CSC263 Tutorial #4 Exercises

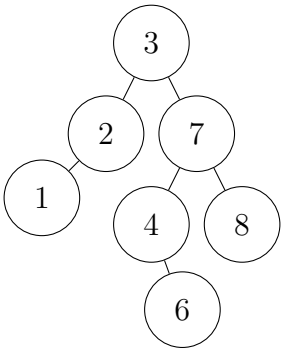
AVL Trees

Alexander He Meng

Prepared for January 31, 2025

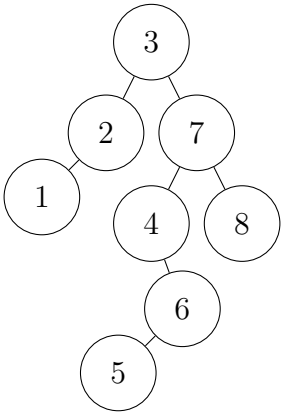
Question 1

Consider the following AVL tree. What happens if we insert a node 5 into the tree? Which node is the lowest ancestor to become unbalanced? What type of rotations need to be performed to rebalance the tree? Perform the AVL-INSERT carefully step by step.



Solution

When we insert a node 5 into the tree, the tree will look like the following:



The lowest ancestor to become unbalanced is node 4. Since the tree is now right-heavy on node 4 due to the insertion of 5, we need to perform a **left rotation** at node 4.

AVL-Insert Process:

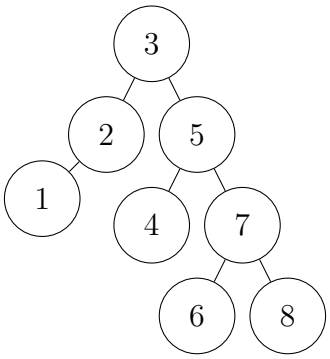
1. Insert 5 as the left child of 6.

2. Update heights and balance factors.

3. Detect imbalance at node 4 (balance factor = 2).

4. Perform a left rotation at node 4, making 6 the new root of the subtree.

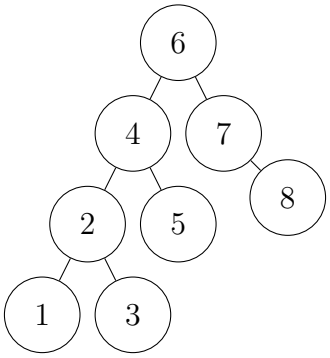
After rotations, the final balanced tree is:



Thus, the tree is balanced again after the left rotation.

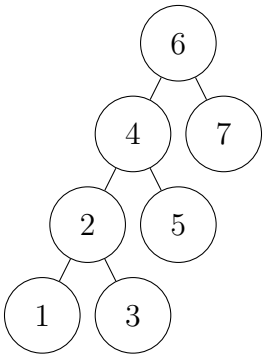
Question 2

Consider the following AVL tree. What happens if we delete the node 8 from the tree? What type of rotations are needed to rebalance the tree? What's the height of the tree before deletion? What's the height of the tree after deletion?



Solution

When we delete the node 8 from the tree, the tree will look like the following:



AVL-Deletion Process

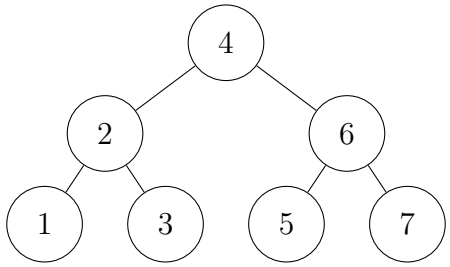
1. **Remove Node 8**
    - Since node 8 is a leaf node, we can delete it directly.
2. **Update Heights and Balance Factors**
    - Before deletion, the height of the right subtree was **3**.

- After deletion, we recalculate the heights:
  - The new height of node 7 is **0** (since it has no children).
  - The new height of node 4 remains **3**, and its balance factor is recalculated.

3. **Check for Imbalances**

- Node 6 has a balance factor of **2** (height of left subtree = 4, height of right subtree = 2).
- Since the balance factor is **not** within the range $[-1, 1]$, and is left heavy, we need to perform a **right rotation** at node 4.

After this right rotation, the tree will be balanced again. The final tree will look like the following:



4. **Perform Right Rotation at Node 6**

- Node 4 becomes the new root.
- Node 6 moves down as the right child of 4.
- The subtree remains balanced.

5. **Final Heights and Balance Factors**

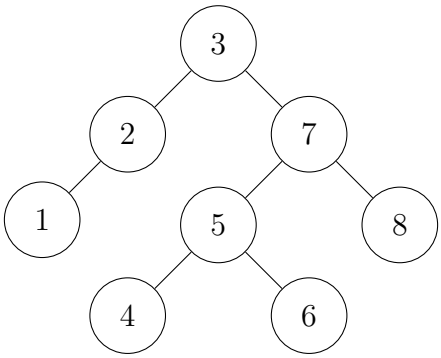
- **Height before deletion**: 4
- **Height after deletion and rebalancing**: 3
- All nodes now have balance factors within $[-1, 1]$, ensuring the tree is a valid AVL tree.

Thus, after deleting node 8 and performing a right rotation at node 6, the AVL tree is successfully rebalanced.

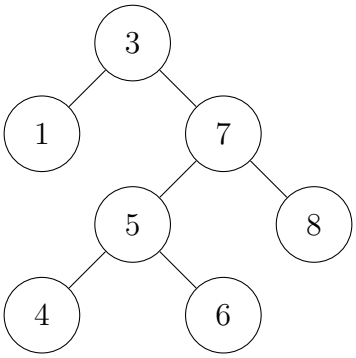
Question 3

Consider the following AVL-tree. What happens if we delete the node 2 from the tree? What type of rotations are needed to rebalance the tree? What's the height of the tree before deletion? What's the height of the tree after deletion?

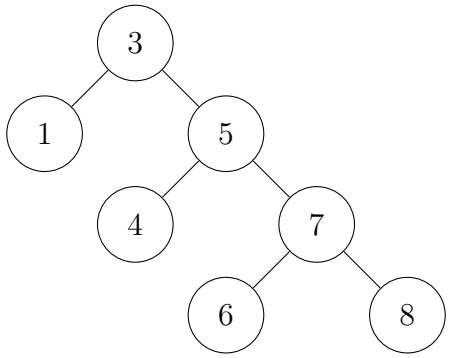
Initial AVL Tree



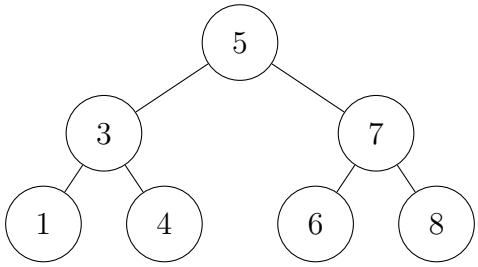
After Deletion of 2 (Replacing with 1)



After Right Rotation on 5



After Left Rotation on 5 (Final Balanced Tree)



Height Analysis

Before deletion, the height of the tree is 4. After deletion and rebalancing, the height becomes 3. The tree is now balanced and satisfies the AVL property.

Question 4

Come up with an example AVL-tree for which deleting a node from the tree causes two levels of double-rotations.

Proof. **Solution:** *(An AVL-tree example and explanation will be provided here.)* □

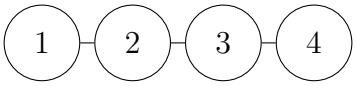
Question 5

Draw a picture of the structure of the AVL tree for which deleting a node would cause $O(\log n)$ rotations, where n is the number of nodes in the tree.

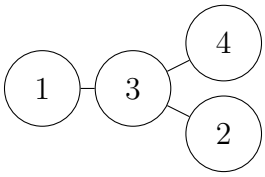
Proof. **Solution:** *(Diagram and explanation will be provided here.)* □

Question 6

Suppose in the AVL-INSERT operation, the tree looks like the following right after inserting the new node and before rebalancing.



Since node 2 is the lowest ancestor that is unbalanced, we do a left rotation around 2, which gives us the following tree:



This tree is still not AVL, which contradicts what we said in the lecture that AVL-INSERT requires only one level of rotation. What went wrong?

Proof. **Solution:** *(Explanation of why this contradiction occurs will be provided here.)* □