Add One Row to Tree

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 value `v` 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.

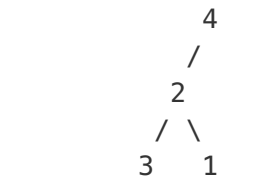
Example 1:

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:

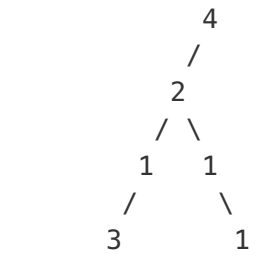
Input:
A binary tree as following:



v = 1

d = 3

Output:



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.

Solution 1

Simply traverse recursively to the depth d - 1 and add nodes accordingly.

```
public class Solution {
 public TreeNode addOneRow(TreeNode root, int v, int d) {
  if (d == 1) {
   TreeNode newRoot = new TreeNode(v);
   newRoot.left = root;
   return newRoot;
  }
  add(root, v, d, 1);
  return root;
 }

 private void add(TreeNode node, int v, int d, int currentDepth) {
  if (node == null) {
   return;
  }

  if (currentDepth == d - 1) {
   TreeNode temp = node.left;
   node.left = new TreeNode(v);
   node.left.left = temp;

   temp = node.right;
   node.right = new TreeNode(v);
   node.right.right = temp;
   return;
  }

  add(node.left, v, d, currentDepth + 1);
  add(node.right, v, d, currentDepth + 1);
 }
}
```

written by soumyadeep2007 original link here

Solution 2

The idea is to:
In addition to use `1` to indicate `attach to left node` as required, we can also use `0` to indicate `attach to right node`;

Compact C++

```
class Solution {
public:
    TreeNode* addOneRow(TreeNode* root, int v, int d) {
        if (d == 0 || d == 1) {
            TreeNode* newroot = new TreeNode(v);
            (d ? newroot->left : newroot->right) = root;
            return newroot;
        }
        if (root && d >= 2) {
            root->left  = addOneRow(root->left,  v, d > 2 ? d - 1 : 1);
            root->right = addOneRow(root->right, v, d > 2 ? d - 1 : 0);
        }
        return root;
    }
};
```

Compact Java

```
public class Solution {
    public TreeNode addOneRow(TreeNode root, int v, int d) {
        if (d == 0 || d == 1) {
            TreeNode newroot = new TreeNode(v);
            newroot.left = d == 1 ? root : null;
            newroot.right = d == 0 ? root : null;
            return newroot;
        }
        if (root != null && d >= 2) {
            root.left  = addOneRow(root.left,  v, d > 2 ? d - 1 : 1);
            root.right = addOneRow(root.right, v, d > 2 ? d - 1 : 0);
        }
        return root;
    }
}
```

Plain C++

```
class Solution {
public:
    TreeNode* addOneRow(TreeNode* root, int v, int d) {
        if (d == 1) {
            TreeNode* newroot = new TreeNode(v);
            newroot->left = root;
            return newroot;
        }
        else if (d == 0) {
            TreeNode* newroot = new TreeNode(v);
            newroot->right = root;
            return newroot;
        }

        if (!root) {
            return nullptr;
        }
        else if (d == 2) {
            root->left  = addOneRow(root->left,  v, 1);
            root->right = addOneRow(root->right, v, 0);
            return root;
        }
        else if (d > 2) {
            root->left  = addOneRow(root->left,  v, d - 1);
            root->right = addOneRow(root->right, v, d - 1);
        }
        return root;
    }
};
```

written by alexander original link here

Solution 3

Go row by row to the row at depth d-1, then insert the new nodes there.

```
def addOneRow(self, root, v, d):
    dummy, dummy.left = TreeNode(None), root
    row = [dummy]
    for _ in range(d - 1):
        row = [kid for node in row for kid in (node.left, node.right) if kid]
    for node in row:
        node.left, node.left.left = TreeNode(v), node.left
        node.right, node.right.right = TreeNode(v), node.right
    return dummy.left
```

written by StefanPochmann original link here