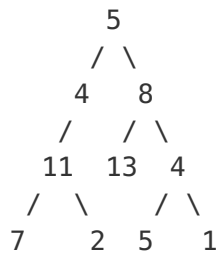


Path Sum II

Given a binary tree and a sum, find all root-to-leaf paths where each path's sum equals the given sum.

For example:

Given the below binary tree and `sum = 22`,



return

```
[  
  [5,4,11,2],  
  [5,8,4,5]  
]
```

Solution 1

```
public List<List<Integer>> pathSum(TreeNode root, int sum){
    List<List<Integer>> result = new LinkedList<List<Integer>>();
    List<Integer> currentResult = new LinkedList<Integer>();
    pathSum(root, sum, currentResult, result);
    return result;
}

public void pathSum(TreeNode root, int sum, List<Integer> currentResult,
    List<List<Integer>> result) {

    if (root == null)
        return;
    currentResult.add(new Integer(root.val));
    if (root.left == null && root.right == null && sum == root.val) {
        result.add(new LinkedList(currentResult));
        currentResult.remove(currentResult.size() - 1); //don't forget to remove the last integer
        return;
    } else {
        pathSum(root.left, sum - root.val, currentResult, result);
        pathSum(root.right, sum - root.val, currentResult, result);
    }
    currentResult.remove(currentResult.size() - 1);
}
```

written by [wdjoxda](#) original link [here](#)

Solution 2

Well, a typical backtracking problem. The code is as follows. You may walk through it using the example in the problem statement to see how it works.

```
class Solution {
public:
    vector<vector<int>> pathSum(TreeNode* root, int sum) {
        vector<vector<int> > paths;
        vector<int> path;
        findPaths(root, sum, path, paths);
        return paths;
    }
private:
    void findPaths(TreeNode* node, int sum, vector<int>& path, vector<vector<int>>
    & paths) {
        if (!node) return;
        path.push_back(node -> val);
        if (!(node -> left) && !(node -> right) && sum == node -> val)
            paths.push_back(path);
        findPaths(node -> left, sum - node -> val, path, paths);
        findPaths(node -> right, sum - node -> val, path, paths);
        path.pop_back();
    }
};
```

written by [jianchao.li.fighter](#) original link [here](#)

Solution 3

```
vector<vector<int> > pathSum(TreeNode *root, int sum) {
    vector<vector<int> > result;
    vector<int> cur_path(0);
    pathSumRec(root, sum, result, cur_path);
    return result;
}

// pass the current path as a reference and remember to pop out the last added
// element
// this improves the performance by 5 times
void pathSumRec(TreeNode* root, int sum, vector<vector<int> >& result, vector
<int>& cur_path) {
    if (root == NULL) {
        return;
    }

    if (root->val == sum && root->left == NULL && root->right == NULL) {
        cur_path.push_back(root->val);
        result.push_back(cur_path);
        cur_path.pop_back();
        return;
    }

    int sum_left = sum - root->val;
    cur_path.push_back(root->val);
    pathSumRec(root->left, sum_left, result, cur_path);
    //cur_path.pop_back();
    pathSumRec(root->right, sum_left, result, cur_path);
    cur_path.pop_back();
}
```

written by [pankit](#) original link [here](#)

From [LeetCoder](#).