二叉树路径总和

2022年8月16日 9:35

给你二叉树的根节点 `root` 和一个表示目标和的整数 `targetSum` 。判断该树中是否存在 **根节点到叶子节点** 的路径，这条路径上所有节点值相加等于目标和 `targetSum` 。如果存在，返回 `true` ；否则，返回 `false` 。

叶子节点 是指没有子节点的节点。

输入： root = [5,4,8,11,null,13,4,7,2,null,null,null,1], targetSum = 22
输出： true
解释： 等于目标和的根节点到叶节点路径如上图所示。

1) 使用递归、dfs的方法进行解决

首先要做的是拆解问题，对于某叶子节点而言，要使路径和满足要求，即叶子节点前的路径之和需要满足targetSum-root->val（叶子节点的值）因此很清晰知道，这是满足递归需求的。

首先，递归中止条件为：

1. 当根节点为空节点时，**不存在根节点到叶子节点的路径**，返回false
2. 当该节点为叶子节点时，判断**val是否满足targetSum**

那么在每一次递归中，需要完成的任务即：

首先将targetSum进行处理（减去val），判断它的左孩子或右孩子是否满足目标和的要求。

```
class Solution {
public:
    bool hasPathSum(TreeNode* root, int targetSum) {
        if(!root)
            return false;
        if(!root->left&&!root->right){
            return root->val==targetSum;
        }
        return hasPathSum(root->left,targetSum-root->val)||hasPathSum(root->right,
targetSum-root->val);
    }
};
```

2) 使用bfs方法解决

由于bfs是一层层进行遍历，因此可以**另设一个队列用来储存根节点到当前节点的路径和**

只需要判断当前节点是否为叶子节点，若是，再判断到当前节点的路径和是否等于目标和targetSum即可

```cpp
class Solution {
public:
    bool hasPathSum(TreeNode* root, int targetSum) {
        if(!root)
            return false;
        queue<TreeNode* >q;
        queue<int> sum;
        q.push(root);
        sum.push(root->val);
        while(!q.empty()){
            TreeNode* now=q.front();
            q.pop();
            int temp=sum.front();
            sum.pop();
            if(!now->left&&!now->right&&temp==targetSum){
                return true;
            }
            if(now->left){
                q.push(now->left);
                sum.push(temp+now->left->val);
            }
            if(now->right){
                q.push(now->right);
                sum.push(temp+now->right->val);
            }
        }
        return false;
    }
};
```