# 671 Second Minimum Node In a Binary Tree

2018年4月22日 17:03

Given a non-empty special binary tree consisting of nodes with the non-negative value, where each node in this tree has exactly two or zero sub-node. If the node has two sub-nodes, then this node's value is the smaller value among its two sub-nodes.
Given such a binary tree, you need to output the **second minimum** value in the set made of all the nodes' value in the whole tree.
If no such second minimum value exists, output -1 instead.

**Example 1:**

**Input:**

```
  2
 / \
2  5
  / \
 5  7
```

**Output:** 5

**Explanation:** The smallest value is 2, the second smallest value is 5.

**Example 2:**

**Input:**

```
  2
 / \
2  2
```

**Output:** -1

**Explanation:** The smallest value is 2, but there isn't any second smallest value.

来自 <https://leetcode.com/problems/second-minimum-node-in-a-binary-tree/description/>

给定一个非空特殊的二叉树，每个节点都是正数，并且每个节点的子节点数量只能为 2 或 0。如果一个节点有两个子节点的话，那么这个节点的值不大于它的子节点的值。

给出这样的一个二叉树，你需要输出所有节点中的**第二小的值**。如果第二小的值不存在的话，输出 -1 .

## Solution for Python3:

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution:
    def findSecondMinimumValue(self, root):
        """
        :type root: TreeNode
        :rtype: int
        """
        if not root:
            return -1
        return self.smVal(root, root.val)
    def smVal(self, root, first):
        if not root:
            return -1
        if root.val != first:
            return root.val
        left, right = self.smVal(root.left, first), self.smVal(root.right, first)
        if left == -1:
            return right
```

```
        if right == -1:
            return left
        return min(left, right)
```

## Solution for C++:

```
class Solution {
public:
    int findSecondMinimumValue(TreeNode* root) {
        if (!root)
            return -1;
        int ans = smVal(root, root->val);
        return ans;
    }
    int smVal(TreeNode* root, int first) {
        if (root == NULL)
            return -1;
        if (root->val != first)
            return root->val;
        int left = smVal(root->left, first), right = smVal(root->right, first);
        if (left == -1)
            return right;
        if (right == -1)
            return left;
        return min(left, right);
    }
};
```