# 题目

给定一个N叉树，返回其节点值的 前序遍历 。

N叉树在输入中按层序遍历进行序列化表示，每组子节点由空值null分隔（请参见示例）。

进阶：

递归法很简单，你可以使用迭代法完成此题吗?

示例 1：



输入：root = [1,null,3,2,4,null,5,6]输出：[1,3,5,6,2,4]

示例 2：



输入：root = [1,null,2,3,4,5,null,null,6,7,null,8,null,9,10,null,null,11,null,12,null,13,null,null,14]

输出：[1,2,3,6,7,11,14,4,8,12,5,9,13,10]

提示：

N叉树的高度小于或等于1000

节点总数在范围[0,10^4]内

# 分析

## 方法一：递归法

class Solution {

public:

vector<int> res;

vector<int> preorder(Node\* root) {

if(!root) return res;

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

for(auto i : root -> children){

preorder(i);

}

return res;

}

};

上述方法采用成员变量res存储结果，并不是很好，这里采用局部变量：【**Leetcode解法**】

/\*

// Definition for a Node.

class Node {

public:

int val;

vector<Node\*> children;

Node() {}

Node(int \_val) {

val = \_val;

}

Node(int \_val, vector<Node\*> \_children) {

val = \_val;

children = \_children;

}

};

\*/

class Solution {

public:

vector<int> preorder(Node\* root) {

vector<int> ans;

dfs(root, ans);

return ans;

}

private:

void dfs(Node\* node, vector<int>& ans) {

if (!node) return;

// 1. 先访问当前节点

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

// 2. 再递归访问所有子节点

for (Node\* child : node->children) {

dfs(child, ans);

}

}

};

**复杂度分析：**

时间复杂度：O(n)

每个节点访问一次；

所有子节点都遍历一遍；

所以总共访问了 n 个节点 → O(n)

空间复杂度：O(n)

最坏情况是树退化成链（每个节点只有一个子节点），那么递归深度就是 n；

还需要一个 ans 数组来存储结果（O(n)）；

所以整体是：O(n)（栈 + 输出）

## 方法二：迭代法

class Solution {

public:

vector<int> preorder(Node\* root) {

vector<int> res;

if(!root) return res;

stack<Node\*> stk;

stk.push(root);

while(!stk.empty()){

Node\* tmp = stk.top();

stk.pop();

res.push\_back(tmp -> val);

for(int i = tmp -> children.size()-1; i >= 0; --i){

stk.push(tmp -> children[i]);

}

}

return res;

}

};

或：

class Solution {

public:

vector<int> preorder(Node\* root) {

if (!root) return {};

vector<int> ans;

stack<Node\*> stk;

stk.push(root);

while (!stk.empty()) {

Node\* node = stk.top(); stk.pop();

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

// 注意：栈是后进先出，要把子节点倒序压入

for (int i = node->children.size() - 1; i >= 0; --i) {

stk.push(node->children[i]);

}

}

return ans;

}

};

**复杂度分析：**

时间复杂度：O(n)

同样每个节点处理一次，push 和 pop 一次；

所以是 O(n)

空间复杂度：O(n)

栈中最多会保存整条分支（最坏情况下是线性链）；

输出 ans 占 O(n) 空间；

所以总空间为 O(n)