Problem 623: Add One Row to Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 64.09%
Paid Only: No
Tags: Tree, Depth-First Search, Breadth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree and two integers `val` and `depth`, add a row of nodes with value `val` at the given depth `depth`.

Note that the `root` node is at depth `1`.

The adding rule is:

* Given the integer `depth`, for each not null tree node `cur` at the depth `depth - 1`, create two tree nodes with value `val` as `cur`'s left subtree root and right subtree root. * `cur`'s original left subtree should be the left subtree of the new left subtree root. * `cur`'s original right subtree should be the right subtree of the new right subtree root. * If `depth == 1` that means there is no depth `depth - 1` at all, then create a tree node with value `val` as the new root of the whole original tree, and the original tree is the new root's left subtree.

Example 1:

![](https://assets.leetcode.com/uploads/2021/03/15/addrow-tree.jpg)

Input: root = [4,2,6,3,1,5], val = 1, depth = 2 **Output:** [4,1,1,2,null,null,6,3,1,5]

Example 2:

![](https://assets.leetcode.com/uploads/2021/03/11/add2-tree.jpg)

Input: root = [4,2,null,3,1], val = 1, depth = 3 **Output:** [4,2,null,1,1,3,null,null,1]

Constraints:

* The number of nodes in the tree is in the range `[1, 104]`. * The depth of the tree is in the range `[1, 104]`. * `-100 <= Node.val <= 100` * `-105 <= val <= 105` * `1 <= depth <= the depth of tree + 1`

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
TreeNode* addOneRow(TreeNode* root, int val, int depth) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode addOneRow(TreeNode root, int val, int depth) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def addOneRow(self, root: Optional[TreeNode], val: int, depth: int) ->
Optional[TreeNode]:
```