Problem 1612: Check If Two Expression Trees are Equivalent

Problem Information

Difficulty: Medium
Acceptance Rate: 71.47%
Paid Only: Yes
Tags: Hash Table, Tree, Depth-First Search, Binary Tree, Counting

Problem Description

A **[binary expression tree](https://en.wikipedia.org/wiki/Binary_expression_tree)** is a kind of binary tree used to represent arithmetic expressions. Each node of a binary expression tree has either zero or two children. Leaf nodes (nodes with 0 children) correspond to operands (variables), and internal nodes (nodes with two children) correspond to the operators. In this problem, we only consider the `'+'` operator (i.e. addition).

You are given the roots of two binary expression trees, `root1` and `root2`. Return `true` _if the two binary expression trees are equivalent_. Otherwise, return `false`.

Two binary expression trees are equivalent if they **evaluate to the same value** regardless of what the variables are set to.

Example 1:

Input: root1 = [x], root2 = [x] **Output:** true

Example 2:

Input: root1 = [+,a,+,null,null,b,c], root2 = [+,+,a,b,c] **Output:** true **Explanation****:** a + (b + c) == (b + c) + a

Example 3:

Input: root1 = [+,a,+,null,null,b,c], root2 = [+,+,a,b,d] **Output:** false **Explanation****:** a + (b + c) != (b + d) + a

Constraints:

* The number of nodes in both trees are equal, odd and, in the range `[1, 4999]`. * `Node.val` is `'+'` or a lower-case English letter. * It's **guaranteed** that the tree given is a valid binary expression tree.

Follow up: What will you change in your solution if the tree also supports the `'-'` operator (i.e. subtraction)?

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct Node {
* char val;
* Node *left;
* Node *right;
* Node() : val(' '), left(nullptr), right(nullptr) {}
* Node(char x) : val(x), left(nullptr), right(nullptr) {}
* Node(char x, Node *left, Node *right) : val(x), left(left), right(right) {}
* };
*/
class Solution {
public:
bool checkEquivalence(Node* root1, Node* root2) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
```

```
* class Node {
* char val;
* Node left;
* Node right;
* Node() {this.val = ' ';}
* Node(char val) { this.val = val; }
* Node(char val, Node left, Node right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public boolean checkEquivalence(Node root1, Node root2) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class Node(object):
# def __init__(self, val=" ", left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def checkEquivalence(self, root1: 'Node', root2: 'Node') -> bool:
```