# 题目

给定一个二叉树，原地将它展开为一个单链表。

例如，给定二叉树

1

/ \

2 5

/ \ \

3 4 6

将其展开为：

1

\

2

\

3

\

4

\

5

\

6

注：可以修改一下原题目，按照层次转换为链表，原理相同，只是采用层次遍历然后设置链表的next指针。

# 分析

## 方法一：前序遍历

**思路：**

先按照前序遍历一遍整个树，然后按照数组的下标重新排列链表。

**代码：**

/\*\*

 \* Definition for a binary tree node.

 \* struct TreeNode {

 \*     int val;

 \*     TreeNode \*left;

 \*     TreeNode \*right;

 \*     TreeNode() : val(0), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

 \* };

 \*/

class Solution {

public:

    void flatten(TreeNode\* root) {

        if(nullptr==root)

            return;

        vector<TreeNode \*> vec;

        preorderTraversal(root,vec);

        for(int i=1;i<vec.size();i++)

        {

            TreeNode \*prev = vec.at(i-1);

            TreeNode \*cur = vec.at(i);

            prev->left = nullptr;

            prev->right = cur;

        }

    }

    void preorderTraversal(TreeNode \*root,vector<TreeNode\*> &vec)

    {

        if(nullptr!=root)

        {

            vec.push\_back(root);

            preorderTraversal(root->left,vec);

            preorderTraversal(root->right,vec);

        }

    }

};

或（推荐）：

class Solution {public:

vector<TreeNode\*> vec;

void flatten(TreeNode\* root) {

if(nullptr == root) return;

preorderTraversal(root);

for(int i=1;i<vec.size();i++)

{

TreeNode \*prev = vec[i-1];

TreeNode \*cur = vec[i];

prev->left = nullptr;

prev->right = cur;

}

}

void preorderTraversal(TreeNode \*root)

{

stack<TreeNode\*> stk;

stk.push(root);

while(!stk.empty())

{

auto node = stk.top();

stk.pop();

vec.push\_back(node);

if(node->right) stk.push(node->right);

if(node->left) stk.push(node->left);

}

}

};

或（推荐）：

class Solution {

public:

    vector<TreeNode\*> vec;

    void flatten(TreeNode\* root) {

        if(nullptr == root) return;

        preorderTraversal(root);

        for(int i=1;i<vec.size();i++)

        {

            TreeNode \*prev = vec[i-1];

            TreeNode \*cur = vec[i];

            prev->left = nullptr;

            prev->right = cur;

        }

    }

    void preorderTraversal(TreeNode \*root)

    {

        stack<TreeNode\*> stk;

        TreeNode \*tmpNode = root;

        while(!stk.empty() || tmpNode)

        {

            while(tmpNode)

            {

                vec.push\_back(tmpNode);

                stk.push(tmpNode);

                tmpNode = tmpNode->left;

            }

            auto node = stk.top();

            stk.pop();

            tmpNode = node->right;

        }

    }

};

## 方法二：递归法

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

void flatten(TreeNode\* root) {

if (root == nullptr) return;

flatten(root->left);//一直遍历到最低层叶子节点

flatten(root->right);

if (root->left != nullptr) {

auto pre = root->left;

while (pre->right != nullptr) pre = pre->right;

pre->right = root->right;

root->right = root->left;

root->left = nullptr;

}

root = root->right;

return;

}

};