Problem 1080: Insufficient Nodes in Root to Leaf Paths

Problem Information

Difficulty: Medium
Acceptance Rate: 54.46%
Paid Only: No
Tags: Tree, Depth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree and an integer `limit`, delete all **insufficient nodes** in the tree simultaneously, and return _the root of the resulting binary tree_.

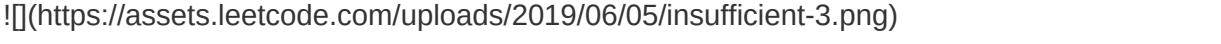
A node is **insufficient** if every root to **leaf** path intersecting this node has a sum strictly less than `limit`.

A **leaf** is a node with no children.

Example 1:

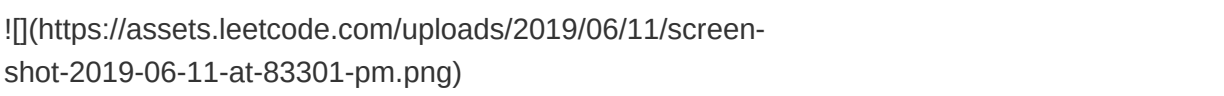
Input: root = [1,2,3,4,-99,-99,7,8,9,-99,-99,12,13,-99,14], limit = 1 **Output:** [1,2,3,4,null,null,7,8,9,null,14]

Example 2:



Input: root = [5,4,8,11,null,17,4,7,1,null,null,5,3], limit = 22 **Output:** [5,4,8,11,null,17,4,7,null,null,null,5]

Example 3:



Input: root = [1,2,-3,-5,null,4,null], limit = -1 **Output:** [1,null,-3,4]

Constraints:

* The number of nodes in the tree is in the range `[1, 5000]`. * `-105 <= Node.val <= 105` * `-109 <= limit <= 109`

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
TreeNode* sufficientSubset(TreeNode* root, int limit) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
```

```
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode sufficientSubset(TreeNode root, int limit) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def sufficientSubset(self, root: Optional[TreeNode], limit: int) ->
Optional[TreeNode]:
```