

663. Equal Tree Partition

Dec. 10, 2017 | 10.1K views

[Previous](#) [Next](#)



Average Rating: 4.69 (13 votes)

Given a binary tree with n nodes, your task is to check if it's possible to partition the tree to two trees which have the equal sum of values after removing **exactly** one edge on the original tree.

Example 1:

Input:
5
/ \
10 10
/ \
2 3

Output: True
Explanation:
5
/
10

Sum: 15

10
/ \
2 3

Sum: 15

Example 2:

Input:
1
/ \
2 10
/ \
2 20

Output: False
Explanation: You can't split the tree into two trees with equal sum after removing exactly one edge.

Note:

1. The range of tree node value is in the range of [-100000, 100000].

2. $1 \leq n \leq 10000$

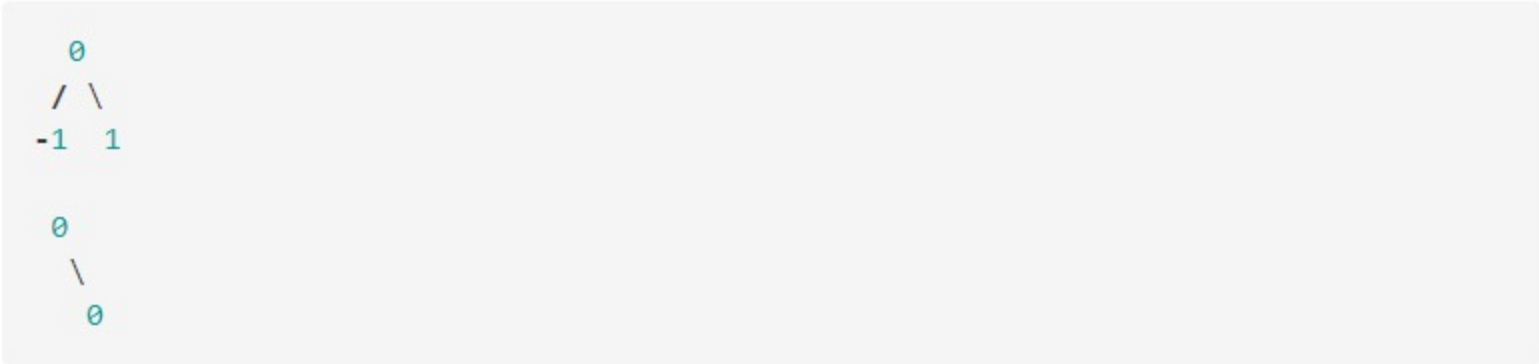
Approach #1: Depth-First Search [Accepted]

Intuition and Algorithm

After removing some edge from **parent** to **child**, (where the **child** cannot be the original **root**) the subtree rooted at **child** must be half the sum of the entire tree.

Let's record the sum of every subtree. We can do this recursively using depth-first search. After, we should check that half the sum of the entire tree occurs somewhere in our recording (and not from the total of the entire tree.)

Our careful treatment and analysis above prevented errors in the case of these trees:



JavaPython

Copy

```
1 class Solution(object):
2     def checkEqualTree(self, root):
3         seen = []
4
5         def sum_(node):
6             if not node: return 0
7             seen.append(sum_(node.left) + sum_(node.right) + node.val)
8             return seen[-1]
9
10        total = sum_(root)
11        seen.pop()
12        return total / 2.0 in seen
```

Complexity Analysis

- Time Complexity: $O(N)$ where N is the number of nodes in the input tree. We traverse every node.

• Space Complexity: $O(N)$, the size of **seen** and the implicit call stack in our DFS.

Analysis written by: @awice.


Rate this article: ★★★★★

[Previous](#)


[Next](#)

Comments: 4


Sort By



Type comment here... (Markdown is supported)

 Preview


[Post](#)



park29★282🕒 July 31, 2019 7:46 AM

We can use O(1) space only.

1👍👎🔗 Share👤 Reply




migfulcrum★485🕒 December 13, 2018 9:31 PM

It would be nice to finish computing as soon as the condition is met.

0👍👎🔗 Share👤 Reply

[SHOW 1 REPLY](#)




sssss29★25🕒 December 2, 2018 4:35 AM

After removing some edge from parent to child, (where the child cannot be the original root) the subtree rooted at child must be half the sum of the entire tree. Why this statement is correct? Can someone explain

0👍👎🔗 Share👤 Reply

[SHOW 2 REPLIES](#)



donne★22🕒 October 12, 2018 6:03 AM

Why do we use stack here? I see the only reason for using stack is to store the sum for each subtree. We can use a set instead. That was look up would be O(1). Am I missing something?

0👍👎🔗 Share👤 Reply

[SHOW 4 REPLIES](#)