Count Univalue Subtrees

Given a binary tree, count the number of uni-value subtrees.

A Uni-value subtree means all nodes of the subtree have the same value.

For example:
Given binary tree,

return `4`.

Solution 1

```
public class Solution {
    public int countUnivalSubtrees(TreeNode root) {
        int[] count = new int[1];
        helper(root, count);
        return count[0];
    }

    private boolean helper(TreeNode node, int[] count) {
        if (node == null) {
            return true;
        }
        boolean left = helper(node.left, count);
        boolean right = helper(node.right, count);
        if (left && right) {
            if (node.left != null && node.val != node.left.val) {
                return false;
            }
            if (node.right != null && node.val != node.right.val) {
                return false;
            }
            count[0]++;
            return true;
        }
        return false;
    }
}
```

written by stevenye original link here

Solution 2

Helper `all` tells whether all nodes in the given tree have the given value. And while doing that, it also counts the uni-value subtrees.

```
public class Solution {
    int count = 0;
    boolean all(TreeNode root, int val) {
        if (root == null)
            return true;
        if (!all(root.left, root.val) | !all(root.right, root.val))
            return false;
        count++;
        return root.val == val;
    }
    public int countUnivalSubtrees(TreeNode root) {
        all(root, 0);
        return count;
    }
}
```

written by StefanPochmann original link here

Solution 3

```
public class Solution {
    public int countUnivalSubtrees(TreeNode root) {
        int[] arr = new int[1];
        postOrder(arr, root);
        return arr[0];
    }
    public boolean postOrder (int[] arr, TreeNode node) {
        if (node == null) return true;
        boolean left = postOrder(arr, node.left);
        boolean right = postOrder(arr, node.right);
        if (left && right) {
            if (node.left != null && node.left.val != node.val) return false;
            if (node.right != null && node.right.val != node.val) return false;
            arr[0]++;
            return true;
        }
        return false;
    }
}
```

written by saybia1993 original link here