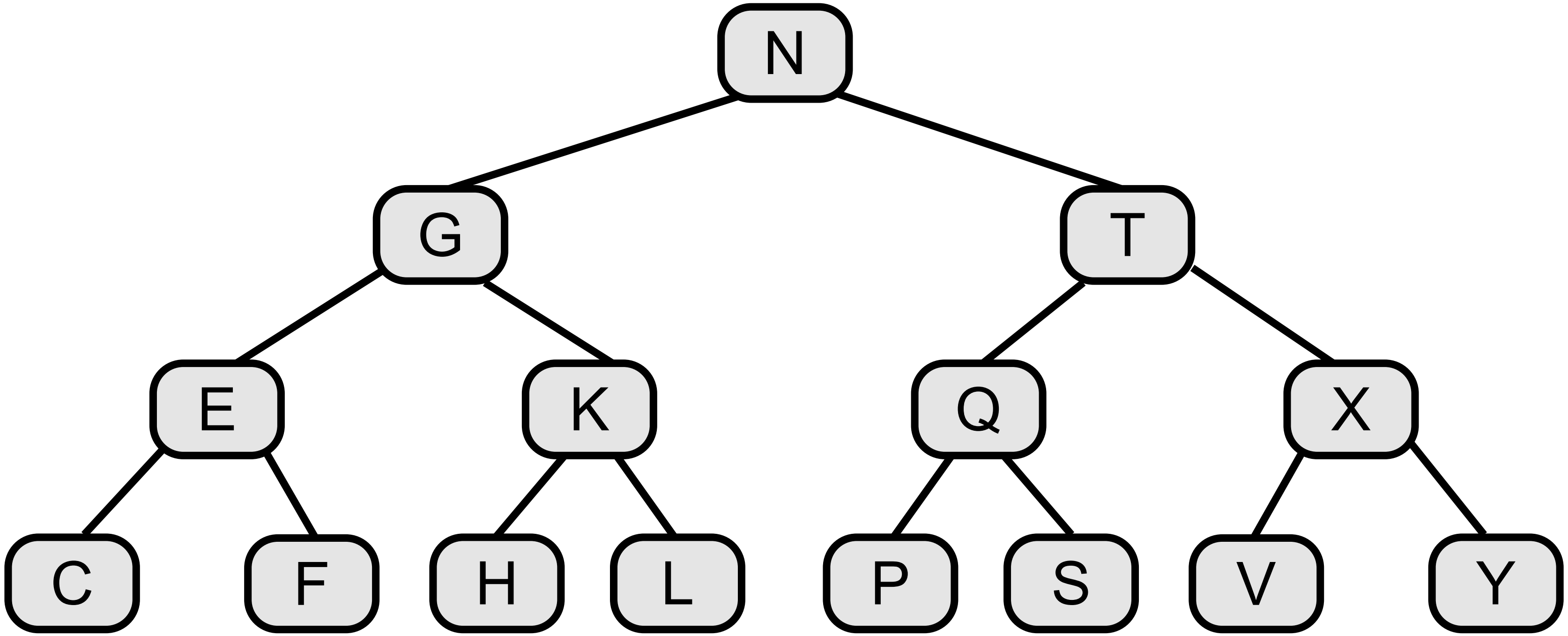
# Node deletion

- How to delete?



# Node deletion

- How to delete?



# Review: Binary search trees

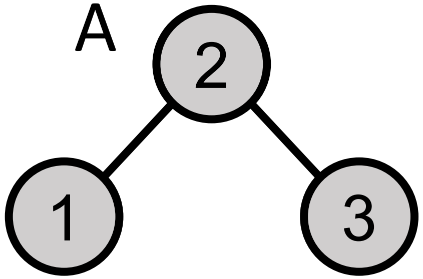
What is the difference between these trees?

A. Items were added in a different order

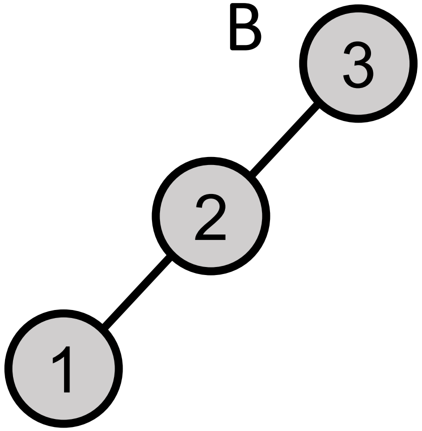
B. A and C are AVL balanced, but B is not

C. A and B are valid binary search trees, but C is not

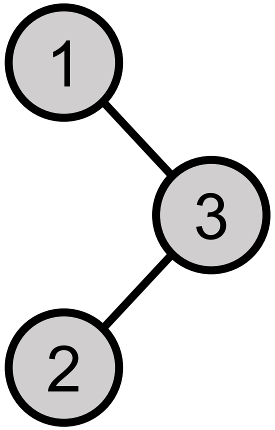
A



B

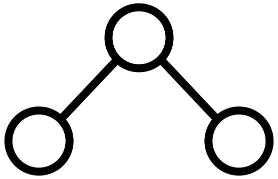


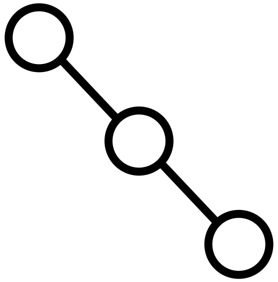
C

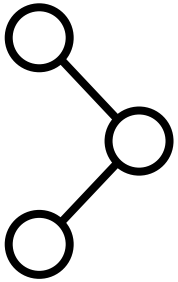


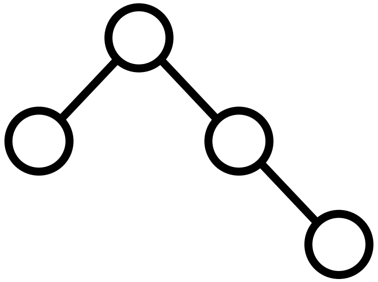
# Are these trees AVL balanced?

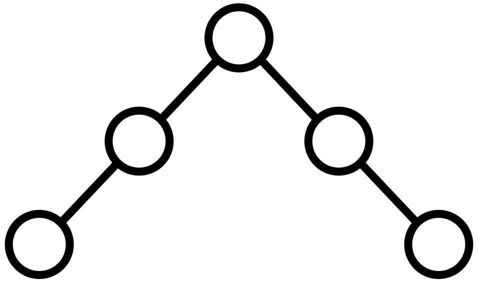
**Depth(left) – Depth(right)**

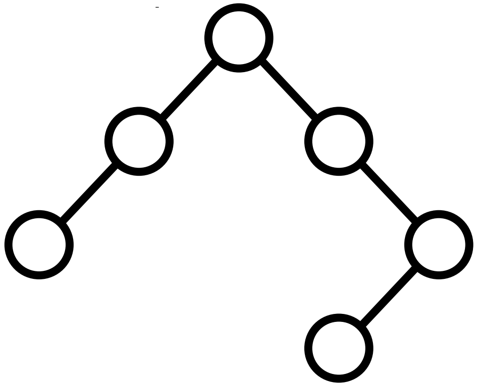




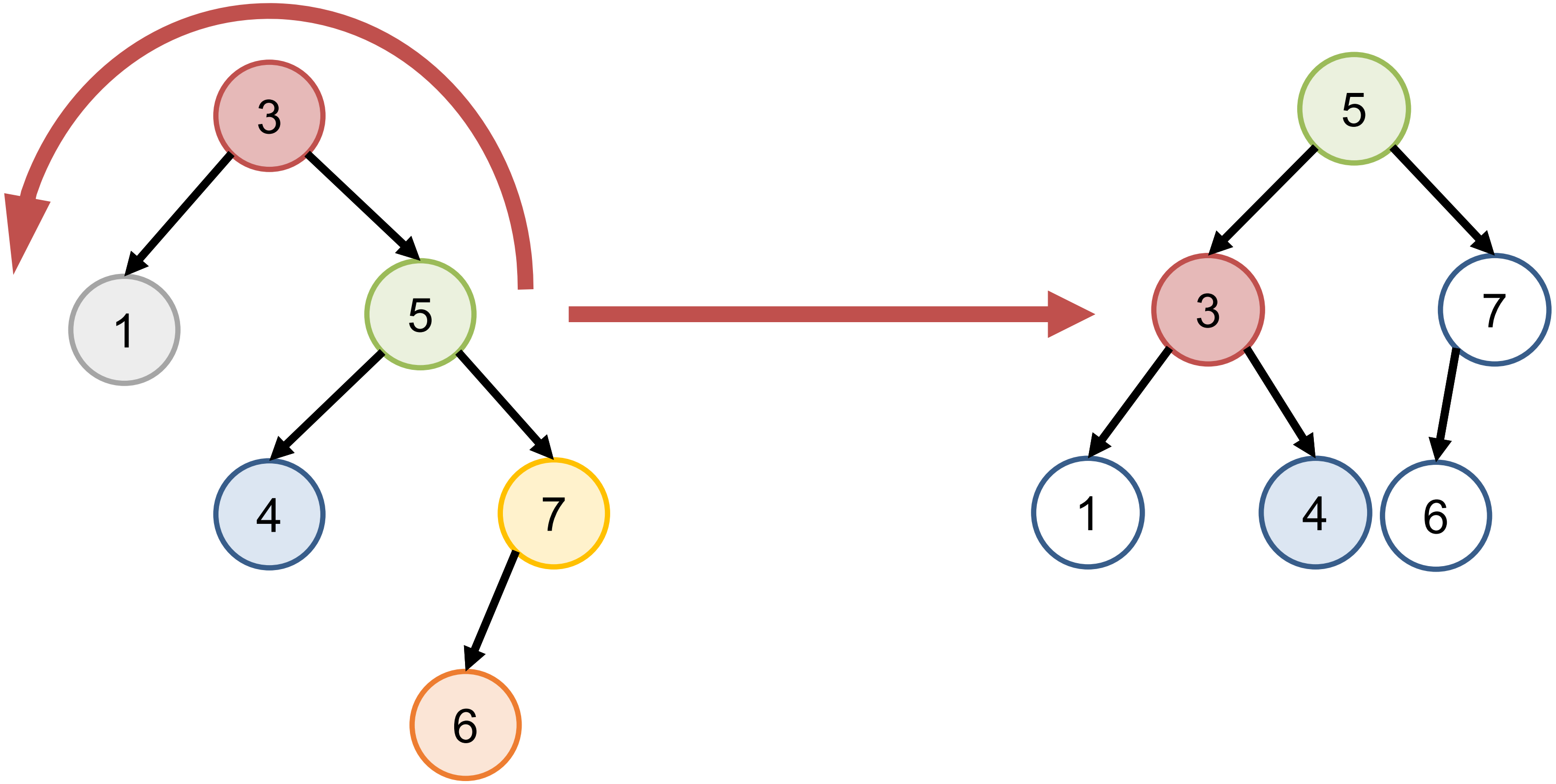






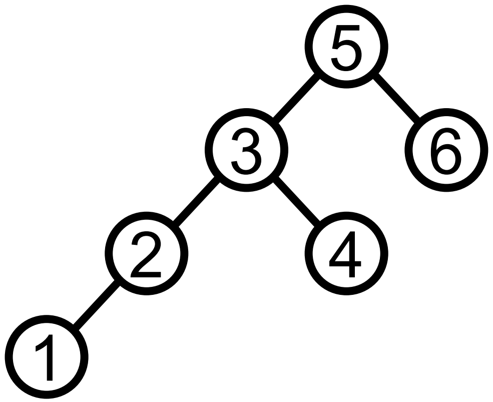


# AVL rotation

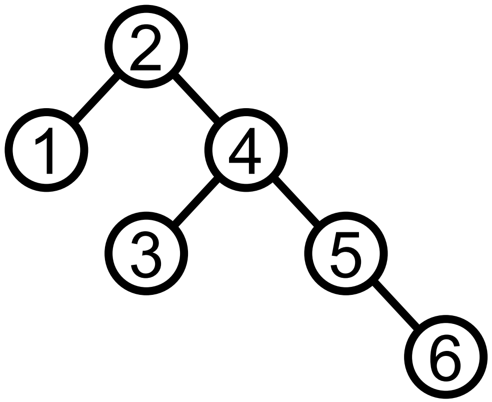


# Review: Rotation

Tree 1



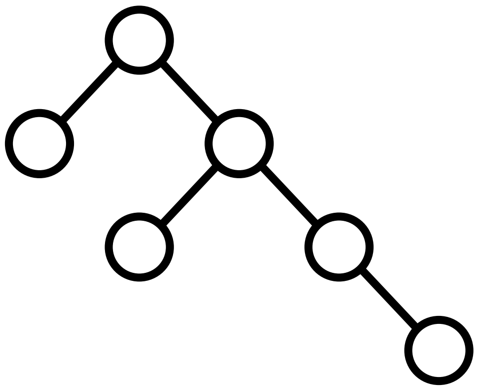
Tree 2



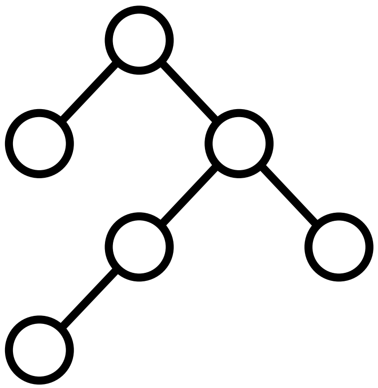
A. Tree1: rotate right and Tree2: rotate left
B. Tree1: rotate left and Tree2: rotate right
C. They are both already balanced

# Review: AVL imbalance

Tree 1

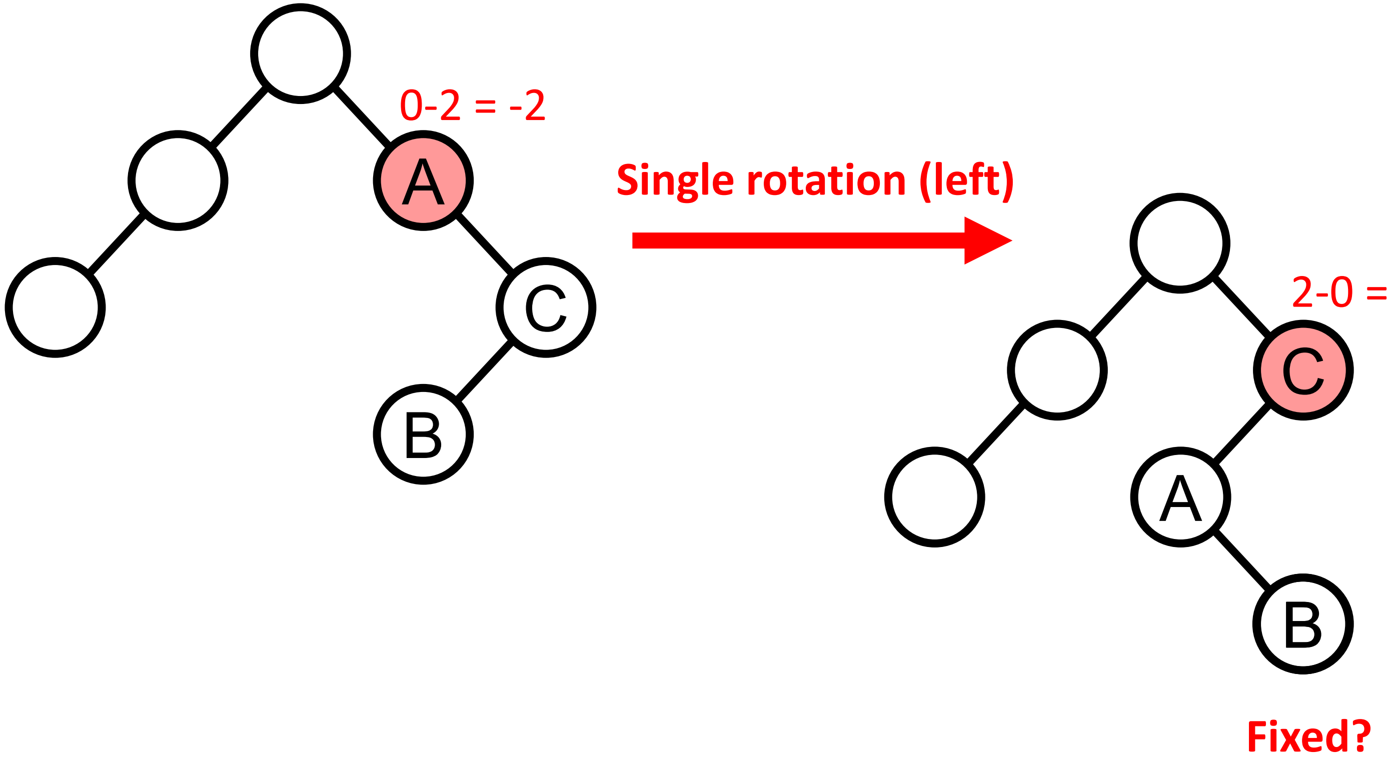


Tree 2



A. Tree1: outside imbalance, Tree2: inside imbalance
B. Tree1: inside imbalance, Tree2: outside imbalance
C. Both trees have an inside imbalance

# Why double rotation?



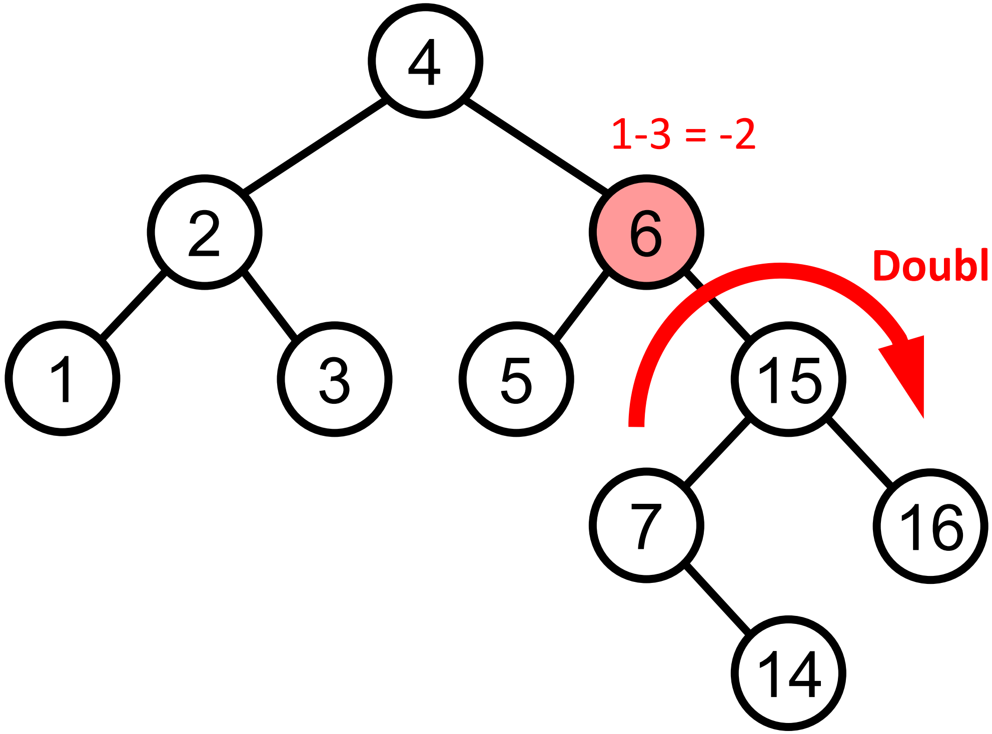
0-2 = -2

Single rotation (left)

2-0 = 2

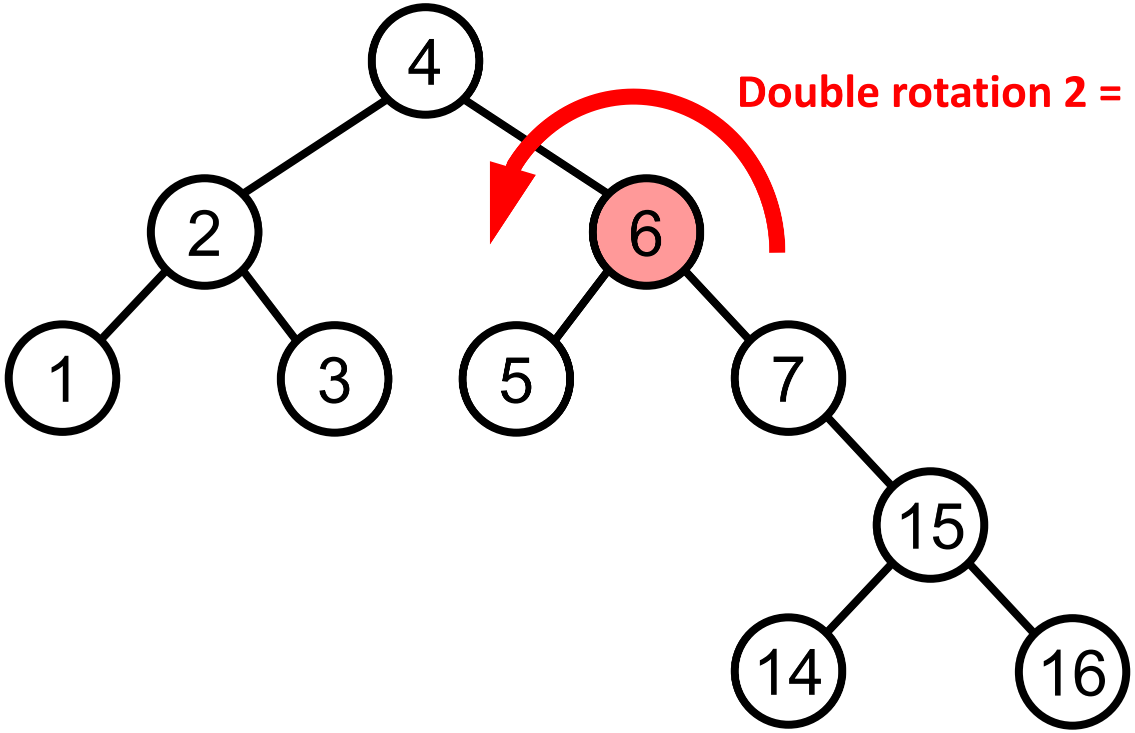
Fixed?

# Exercise: AVL trees



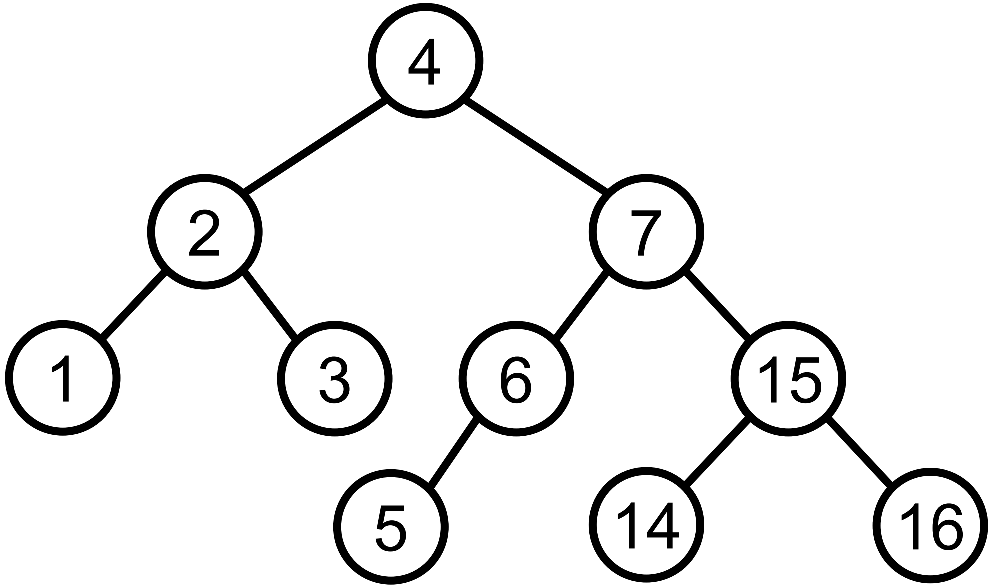
Double rotation 1 = Right rotation

# Exercise: AVL trees



Double rotation 2 = Left rotation

# Exercise: AVL trees


4
2
7
1
3
6
15
5
14
16