623. Add One Row to Tree

Description HintsSubmissionsSolutions

Discuss Editorial Solution Pick One

Given the root of a binary tree, then value v and depth d, you need to add a row of nodes with value v at the given depth d. The root node is at depth 1.

The adding rule is: given a positive integer depth d, for each NOT null tree nodes N in depth d-1, create two tree nodes with valuev as N's left subtree root and right subtree root. And N's **original left subtree** should be the left subtree of the new left subtree root, its **original right subtree** should be the right subtree of the new right subtree root. If depth d is 1 that means there is no depth d-1 at all, then create a tree node with value **v** as the new root of the whole original tree, and the original tree is the new root's left subtree.

Example 1:

```
Input:
A binary tree as following:
       4
     /   \
    2     6
   / \   /
  3   1 5

v = 1

d = 2

Output:
       4
```

```
      / \
     1   1
    /     \
   2        6
  / \     /
 3   1   5
```

Example 2:

Input:

A binary tree as following:

```
      4
     /
    2
   / \
  3   1
```

v = 1

d = 3

Output:

```
      4
     /
    2
   / \
  1   1
 /     \
3       1
```

Note:

1. The given d is in range [1, maximum depth of the given tree + 1].
2. The given binary tree has at least one tree node.

Seen this question in a real interview before?

Yes

```
Author:MR.BLACK
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:

    void levelTraverse(TreeNode* root, int v, int d, int level)
    {
        if(root==NULL) return;
        if(level == d-1)
        {
            TreeNode *root_left = root->left;
            TreeNode *root_right = root->right;
            TreeNode *add_left = new TreeNode(v);
            TreeNode *add_right = new TreeNode(v);
            root->left = add_left;
            root->right = add_right;
            add_left->left = root_left;
            add_right->right = root_right;
        }
        levelTraverse(root->left,v,d,level+1);
        levelTraverse(root->right,v,d,level+1);
    }
    TreeNode* addOneRow(TreeNode* root, int v, int d) {
        if(d==1)
        {
            TreeNode *newNode = new TreeNode(v);
            newNode->left = root;
            return newNode;
        }
        int level = 1;
        TreeNode *p = root;
        levelTraverse( p,  v,  d, level);
        return root;
    }
};
```