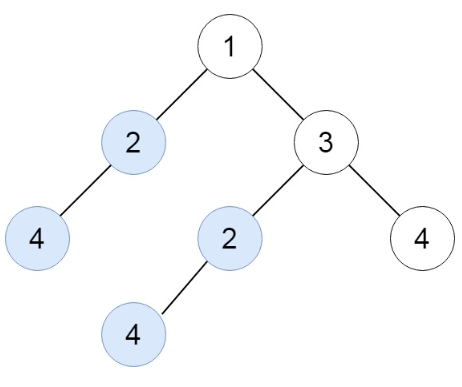
# 题目

给定一棵二叉树，返回所有重复的子树。对于同一类的重复子树，你只需要返回其中任意一棵的根结点即可。

两棵树重复是指它们具有相同的结构以及相同的结点值。

示例 1：



1

/ \

2 3

/ / \

4 2 4

/

4

下面是两个重复的子树：

2

/

4

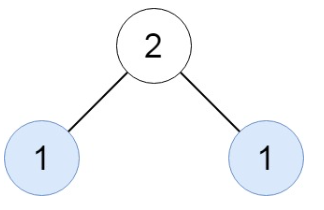
和

4

输入：root = [1,2,3,4,null,2,4,null,null,4]

输出：[[2,4],[4]]

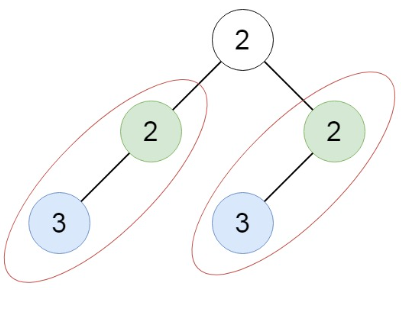
示例 2：



输入：root = [2,1,1]

输出：[[1]]

示例 3：



输入：root = [2,2,2,3,null,3,null]

输出：[[2,3],[3]]

提示：

树中的结点数在 [1, 5000] 范围内。

-200 <= Node.val <= 200

# 分析

## 方法一：使用序列化进行唯一表示

**思路：**

一种容易想到的方法是将每一棵子树都「序列化」成一个字符串，并且保证：

相同的子树会被序列化成相同的子串；

不同的子树会被序列化成不同的子串。

那么我们只要使用一个哈希表存储所有子树的序列化结果，如果某一个结果出现了超过一次，我们就发现了一类重复子树。

序列化一棵二叉树的方法可以参考「剑指 Offer 37. 序列化二叉树」的官方题解，这里也给出两种常用的序列化方法：

第一种方法是使用层序遍历的方法进行序列化，例如示例 1中的二叉树可以序列化为：

1,2,3,4,null,2,4,null,null,4

这也是力扣平台上测试代码时输入一棵二叉树的默认方法。

第二种方法是使用递归的方法进行序列化。我们可以将一棵以 xxx 为根节点值的子树序列化为：

x(左子树的序列化结果)(右子树的序列化结果)

左右子树分别递归地进行序列化。如果子树为空，那么序列化结果为空串。示例 1中的二叉树可以序列化为：

1(2(4()())())(3(2(4()())())(4()()))

下面的代码使用的是第二种方法。我们需要使用一个哈希映射seen存储序列到子树的映射。如果在计算序列时，通过seen查找到了已经出现过的序列，那么就可以把对应的子树放到哈希集合repeat中，这样就可以保证同一类的重复子树只会被存储在答案中一次。

**代码：**

class Solution {

private:

unordered\_map<string, TreeNode\*> seen; // 记录是否已经遍历过

unordered\_set<TreeNode\*> repeat; // 记录节点

public:

vector<TreeNode\*> findDuplicateSubtrees(TreeNode\* root) {

dfs(root);

return {repeat.begin(), repeat.end()};

}

string dfs(TreeNode \* node){

if (node == nullptr) return "";

// 获取当前节点的值以及左右节点的值组成的字符串

string serial = to\_string(node->val) + "(" + dfs(node->left) + ")(" + dfs(node->right) + ")";

auto it = seen.find(serial);

if (it != seen.end()) {

repeat.insert(it->second);

} else {

seen[serial] = node;

}

return serial;

}

};

**复杂度分析：**

时间复杂度：O(n^2)，其中n是树中节点的数目。在最坏情况下，树呈现链状的结构，而每一个节点都会在其左右子树序列的基础上再增加最多9个字符（两组括号以及节点本身的值），那么所有子树的序列长度之和为∑ni=09n=O(n2)。构造出这些序列就需要O(n^2)的时间。

空间复杂度：O(n^2)，即为哈希表需要使用的空间。