# 题目

给定一个二叉树，返回它的前序遍历。

**示例:**

**输入:** [1,null,2,3]

1

\

2

/

3

**输出:** [1,2,3]

**进阶:**递归算法很简单，你可以通过迭代算法完成吗？

# 分析

针对树的遍历，一般都是可以采用递归法和迭代法。

## 方法一：递归法

**思路：**

每次进入函数都要存储一下val，先父节点，然后左子树，接着右子树。

**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) {}

\* };

\*/

class Solution {

private:

vector<int> tmpArr;

//不要定义在方法体中，否则为临时变量每次都初始化

public:

vector<int> preorderTraversal(TreeNode\* root) {

if(nullptr != root)

{

//执行的操作（暂存到数组中，变式题此处设置终止条件）

tmpArr.push\_back(root->val);

preorderTraversal(root->left); //先左子树

preorderTraversal(root->right); //后右子树

}

return tmpArr;

}

};

## 方法二：迭代法（栈）

**思路：**利用**栈**实现**深度优先**遍历

非递归的方法：主要考虑什么时候输出；

关键点：

1、**前序入栈就输出，中序出栈再输出，后序需要访问第二次，出栈再输出**。

while(cur != NULL) {

stack.push(cur);

vec.push\_back(cur->val);

cur = cur->left;

}

2、cur = cur->right; 无需判断右子树空与否

**代码：**

class Solution {

public:

vector<int> preorderTraversal(TreeNode\* root) {

stack<TreeNode\*> stk; //中间临时变量

vector<int> ans; //返回值

TreeNode\* temp = root;

while(temp != NULL || !stk.empty()){

while(temp != NULL) {

ans.push\_back(temp -> val);

stk.push(temp); //处理左子树

temp = temp -> left;

}

//上述操作处理完所有的左子树

temp = stk.top(); //处理右子树

stk.pop();

temp = temp -> right;

}

return ans;

}

};