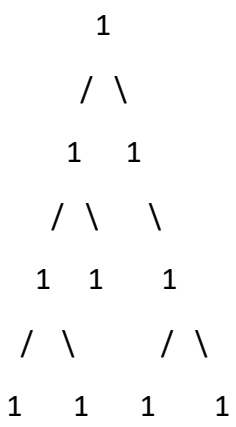
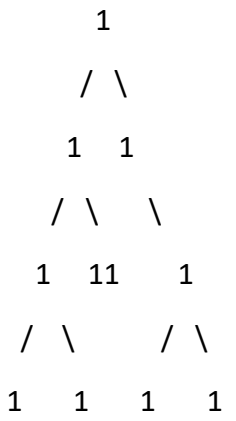
Uni-Valued Binary Tree LeetCode

A binary tree is **uni-valued** if every node in the tree has the same value.

Example 1: Uni-Valued Binary Tree:



Example 2: Multi-Value Binary Tree



Approach 1: Recursive approach to check if a binary tree is univalued.

- In the **solve** helper function, you recursively check if a binary tree is univalued.
- The base case checks if the current node is null, in which case it's considered univalued.
- If the current node's value doesn't match the given value (**val**), the tree is not univalued, and the function returns false.
- The function then recursively checks both the left and right subtrees.
- The **isUnivalTreeRecursively** function calls the helper function to determine if the entire tree is univalued.

Time Complexity: O(N), where N is the number of nodes in the binary tree. You visit each node once.

Space Complexity: O(H), where H is the height of the binary tree due to the function call stack.

Approach 2: Iterative approach to check if a binary tree is univalued

- The **isUnivalTree** function uses an iterative approach to check if a binary tree is univalued.
- It initializes a stack with the root node and enters a while loop.
- In each iteration, it pops a node from the stack and checks if its value matches the given value (**val**).
- If a mismatch is found, the tree is not univalued, and the function returns false.
- If the values match, it pushes the left and right children onto the stack for further processing.
- The loop continues until the stack is empty, and if it does, the function returns true.

Time Complexity: O(N), where N is the number of nodes in the binary tree. You visit each node once.

Space Complexity: O(W), where W is the maximum width of the binary tree at any level.