链滴

# [每日 LeetCode] 671. Second Minimum Node In a Binary Tree

**Description:**

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 his node's value is the smaller value among its two sub-nodes. More formally, the property root.val = min(root.left.val, root.right.val) always holds.

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.

---

思路：本题要求二叉树中存在的第二小的数。这里使用STL中的set容器，set可以在快速插入元素的时对元素自动排序。遍历的方法采用 [每日LeetCode] 102. Binary Tree Level Order Traversal 层序遍历思想。

---

C++代码

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
```

```cpp
 */
class Solution {
public:
    int findSecondMinimumValue(TreeNode* root) {
        set<int> vals;
        stack<TreeNode*> nodes;
        nodes.push(root);
        while (!nodes.empty()) {//BFS
            TreeNode* node = nodes.top();
            nodes.pop();
            vals.insert(node->val);
            if (node->left)
                nodes.push(node->left);
            if (node->right)
                nodes.push(node->right);
        }
        set<int>::iterator it = vals.begin();
        if (vals.size() > 1)//取第二个元素
            return *(++it);
        else
            return -1;
    }
};
```

---

运行时间：4ms

运行内存：8.6M