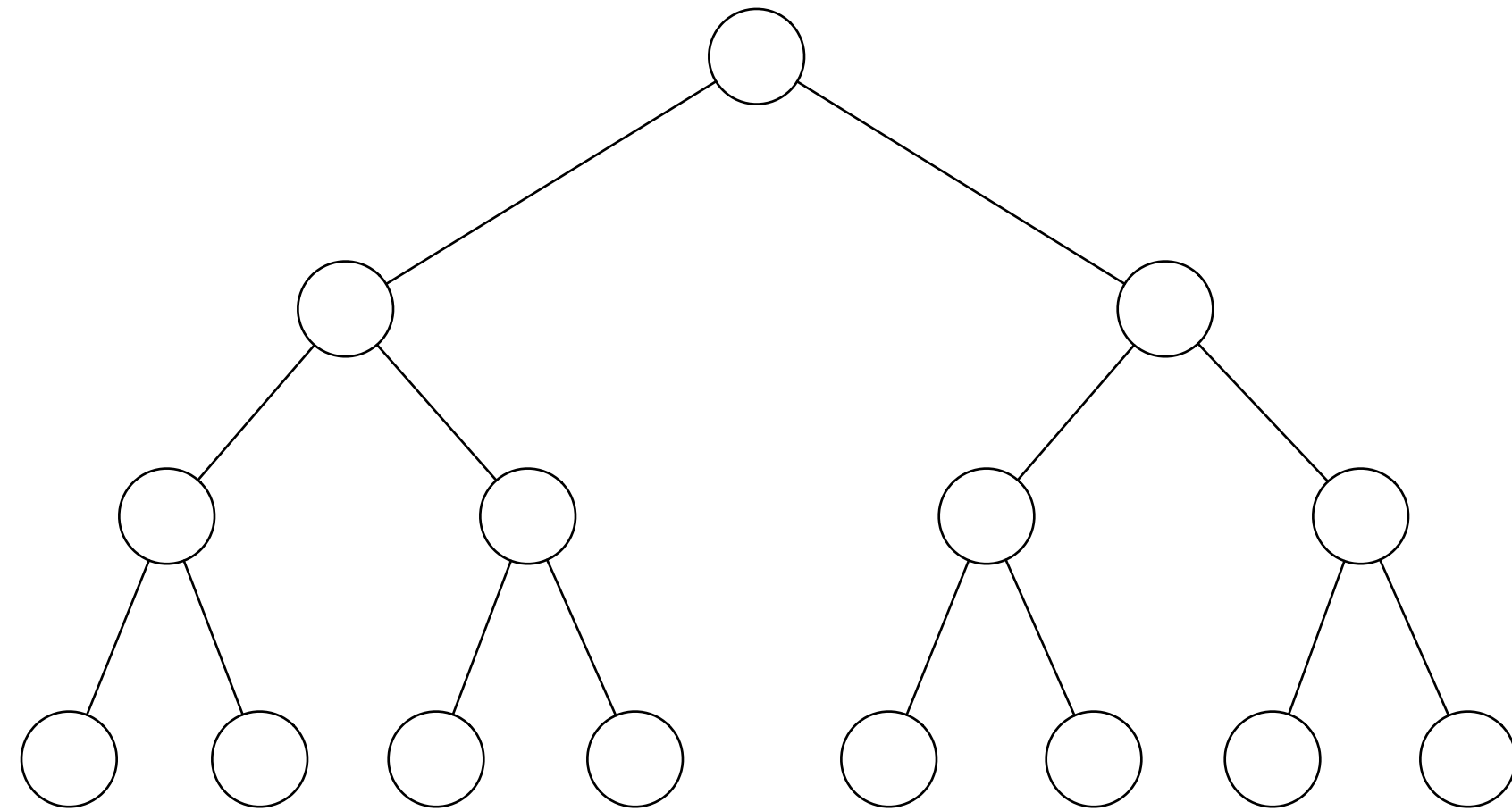




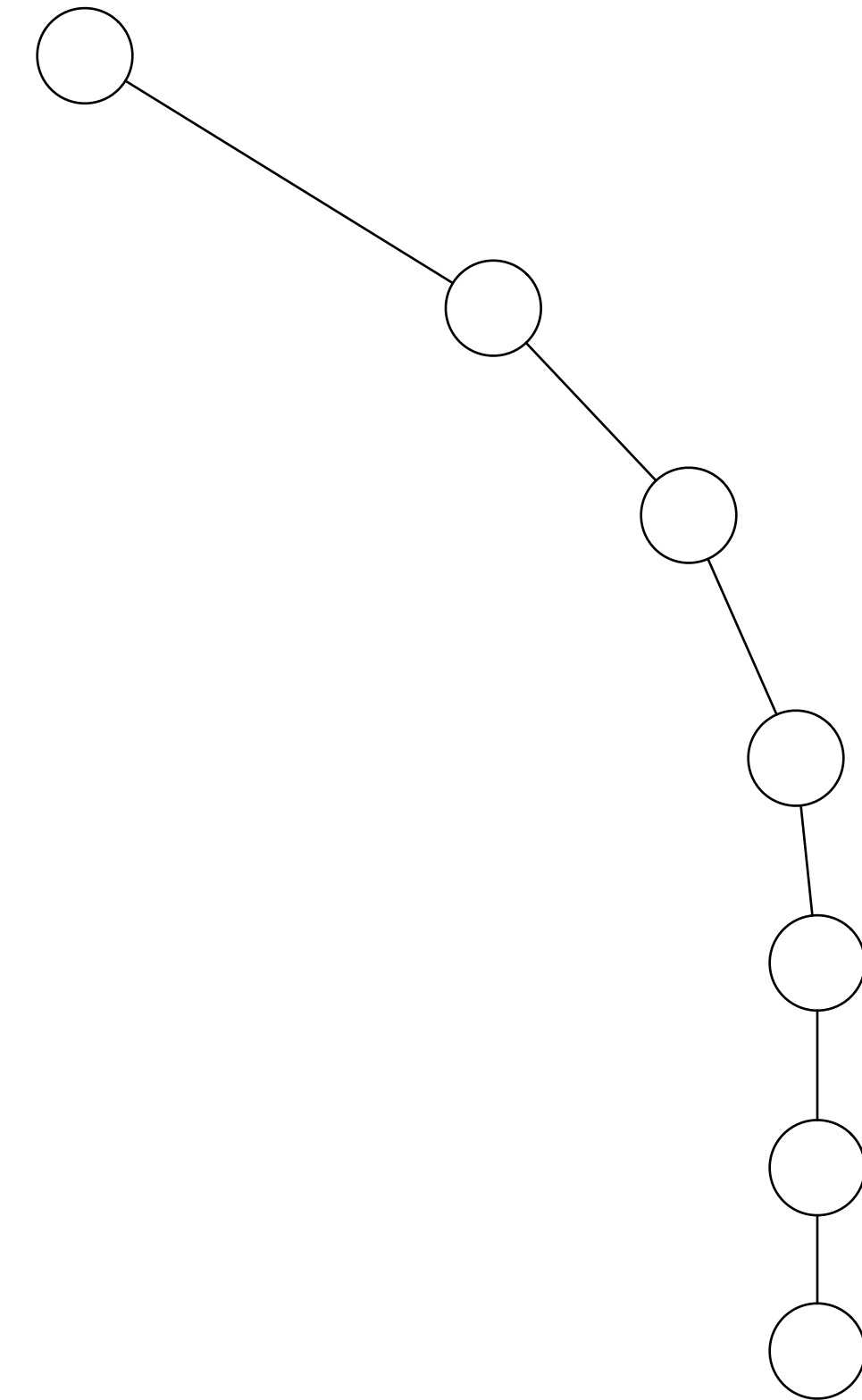
THE UNIVERSITY OF VERMONT  
COLLEGE OF ENGINEERING &  
MATHEMATICAL SCIENCES

# AVL Tree

# Complexity of search



Complete or perfect tree?  
 $O(\log N)$



Pathological tree?  
 $O(h) = O(N - 1) = O(N)$

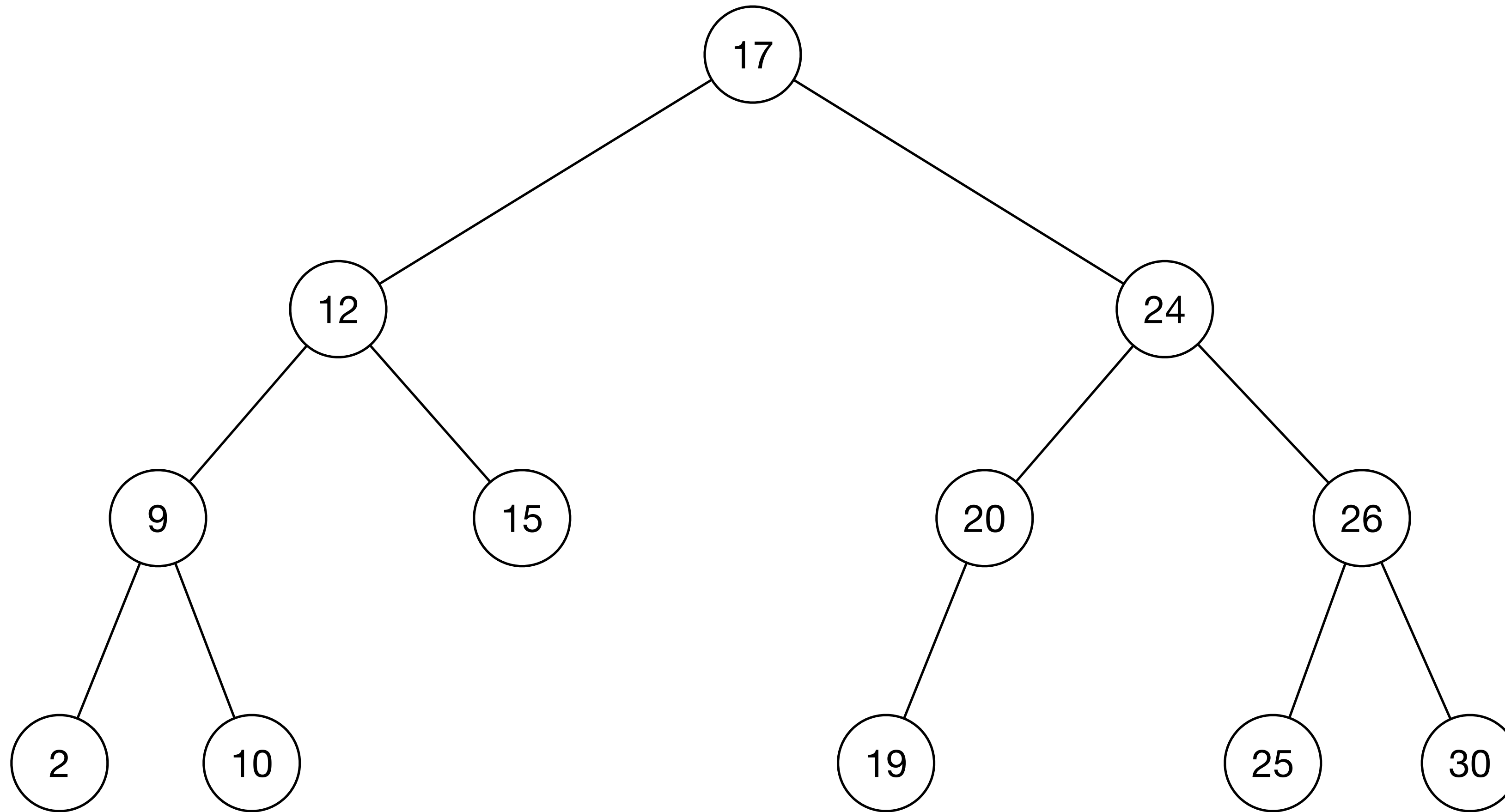
# AVL Tree

- An AVL tree is a binary search tree with one additional property:

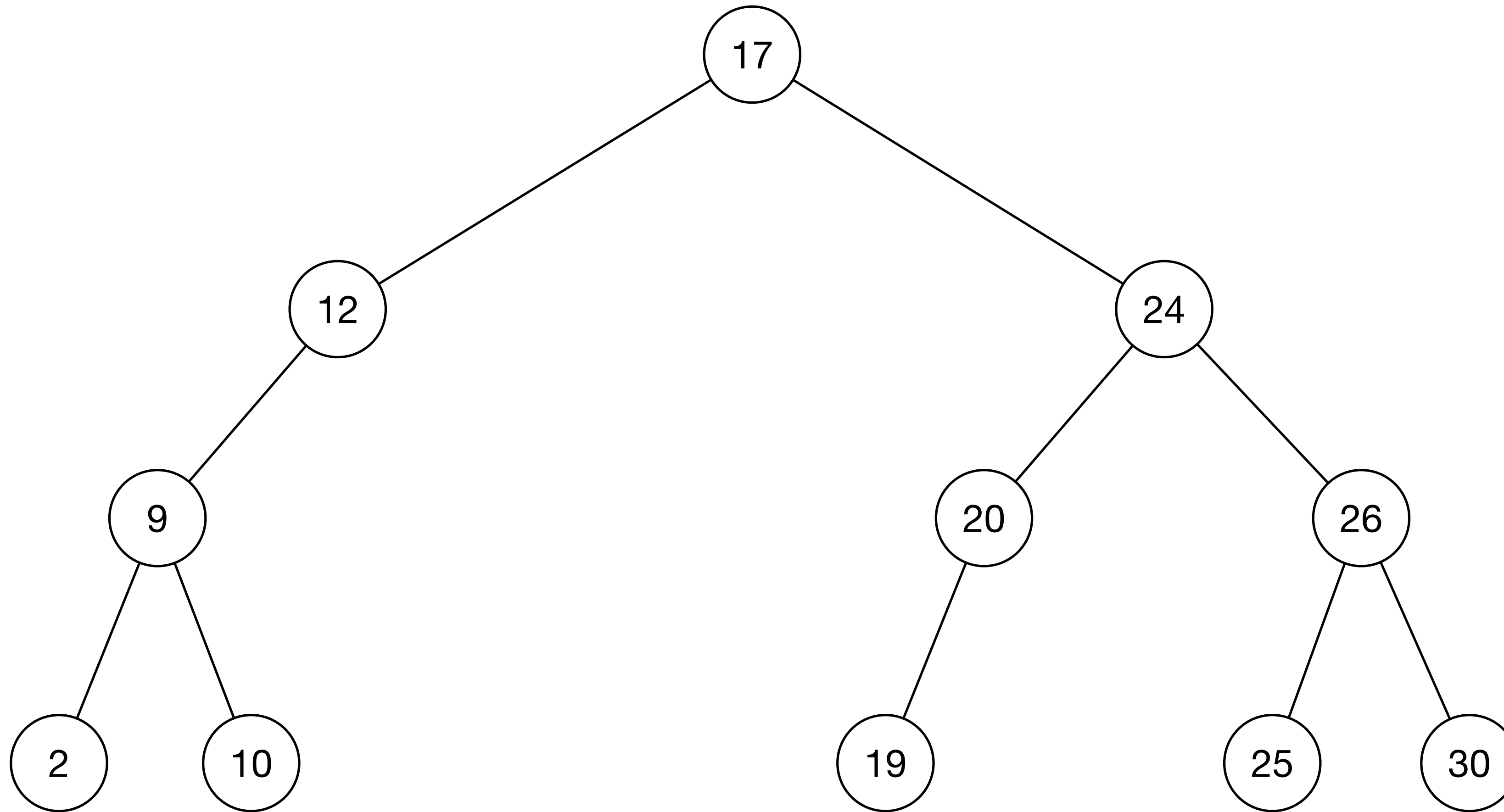
For each node in the tree, the height of left and right subtrees can differ by at most 1.

- Why "AVL"? Named after its inventors, Adelson-Velsky and Landis (1962).
- With this property, AVL trees are "self-balancing." This preserves the  $O(\log N)$  time for search, insertion and deletion, by avoiding pathological structures and substructures.

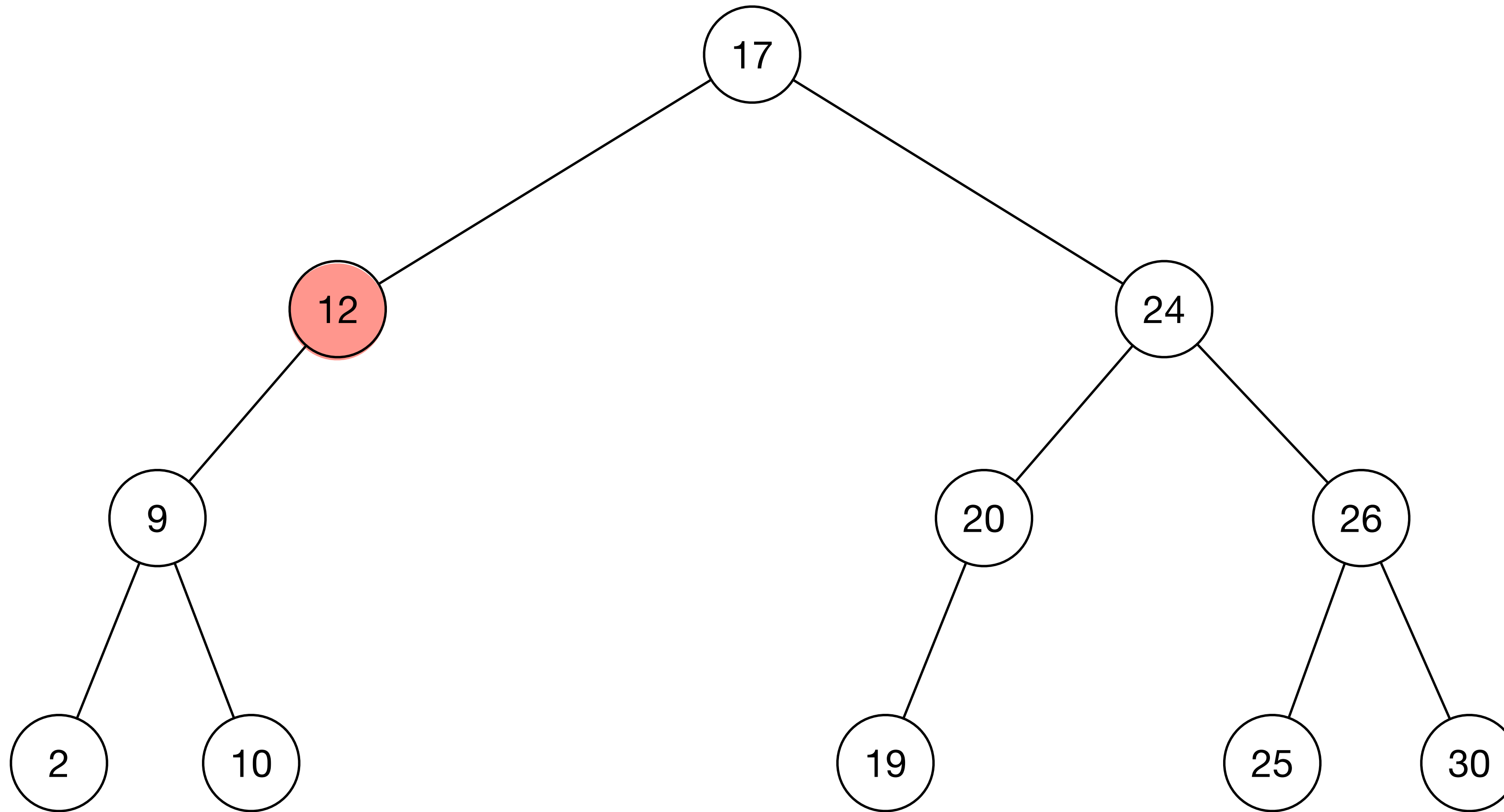
# AVL Tree Example



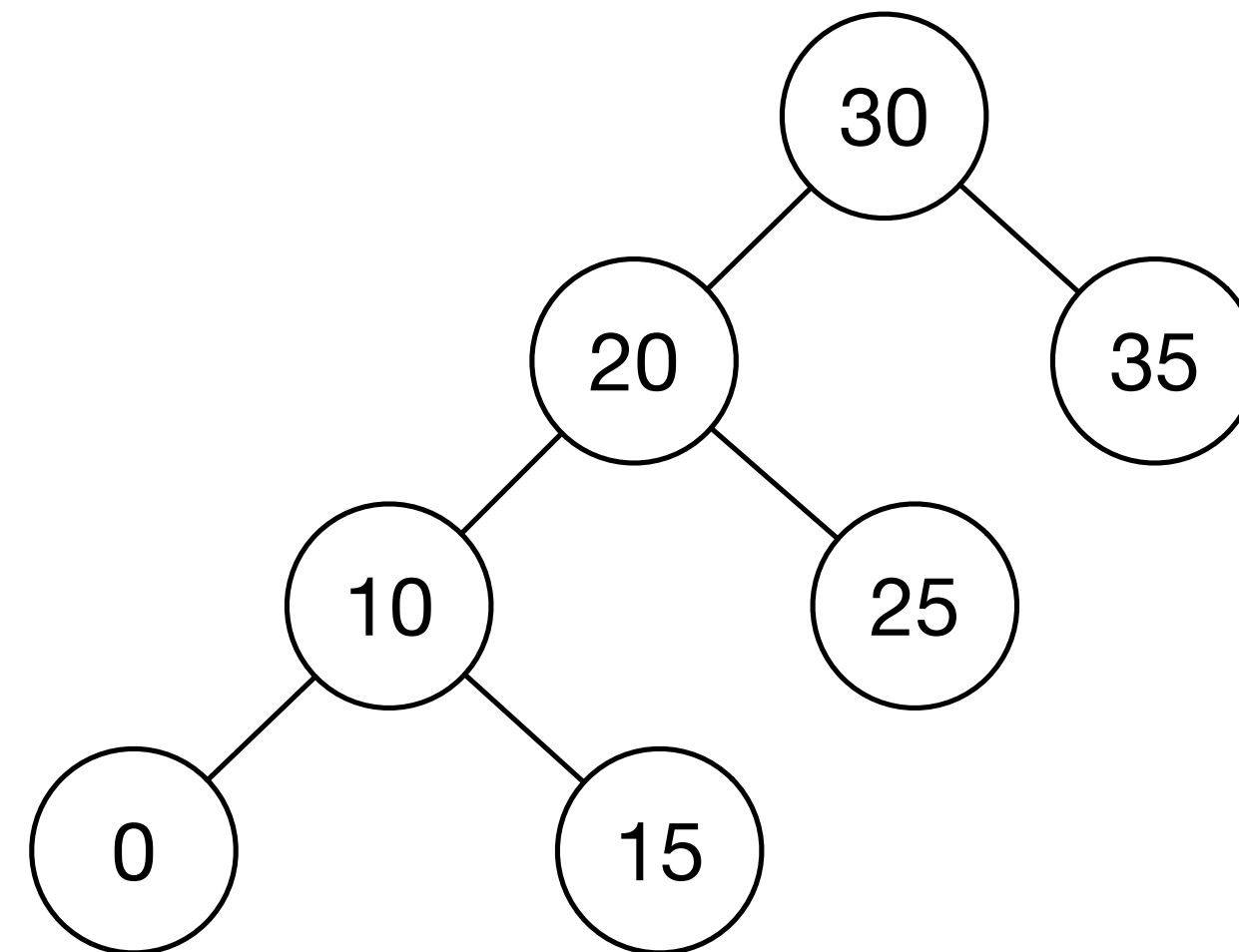
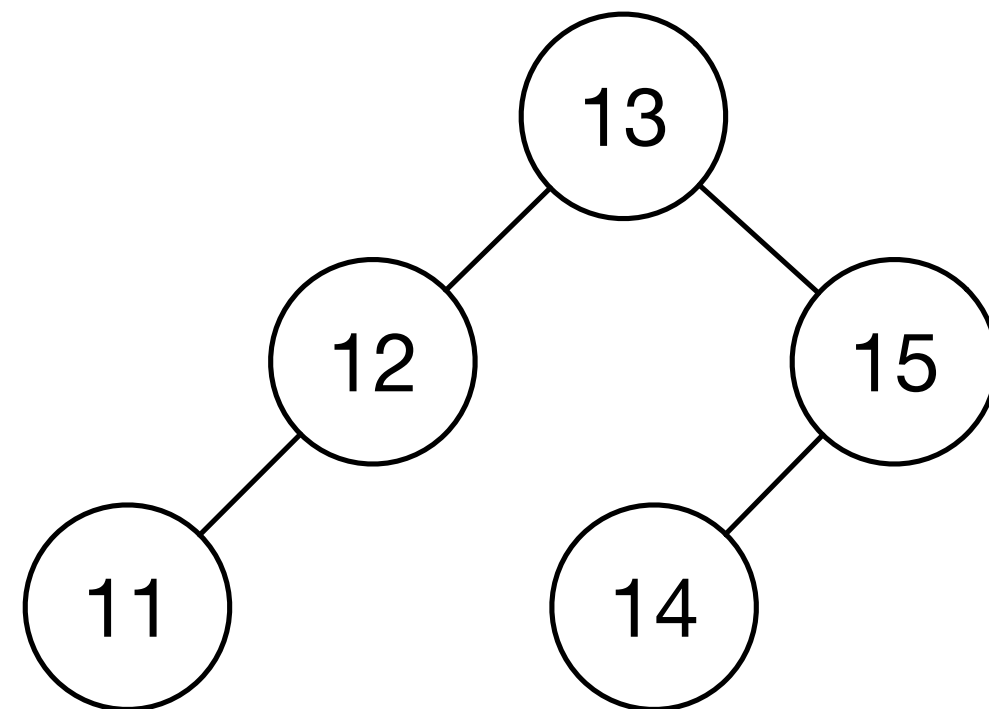
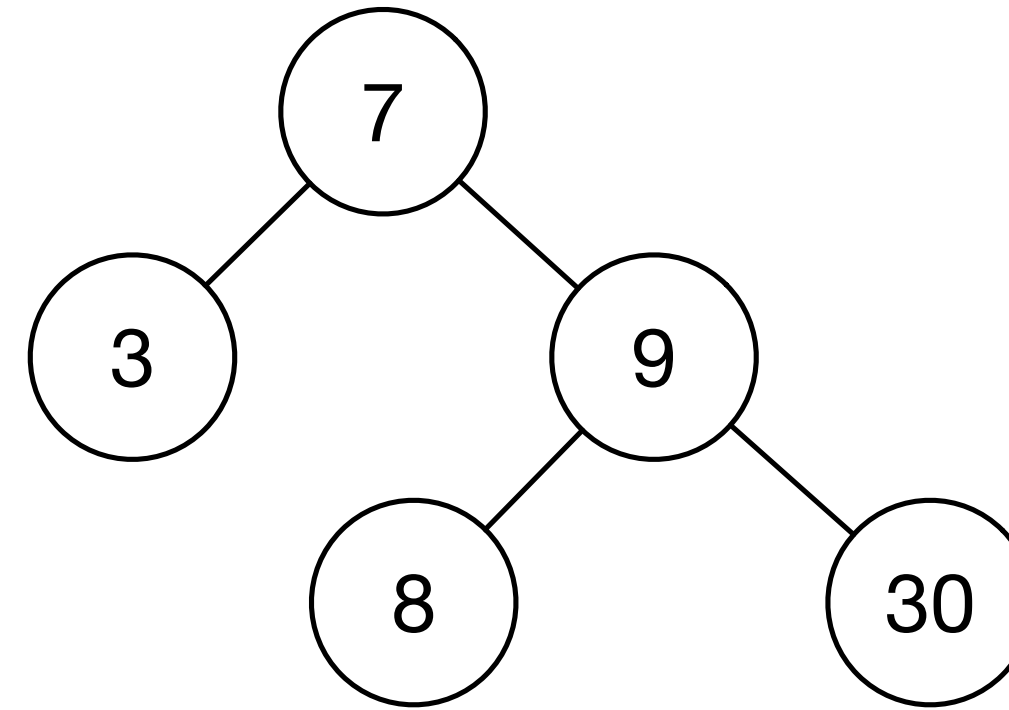
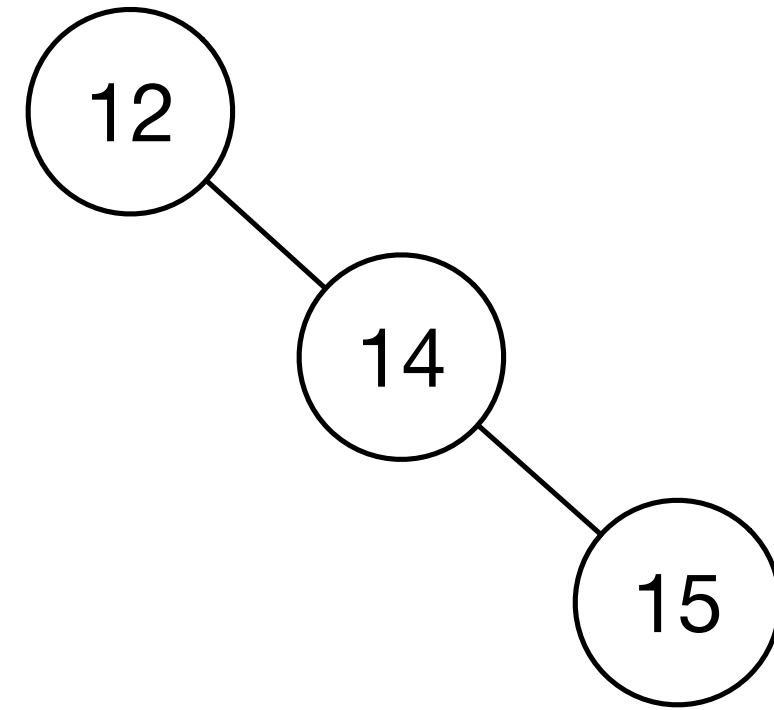
# AVL Tree Counter-example



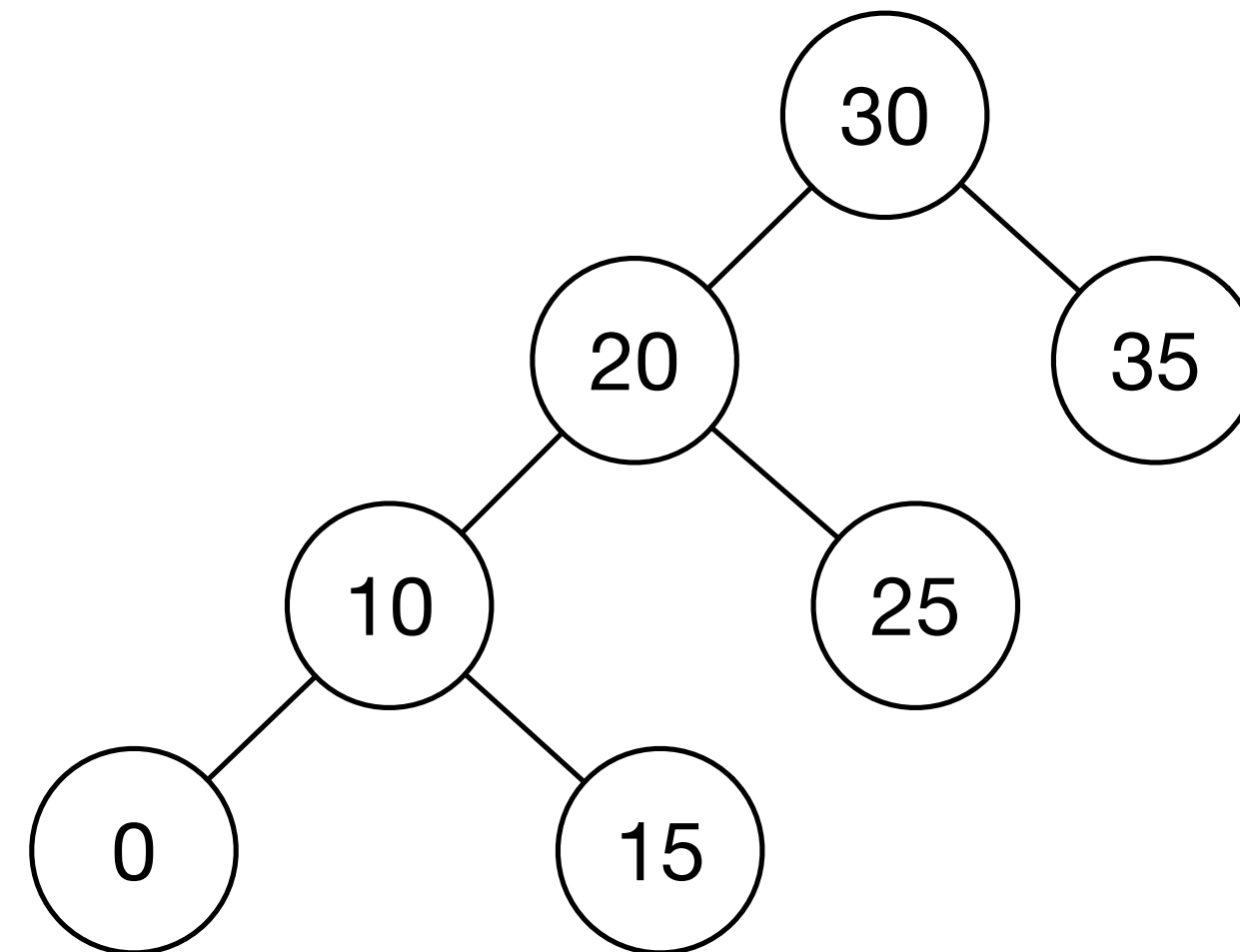
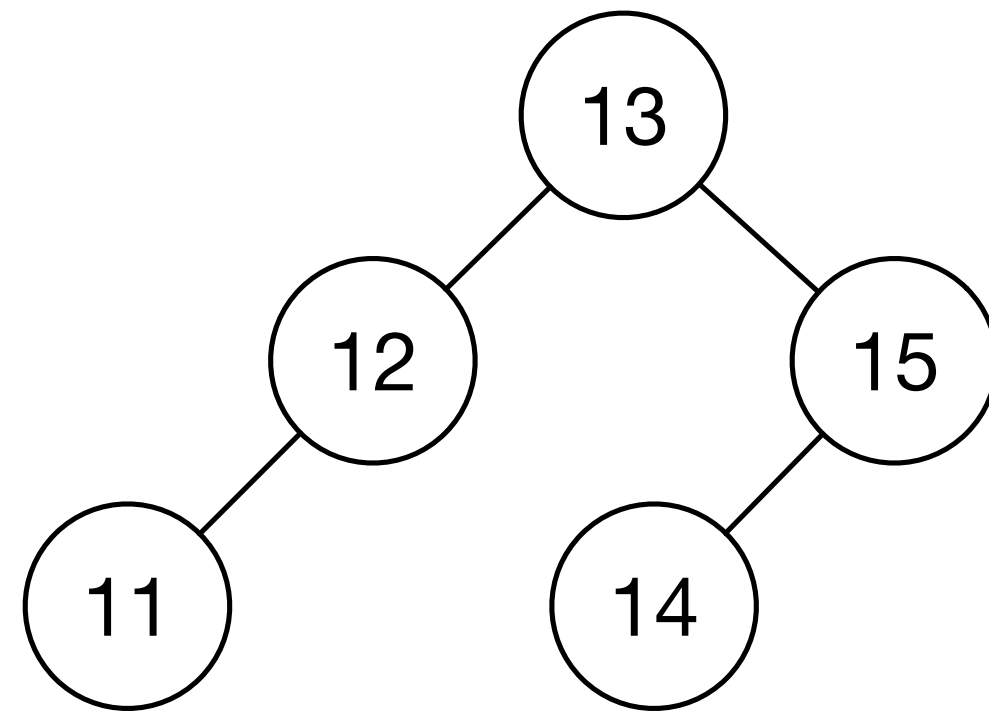
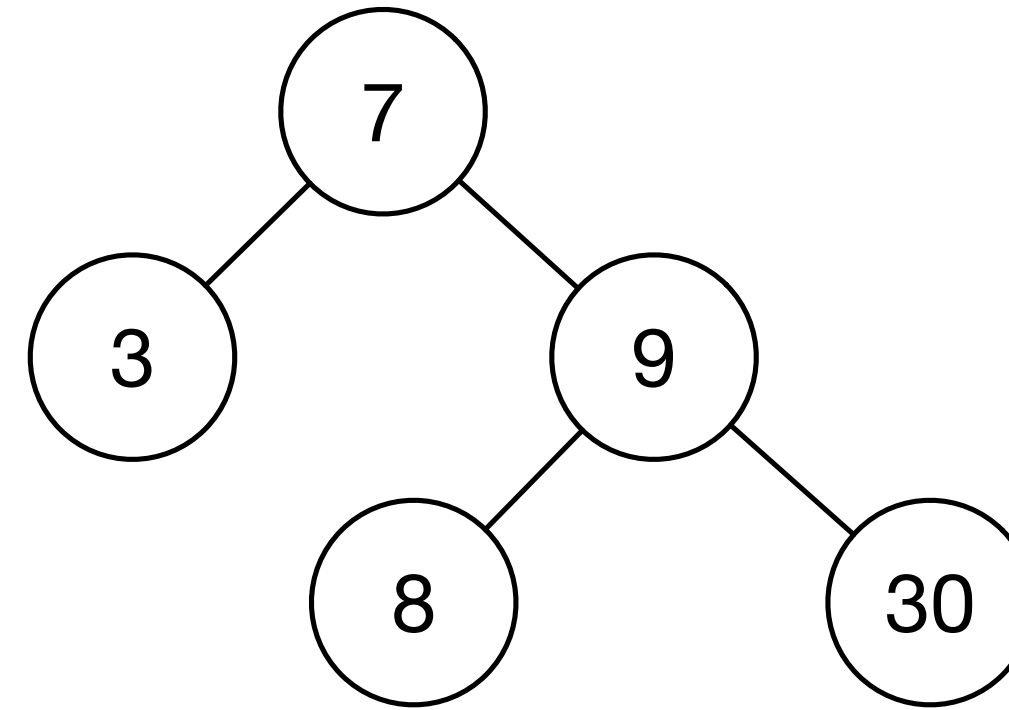
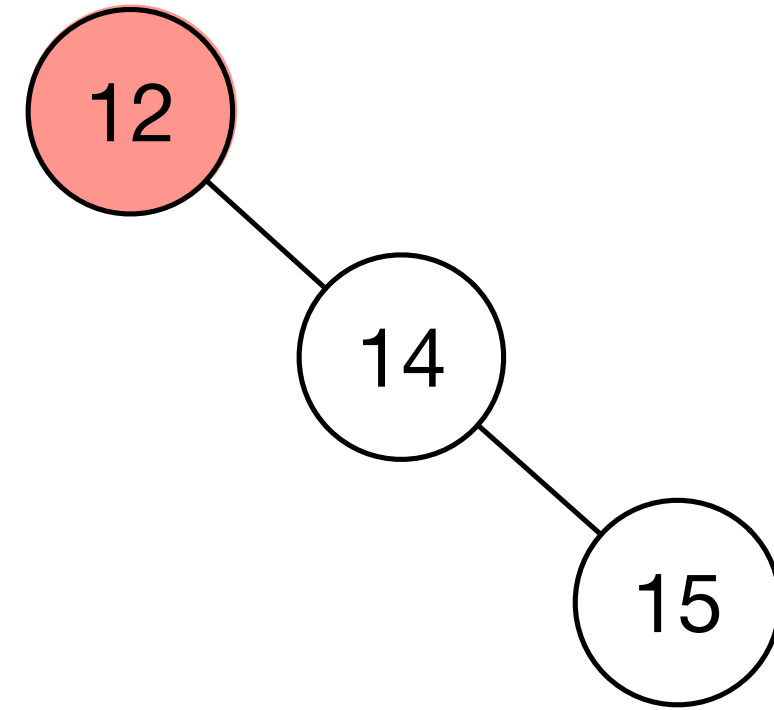
# AVL Tree Counter-example



# Which of these are AVL trees?

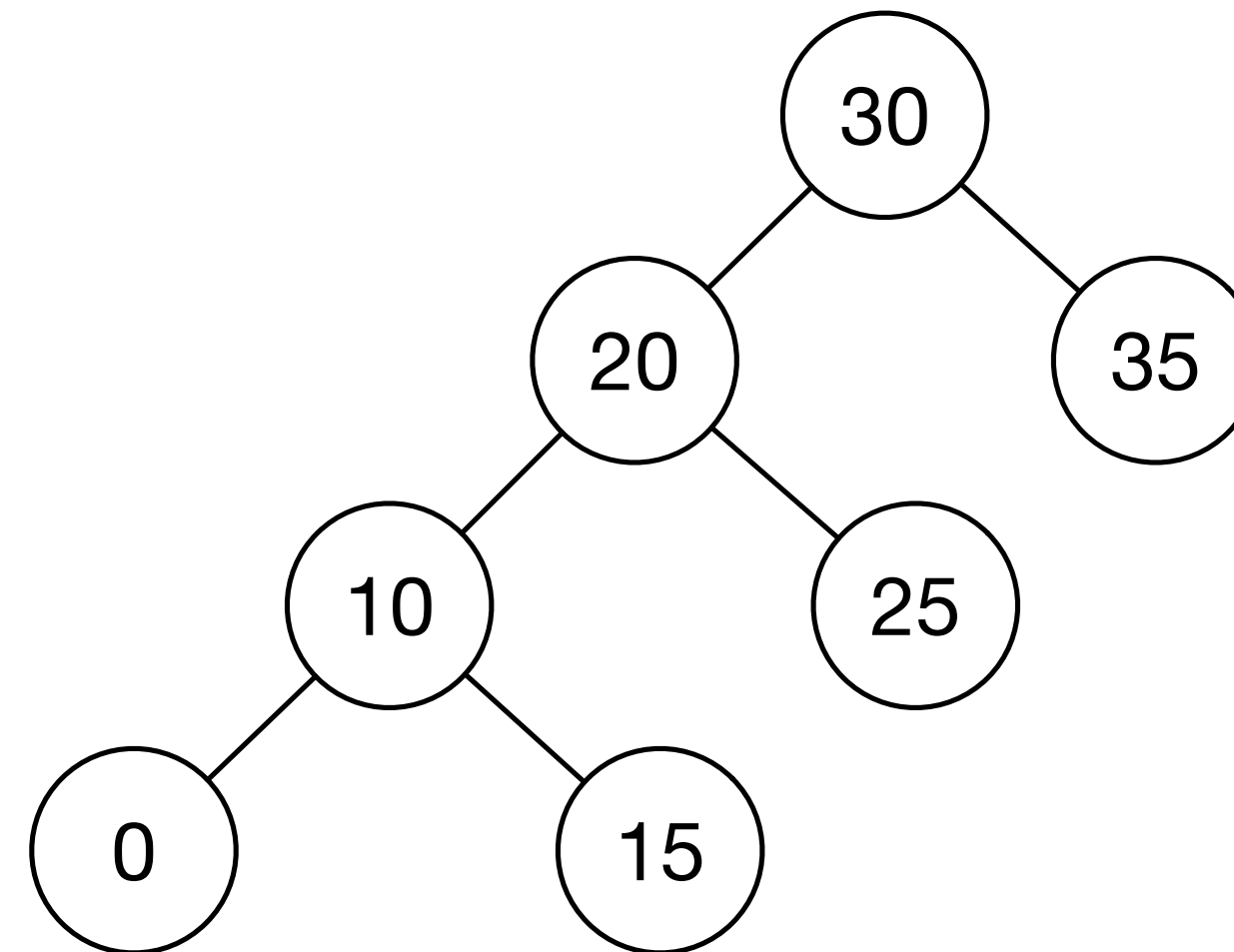
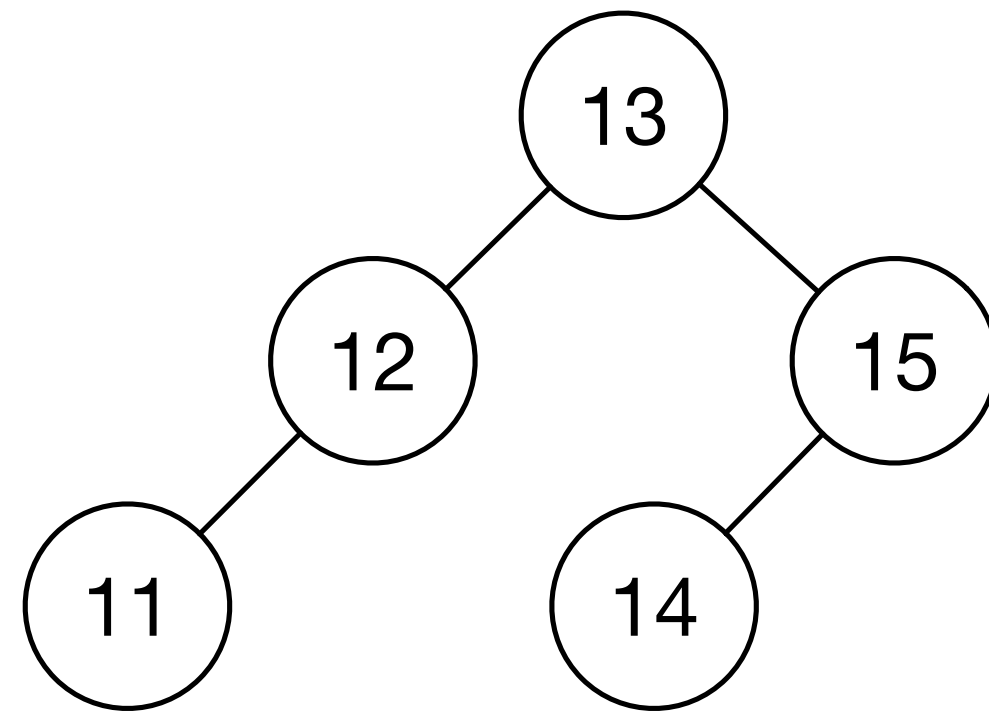
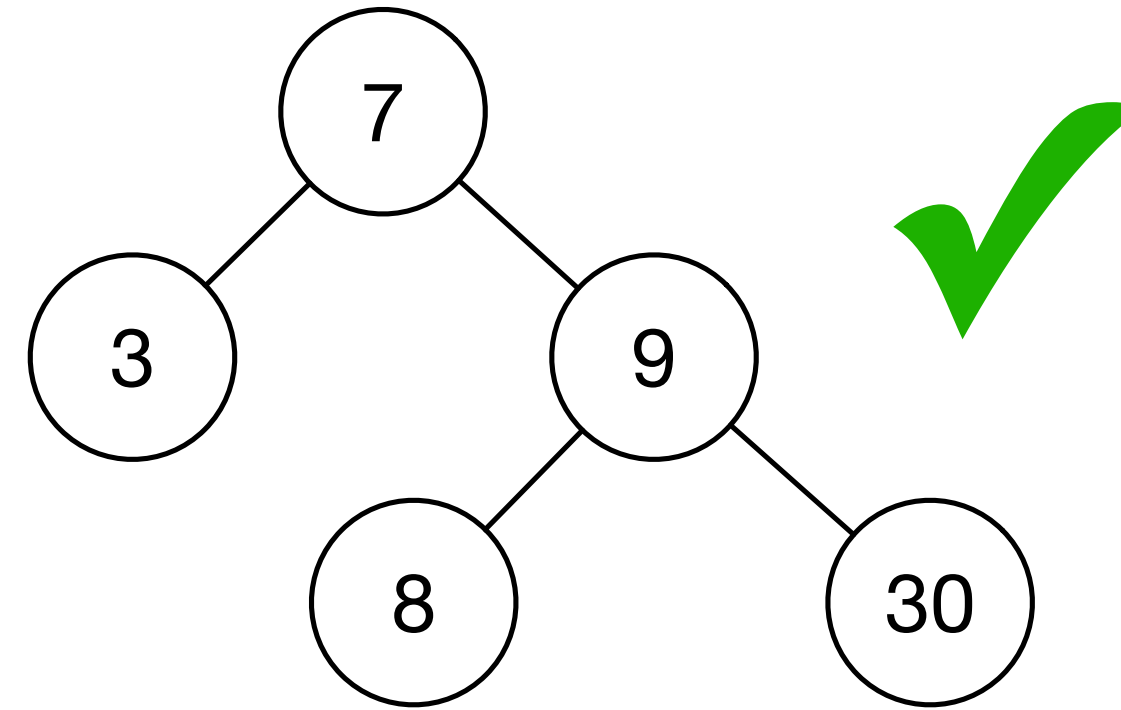
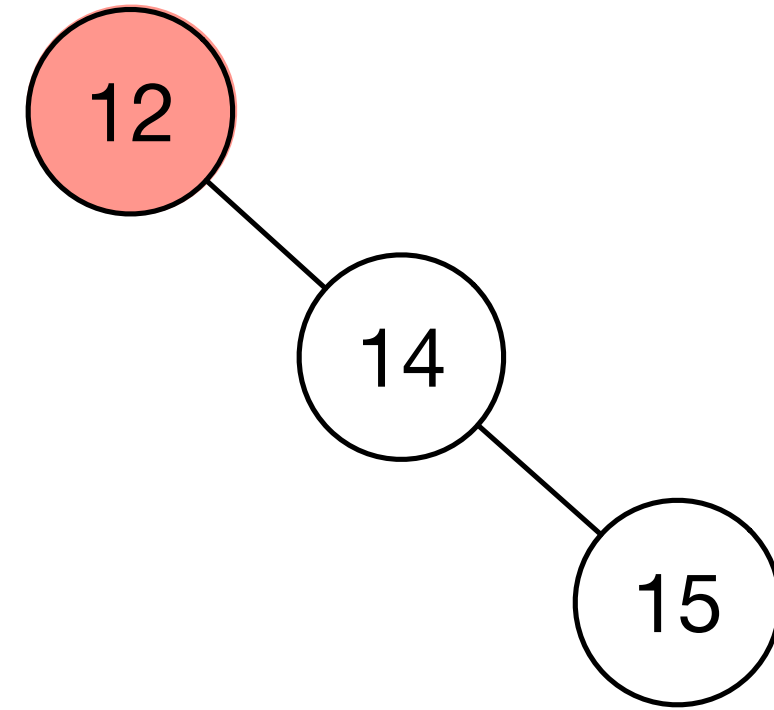


# Which of these are AVL trees?

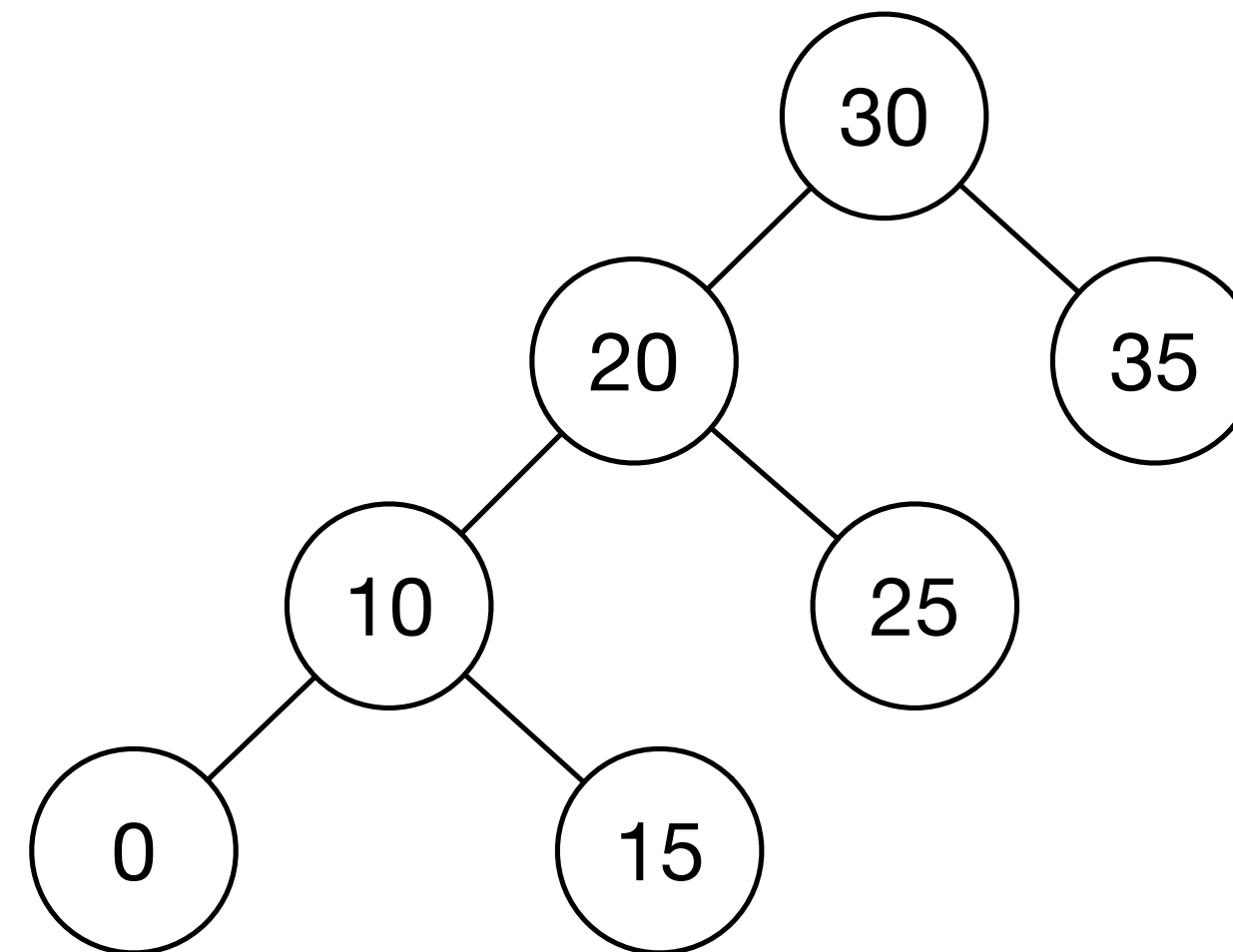
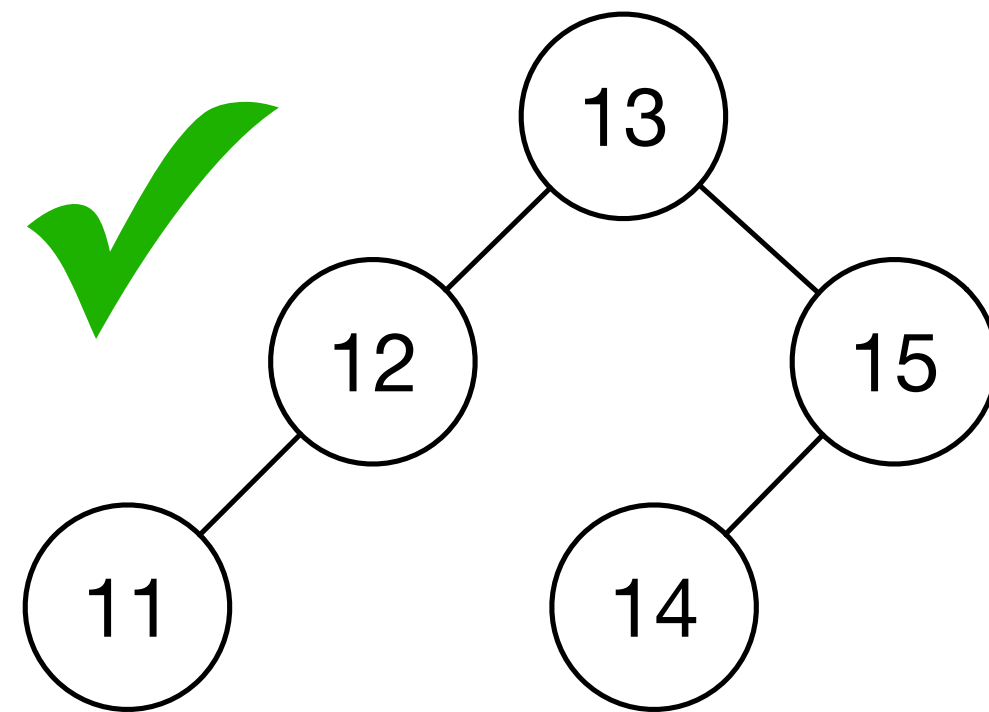
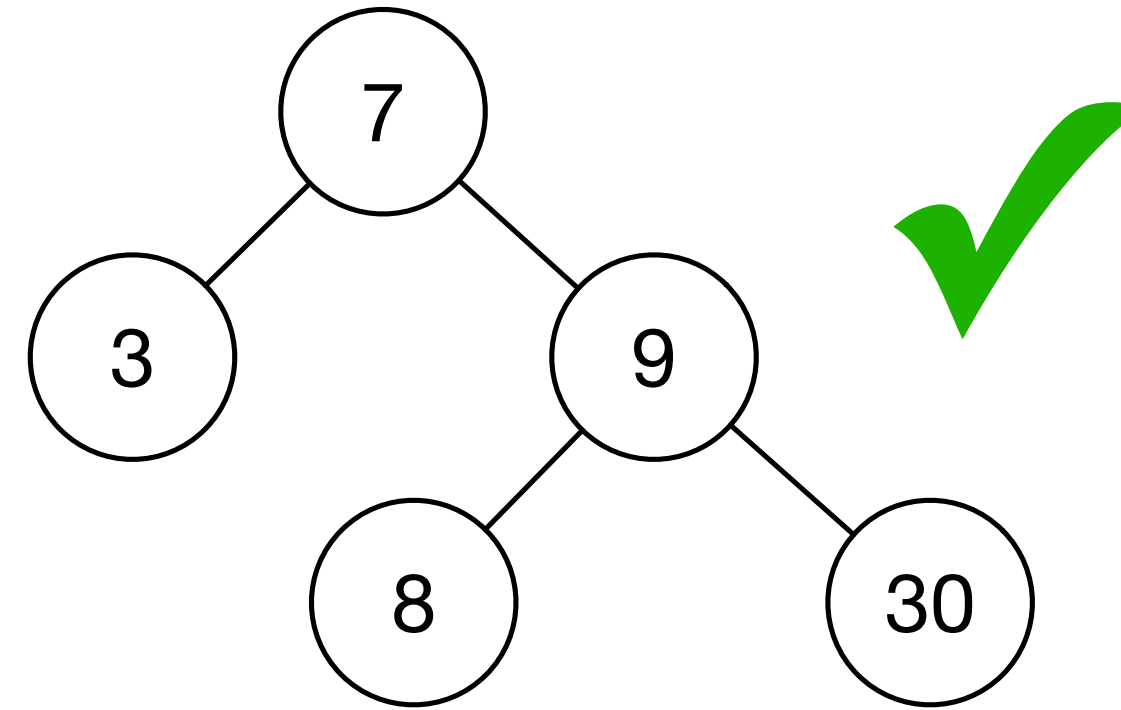
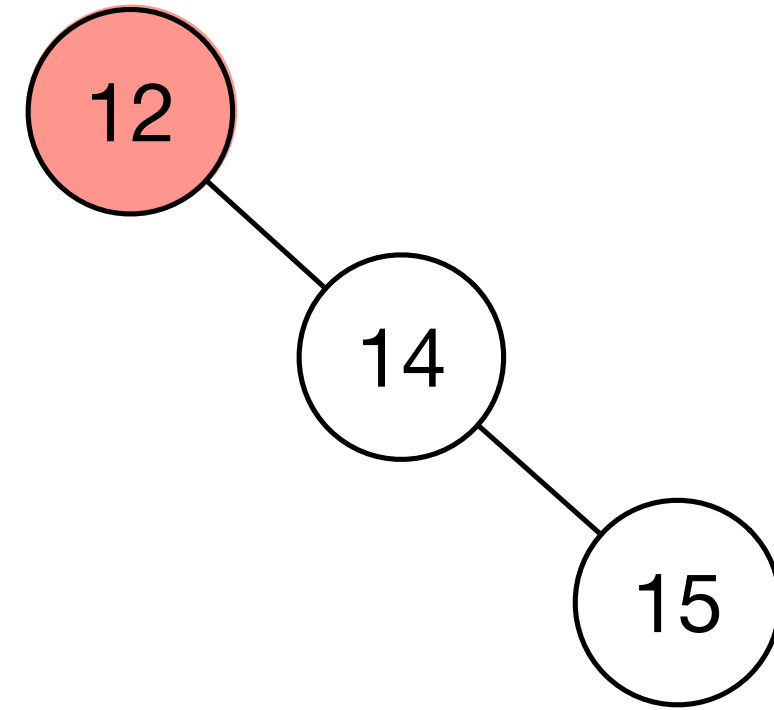




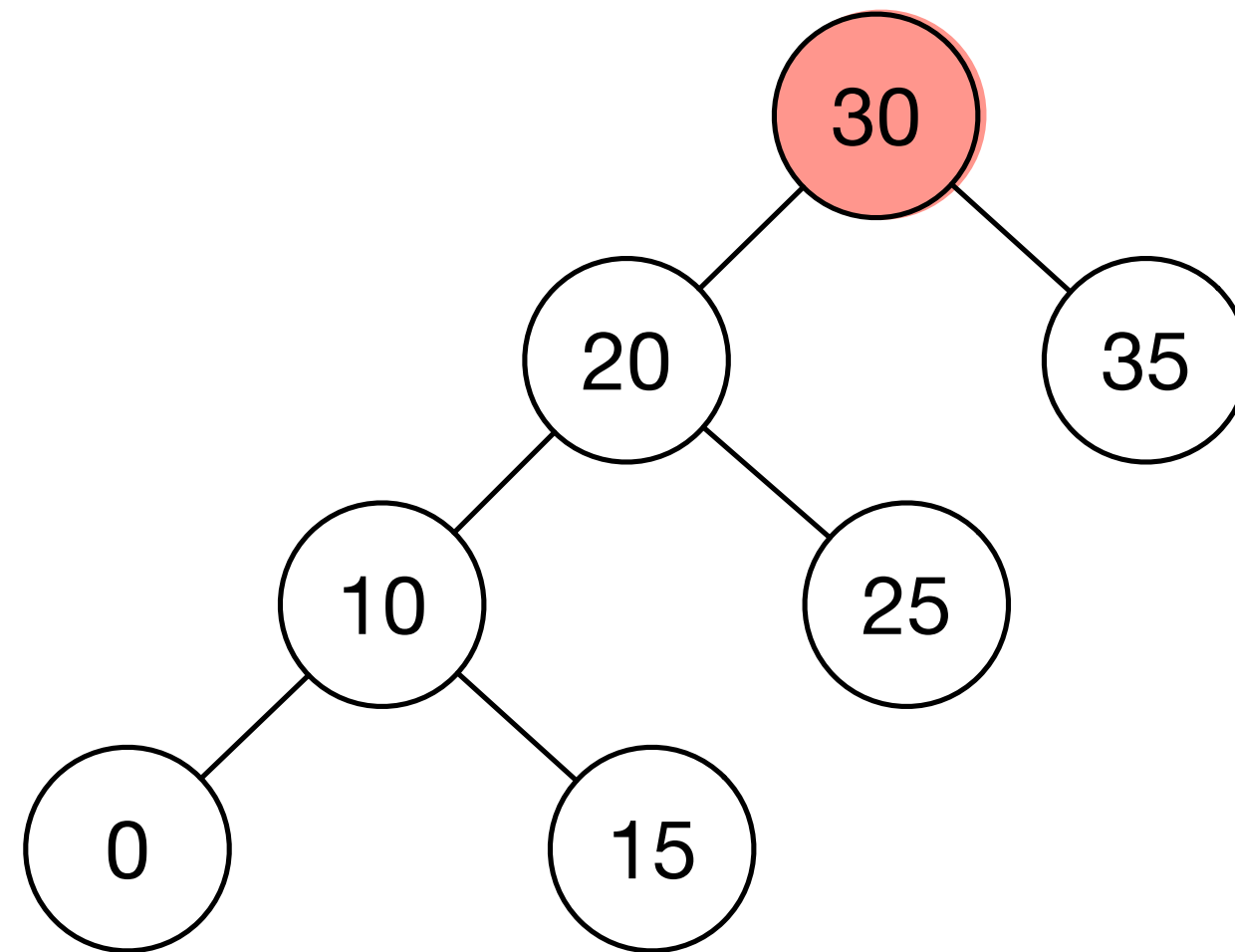
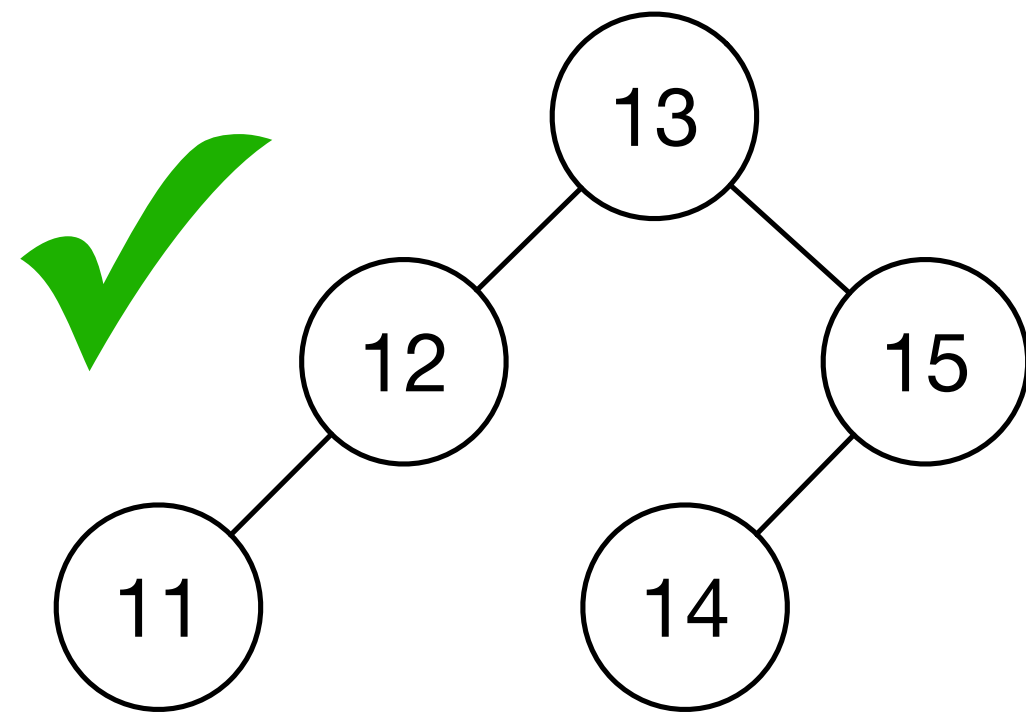
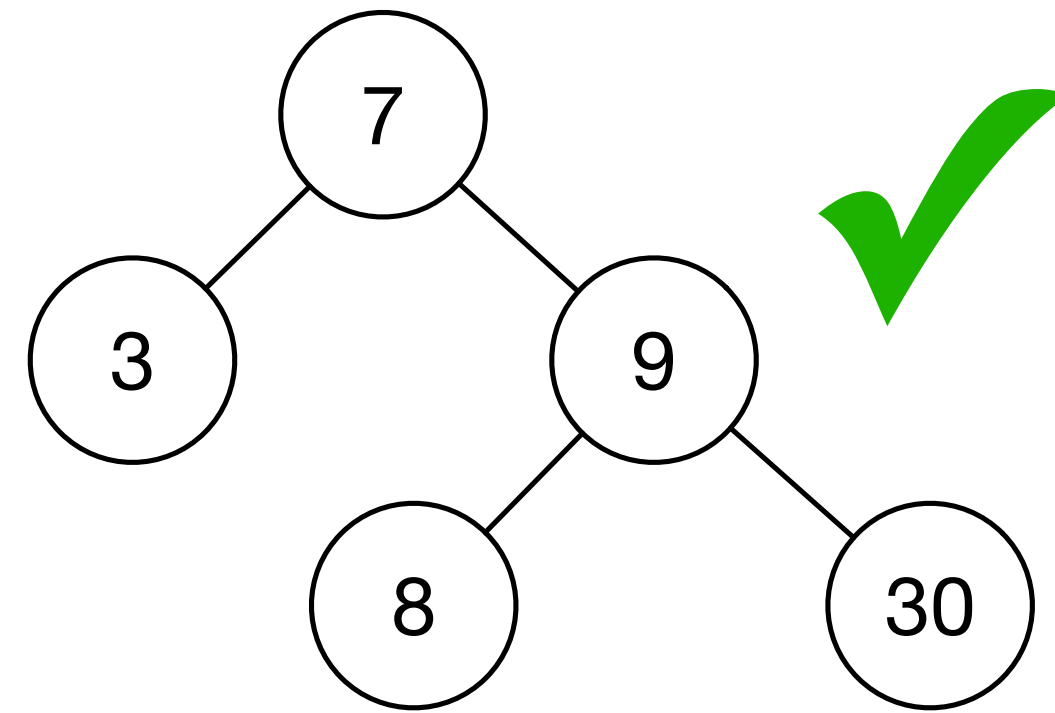
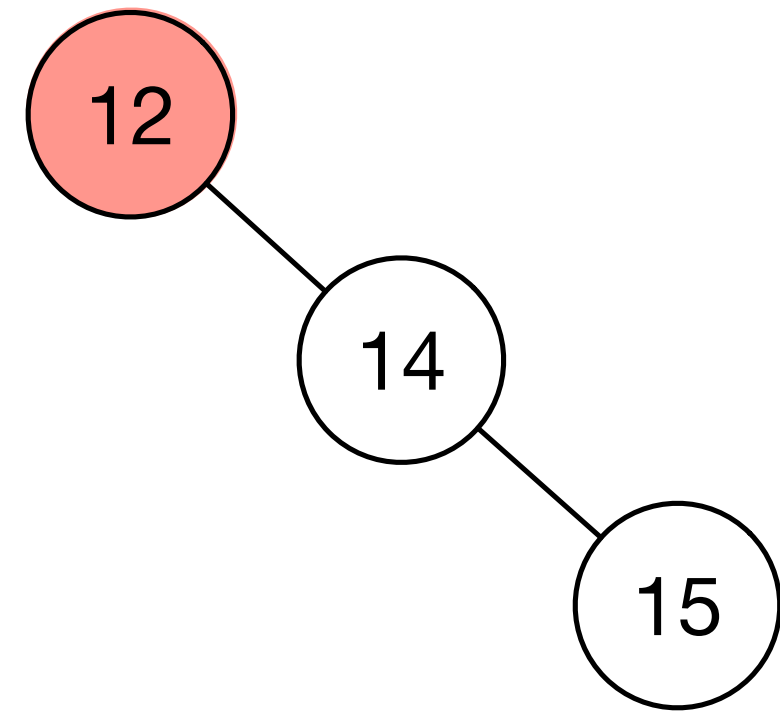
# Which of these are AVL trees?



# Which of these are AVL trees?



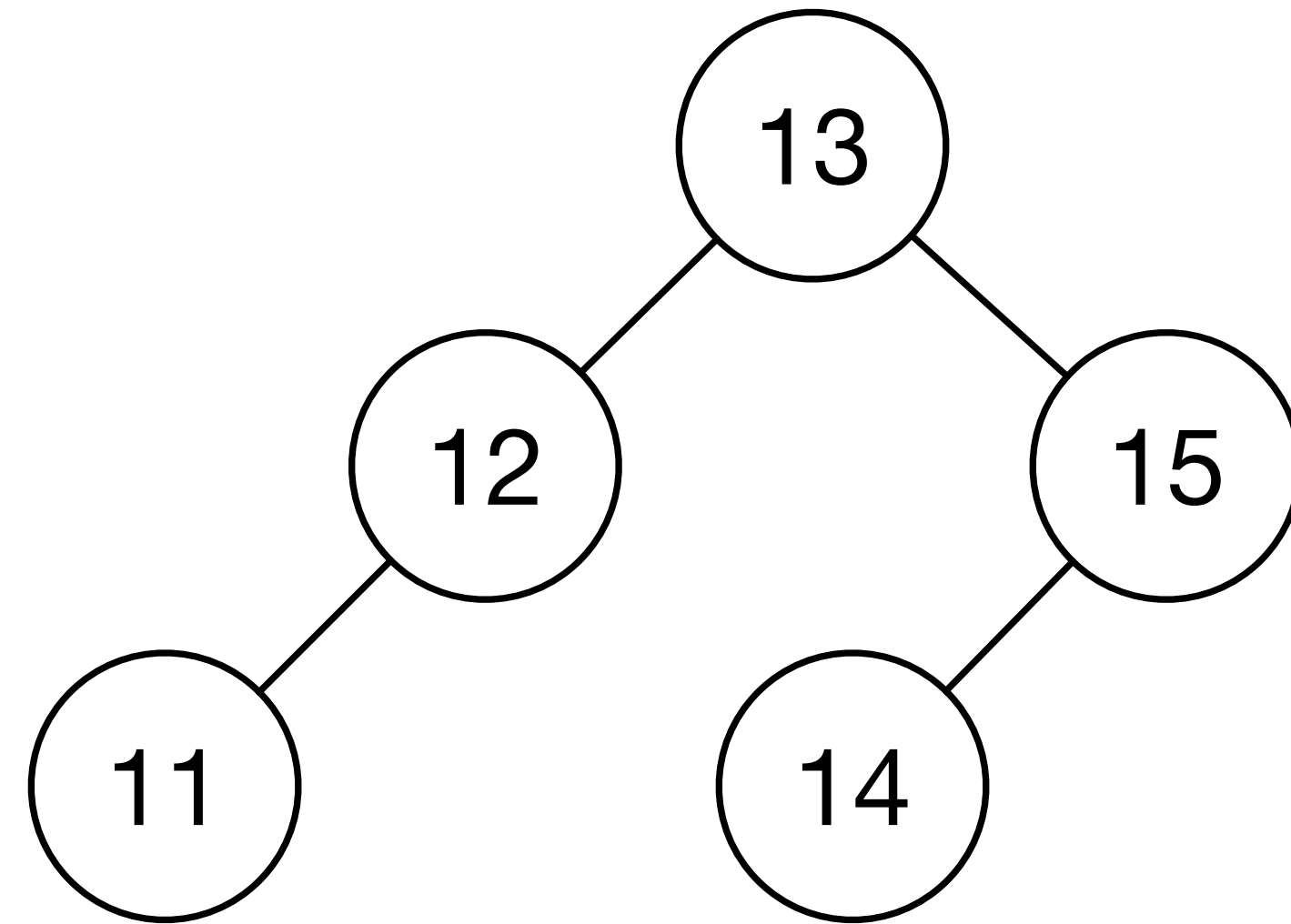
# Which of these are AVL trees?



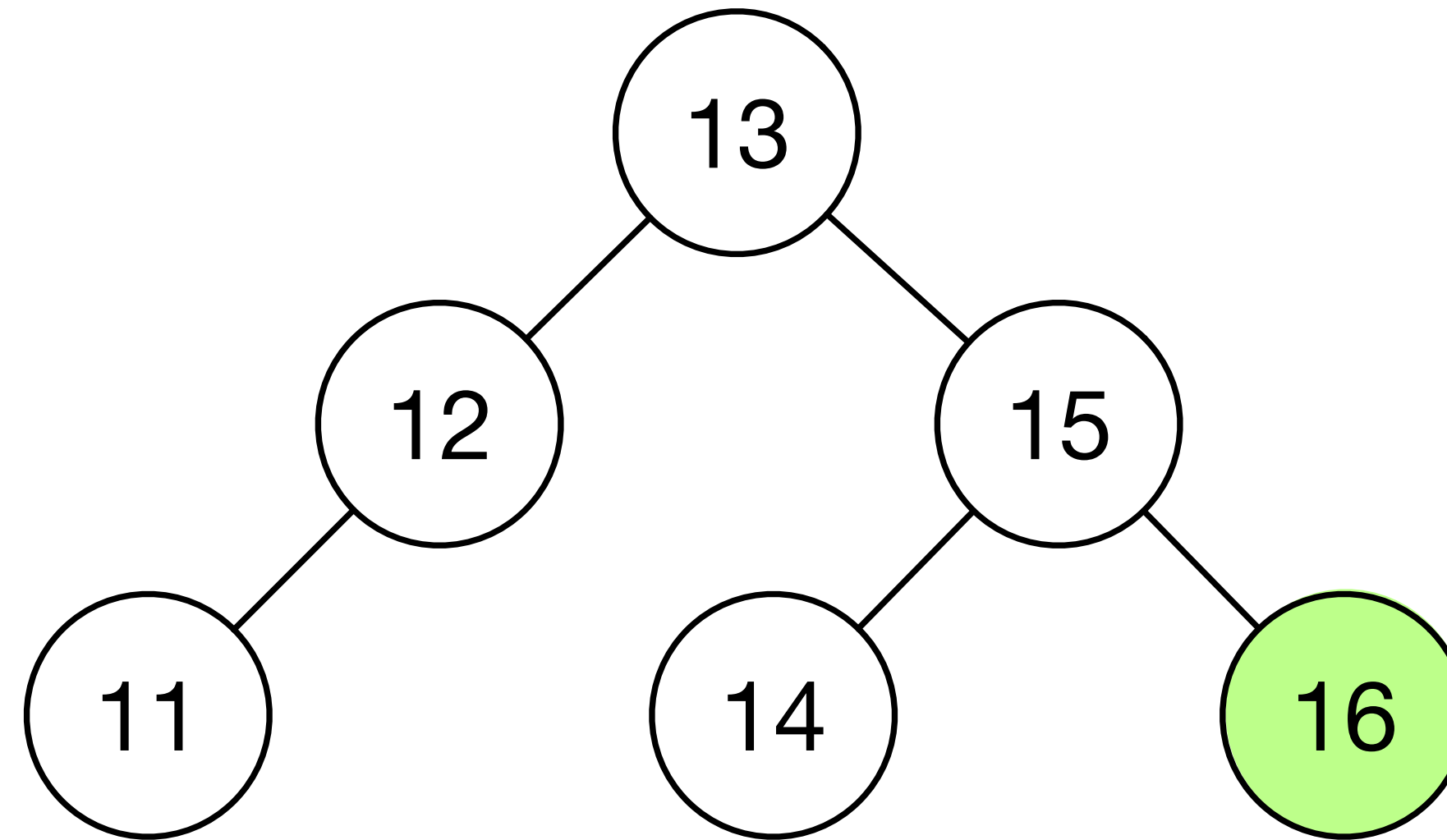
# Self-balancing: Preserving the AVL property

- What does it mean to be "self-balancing"?
- Recall we have methods for handling insertions and deletions for binary search trees that preserve the properties of a binary search tree.
- Similarly, AVL trees have methods for handling insertions and deletions that preserve its BST properties *and* keep it balanced, preserving the AVL property.

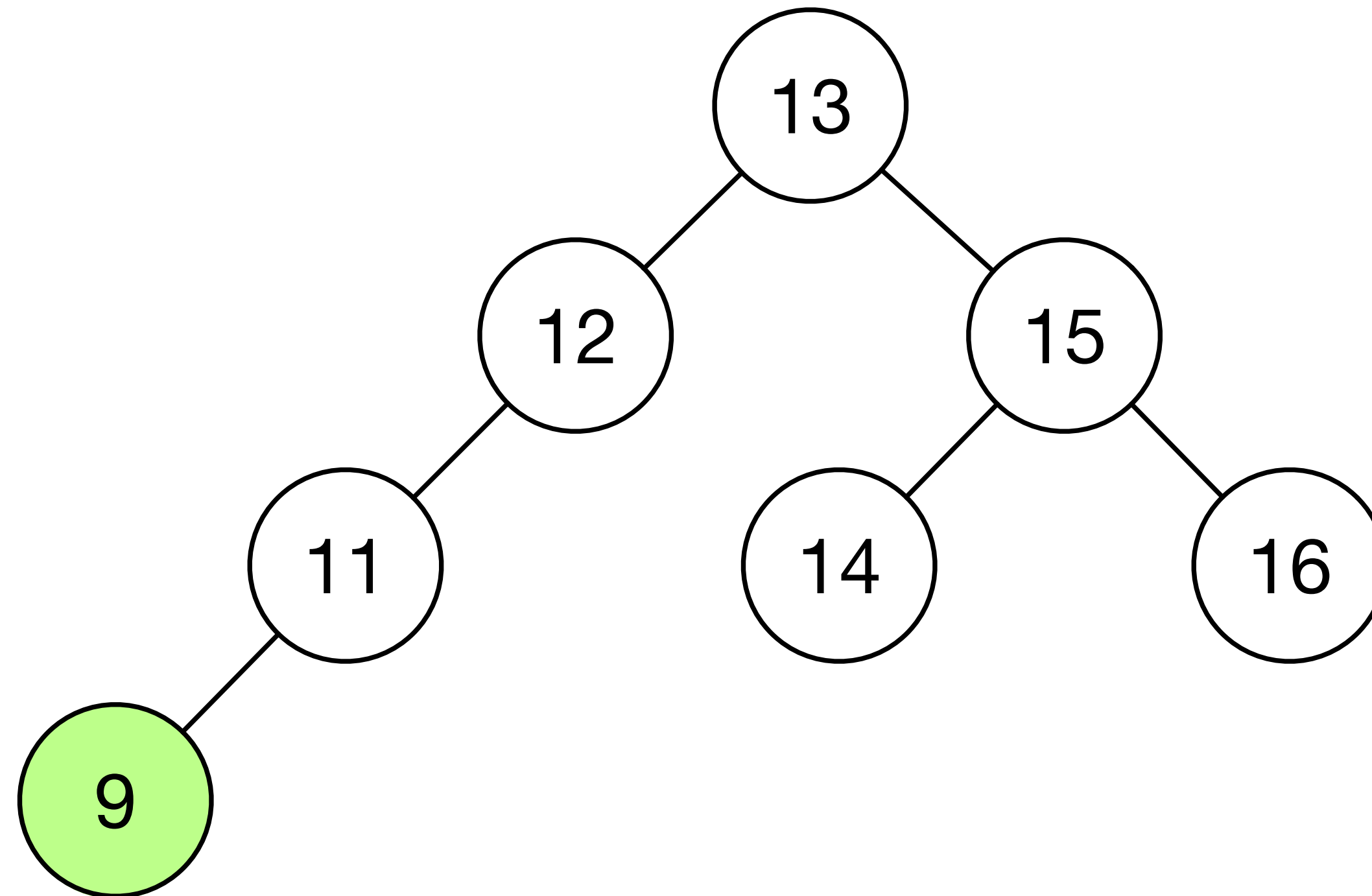
# Self-balancing: Preserving the AVL property



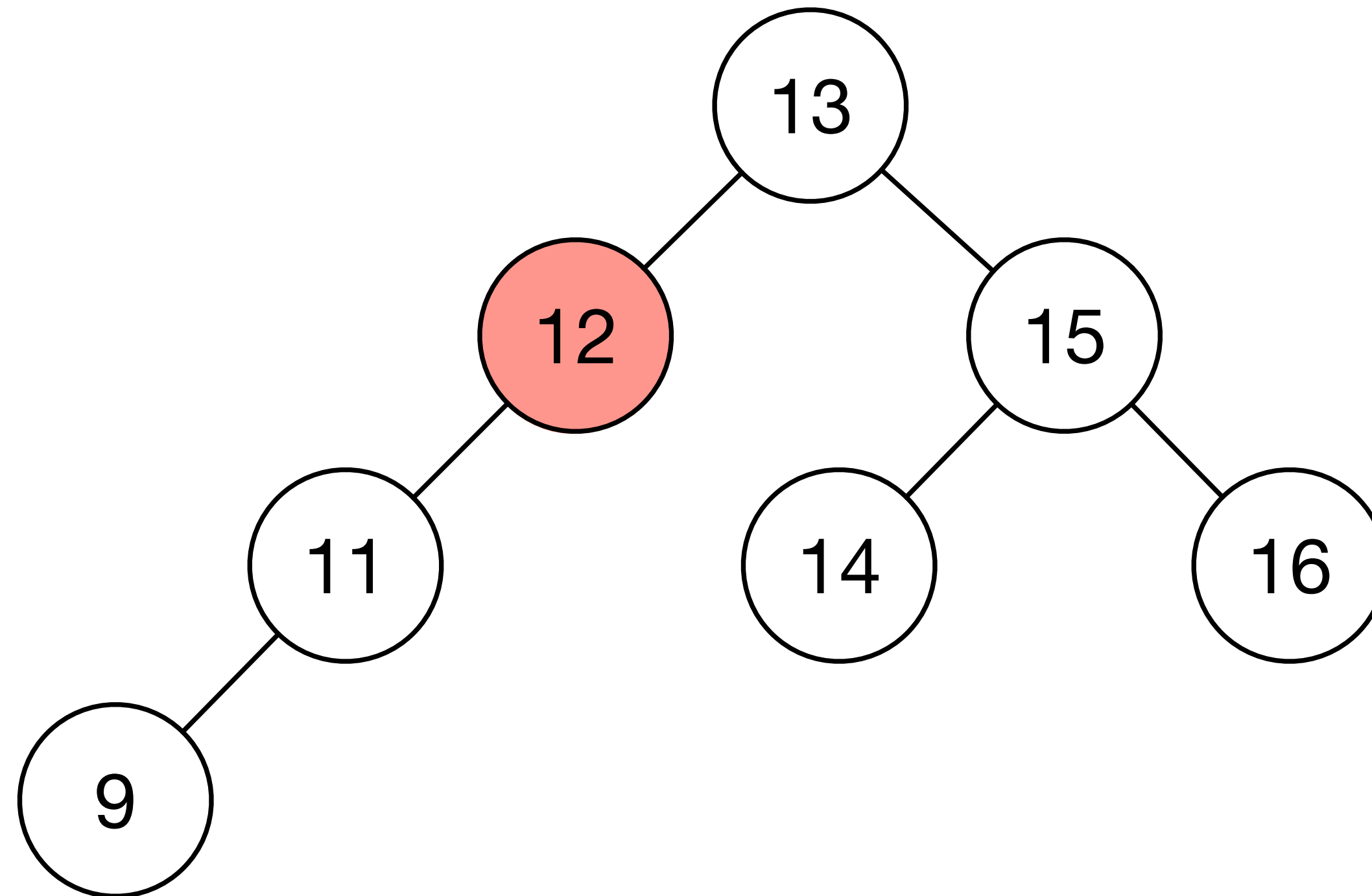
# Self-balancing: Preserving the AVL property



# Self-balancing: Preserving the AVL property

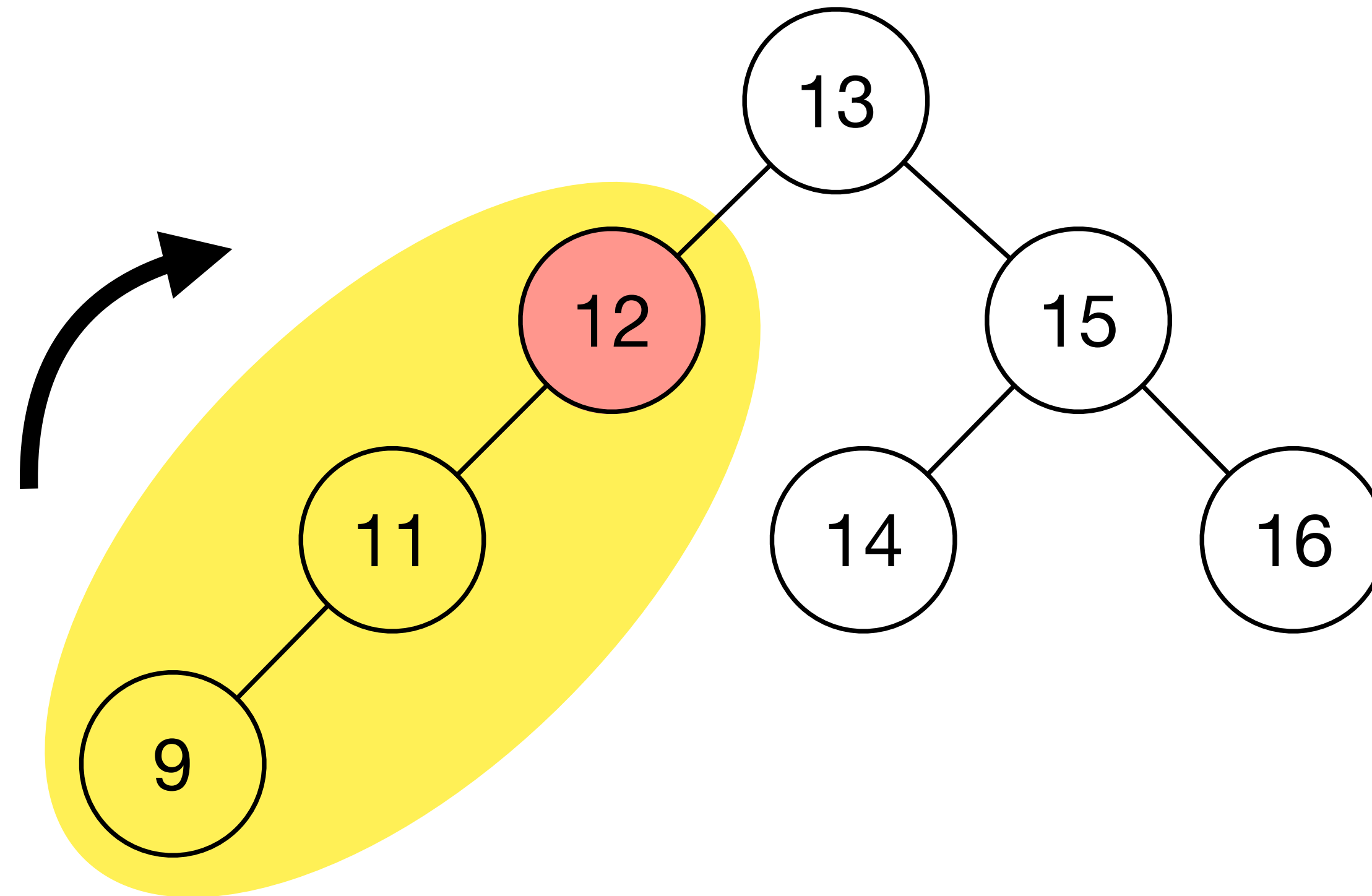


# Self-balancing: Preserving the AVL property

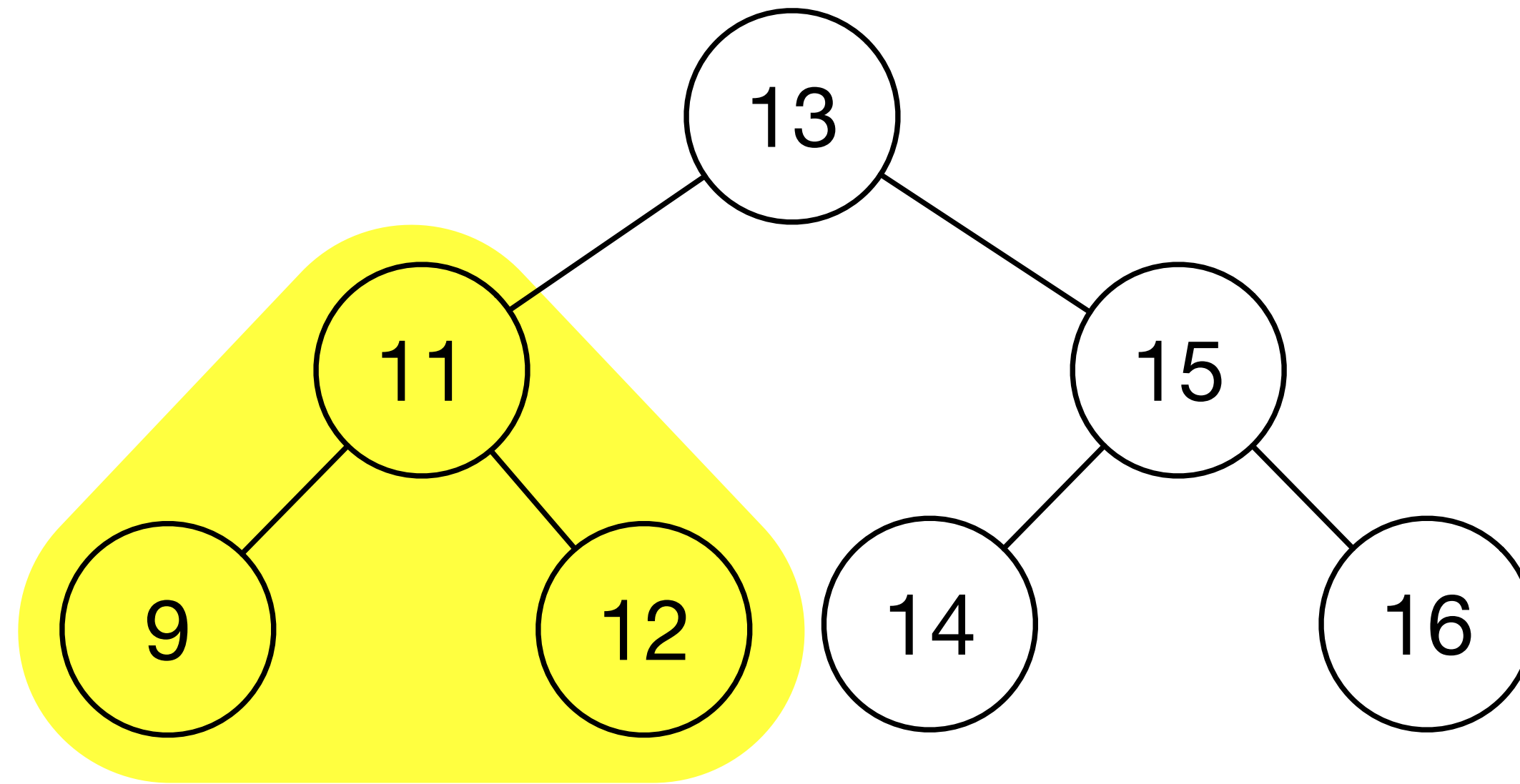




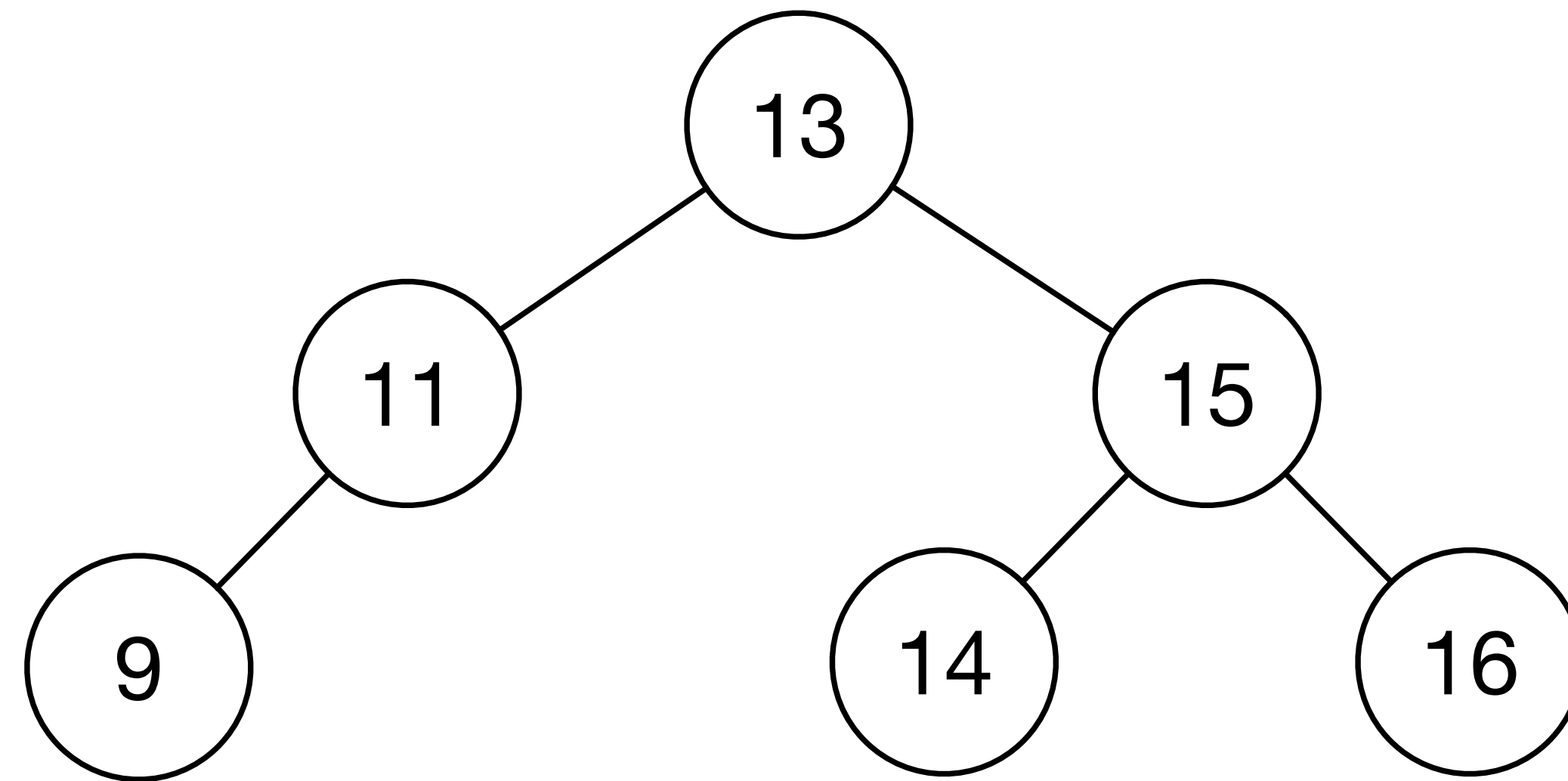
# Self-balancing: Preserving the AVL property



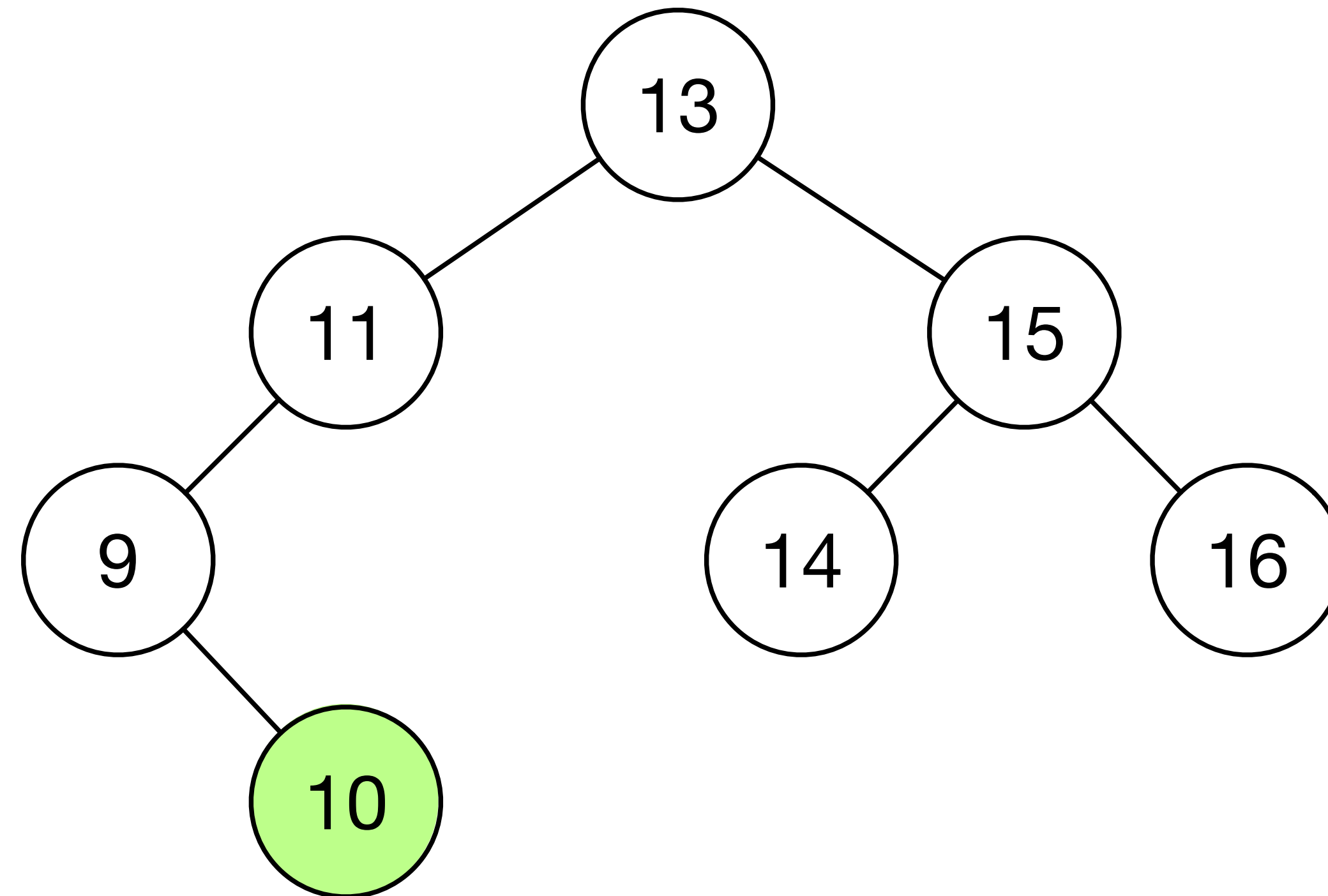
# Self-balancing: Preserving the AVL property



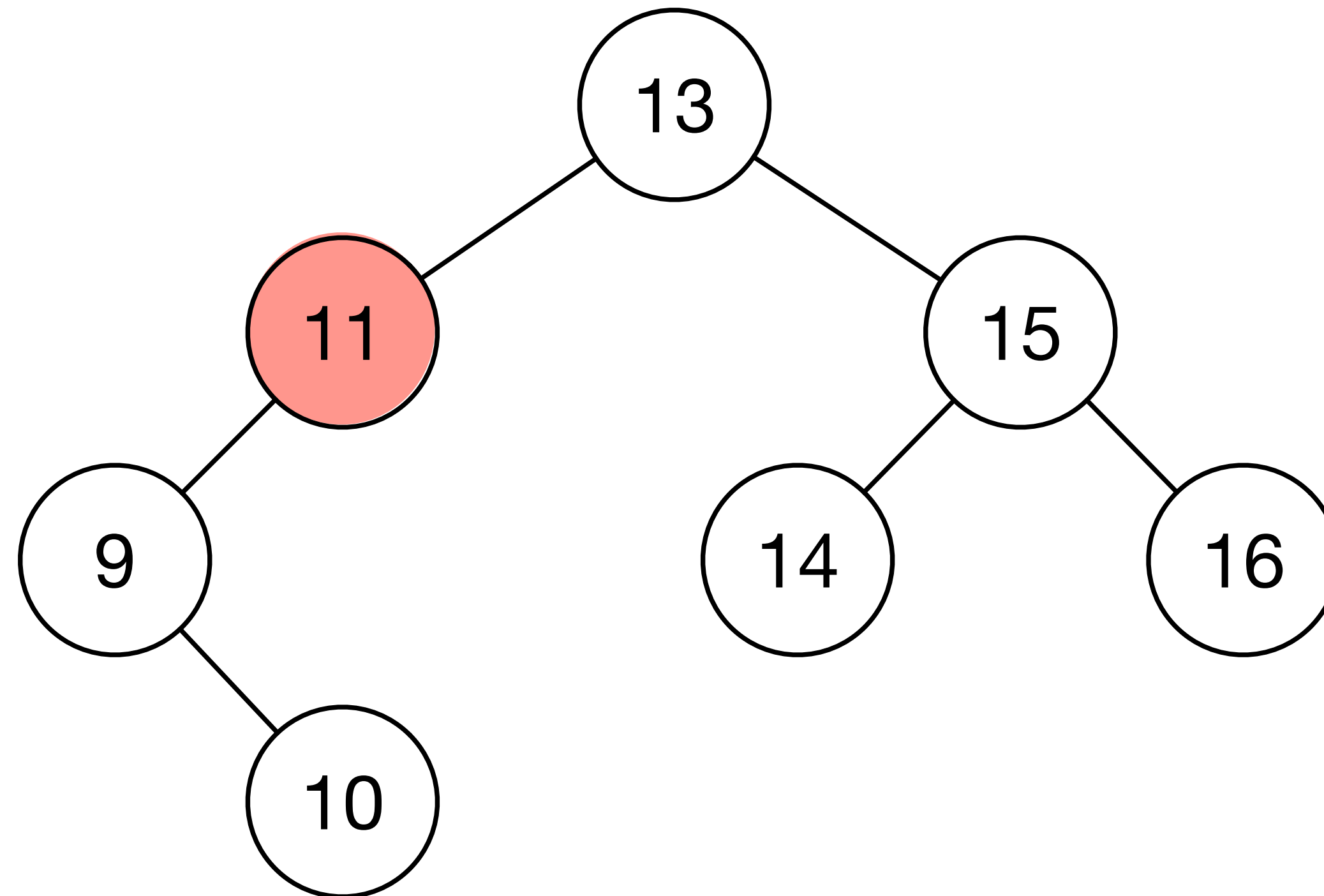
# Self-balancing: Preserving the AVL property



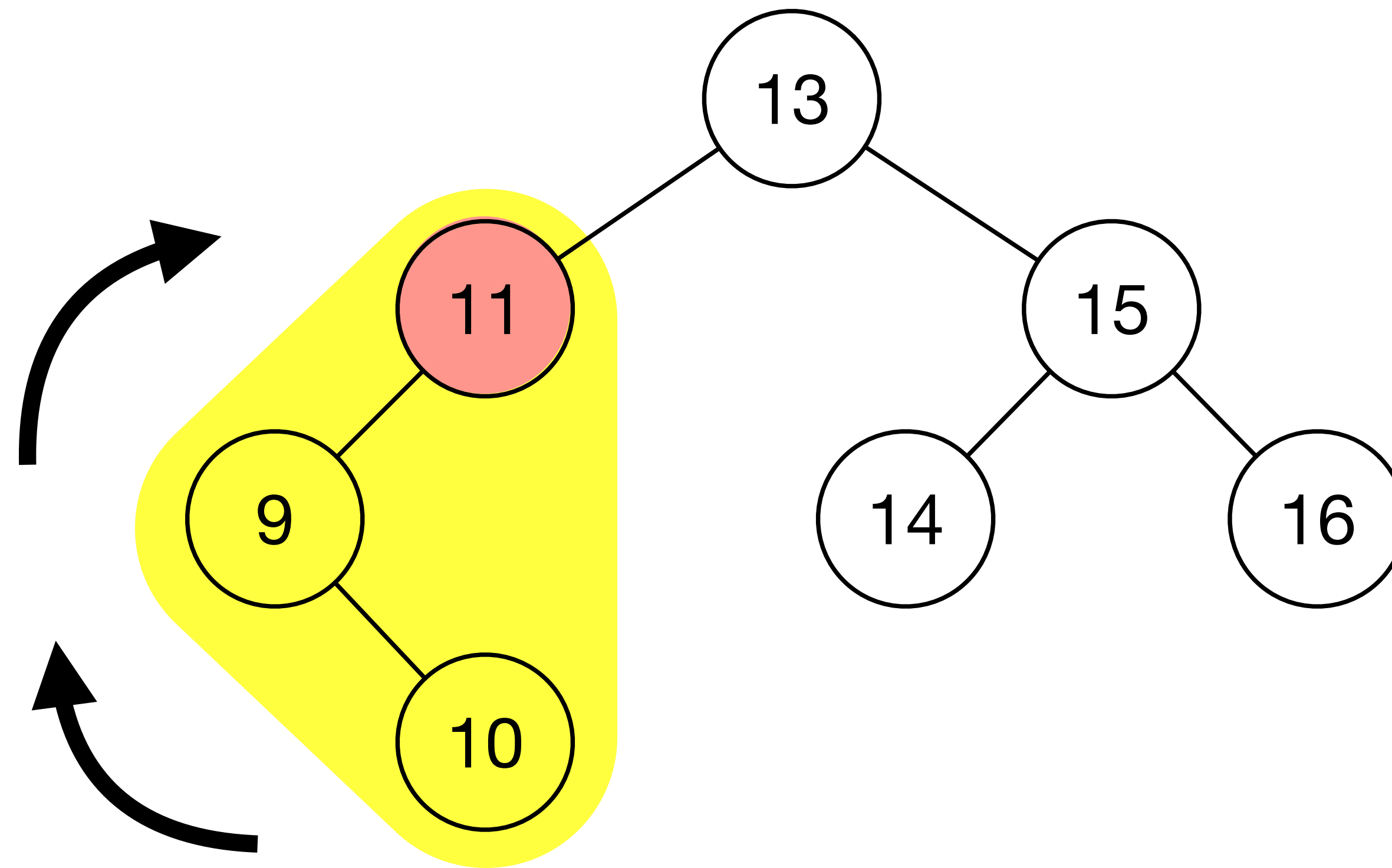
# Self-balancing: Preserving the AVL property



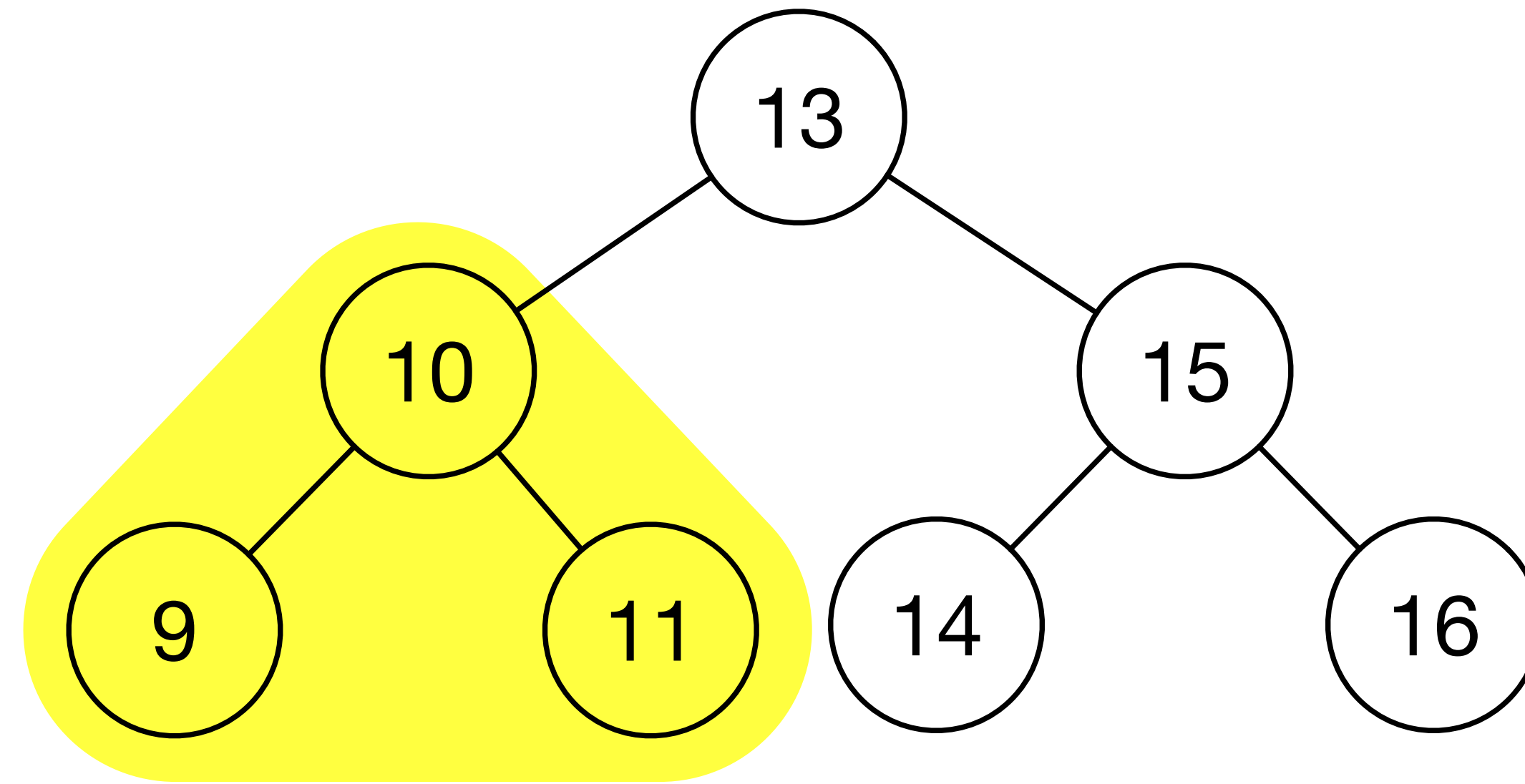
# Self-balancing: Preserving the AVL property



# Self-balancing: Preserving the AVL property



# Self-balancing: Preserving the AVL property



# AVL: General Algorithm

- Perform the insertion or deletion you would as for a BST.
- Check to see if the AVL property holds -- that is, for each node in the tree, the height of its left and right subtrees differ by at most one.
- If the tree does not have the AVL property, find the lowest node in the tree that has subtrees differing in height by more than one -- this is the "problem node"
- Perform necessary rotation(s).



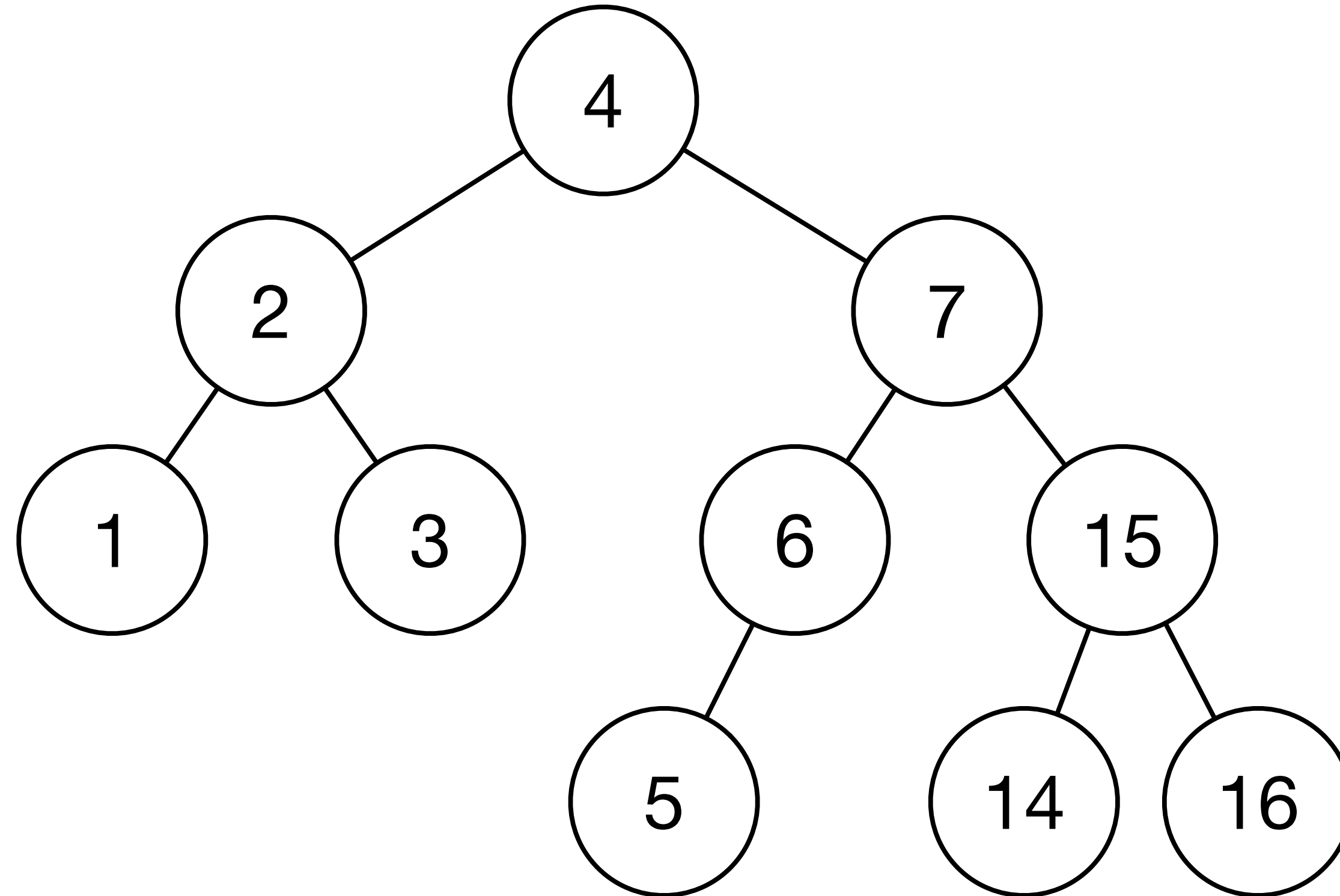
# AVL: General Algorithm

## Determining the necessary rotation

- Examine the path from the "problem node" to its most distant leaf.
- "Zig-zig": If the path goes in the same direction for the first two generations (*i.e.*, both left children or both right children), do a single rotation.
- "Zig-zag": If the path follows different directions (*i.e.*, a left child then a right child, or *vice versa*), do a double rotation.
- If there are multiple leaves that are farthest from the problem node and both paths are possible, either rotation will work to make the tree AVL again.

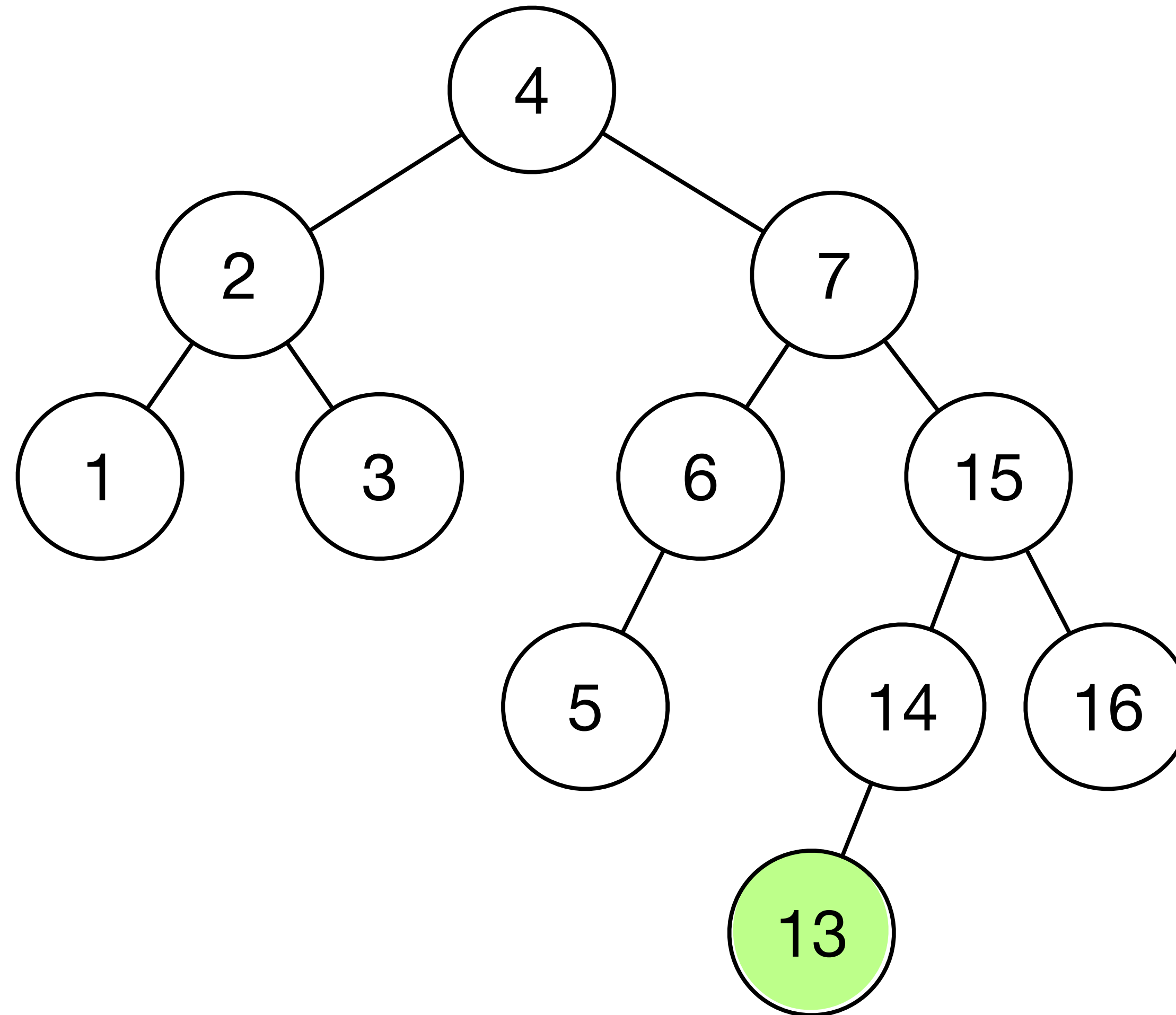
# Self-balancing: Preserving the AVL property

Another example



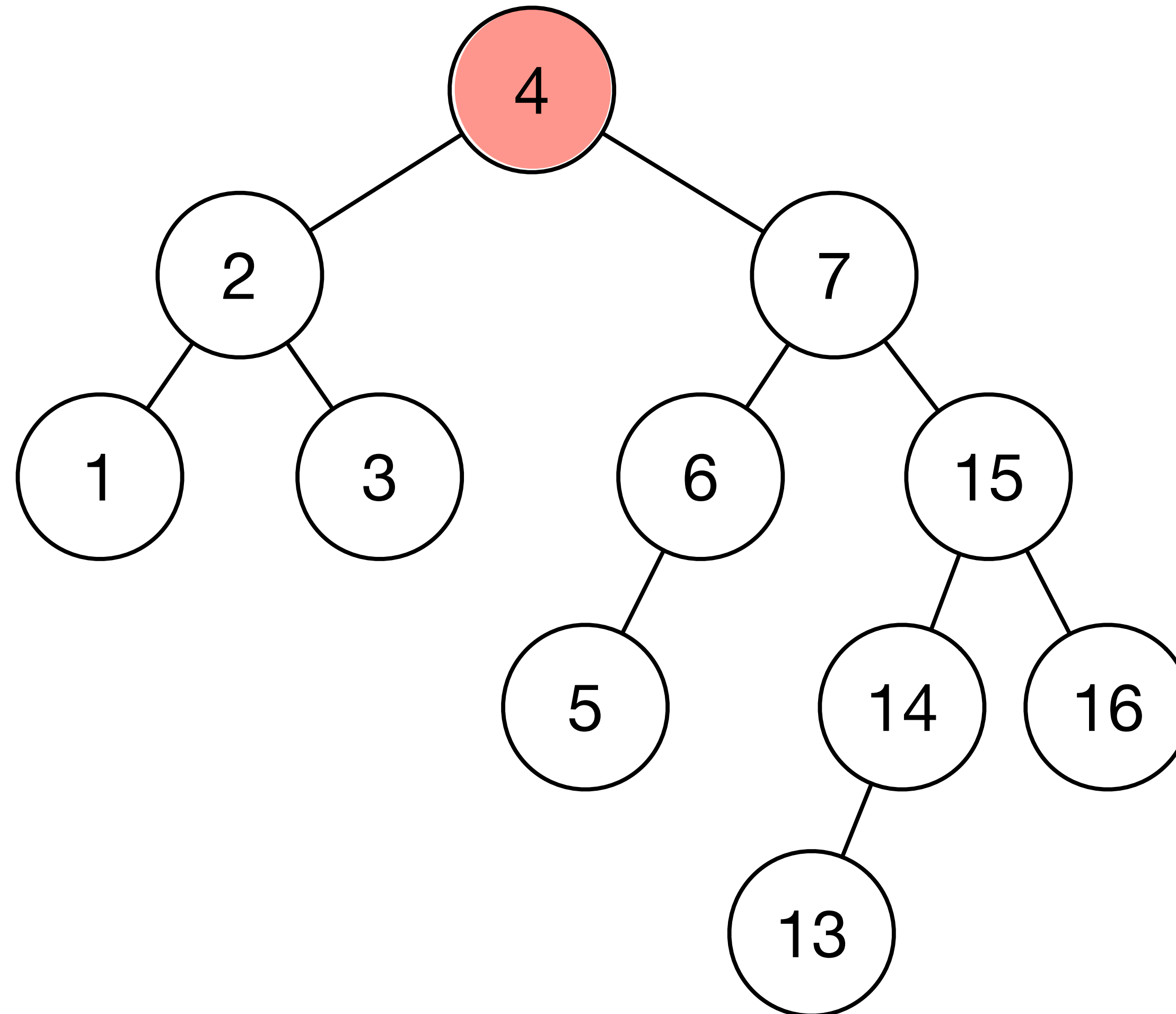
# Self-balancing: Preserving the AVL property

Another example



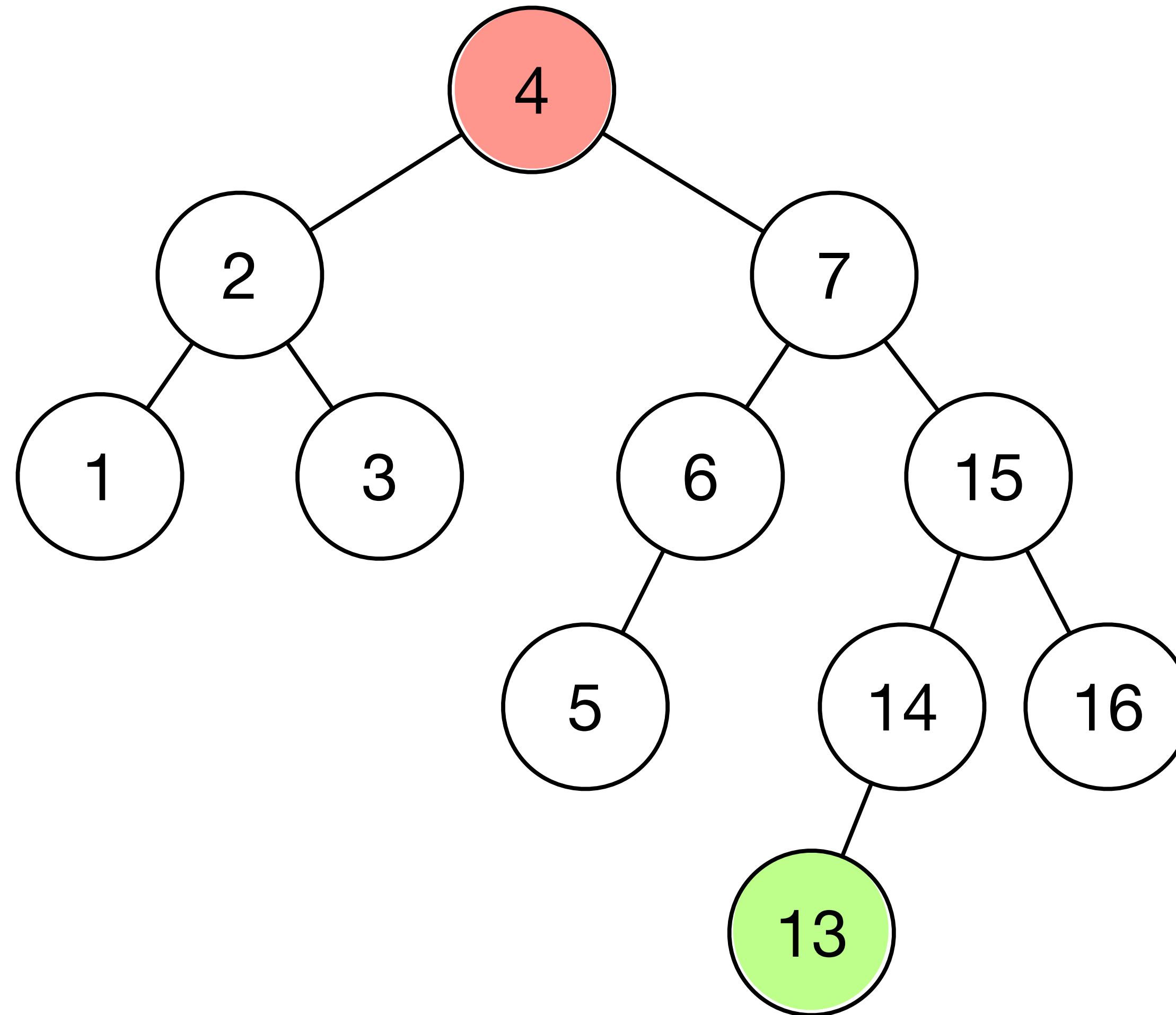
# Self-balancing: Preserving the AVL property

Another example



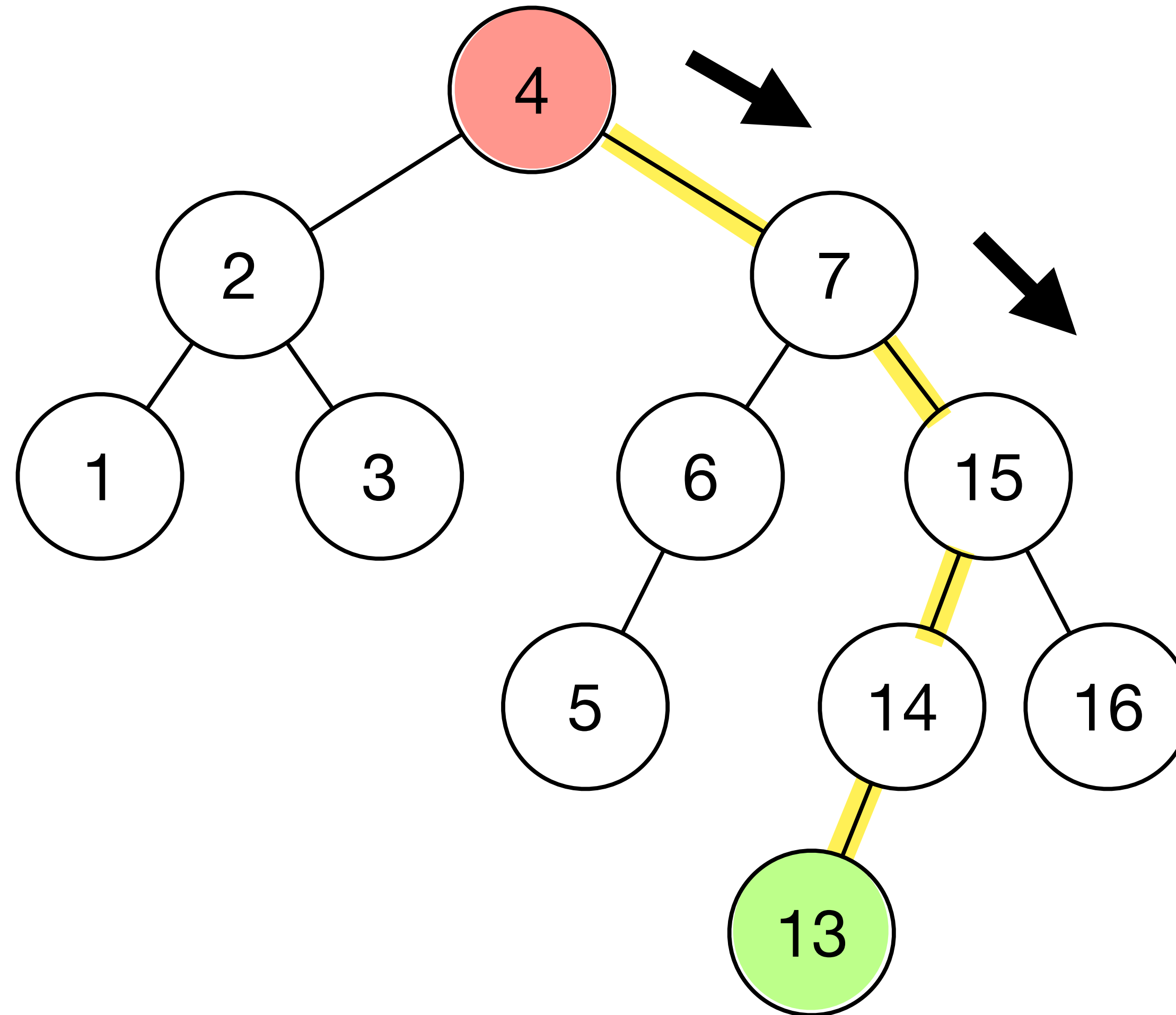
# Self-balancing: Preserving the AVL property

Another example



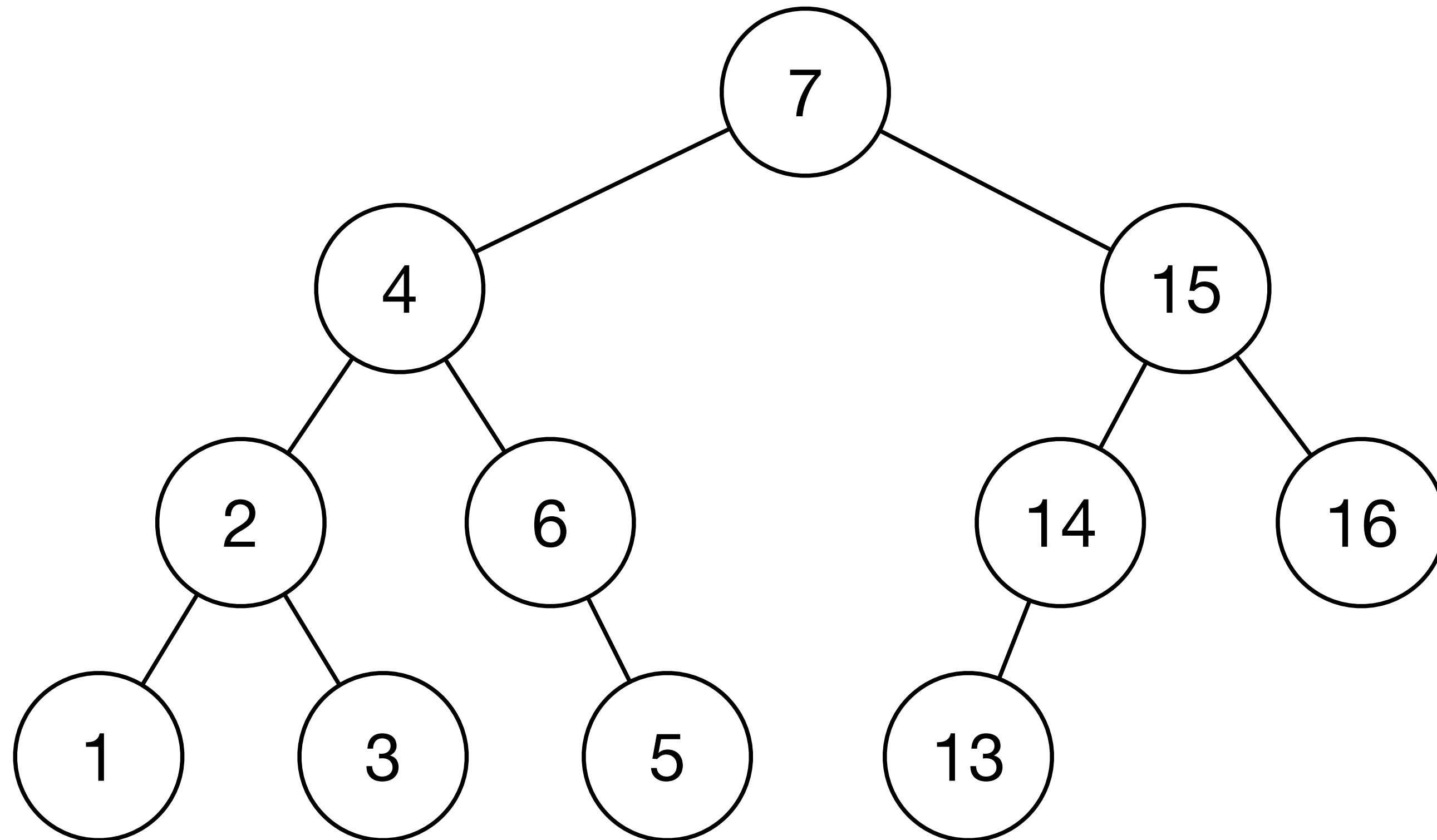
# Self-balancing: Preserving the AVL property

Another example



# Self-balancing: Preserving the AVL property

Another example



# Some visualisations: a helpful tool

The image shows a screenshot of a web-based AVL Tree visualization tool. At the top, a dark green header contains the text "AVL Tree" in yellow. Below this is a light green control bar with buttons for "Insert", "Delete", "Find", and "Print", each preceded by a small empty input field. The main area of the tool is currently blank. At the bottom, another light green control bar contains a status indicator "Animation Completed", a set of navigation buttons ("Skip Back", "Step Back", "Pause", "Step Forward", "Skip Forward"), an "Animation Speed" slider, and input fields for "w: 1000" and "h: 500". To the right of these are buttons for "Change Canvas Size" and "Move Controls". A dark green footer at the very bottom contains the text "Algorithm Visualizations" in yellow.

<https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>