1. 二叉树的递归遍历模板：

前序遍历：

class Solution {

public:

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

        if (root == nullptr) {

            return {};

        }

        vector<int> ans;

        Dfs(root, ans);

        return ans;

    }

    void Dfs(TreeNode\* root, vector<int>& ans)

    {

        if (root == nullptr) {

            return;

        }

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

        Dfs(root->left, ans);

        Dfs(root->right, ans);

    }

};

中序遍历：

class Solution {

public:

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

        if (root == nullptr) {

            return {};

        }

        vector<int> ans;

        Dfs(root, ans);

        return ans;

    }

    void Dfs(TreeNode\* root, vector<int>& ans)

    {

        if (root == nullptr) {

            return;

        }

        Dfs(root->left, ans);

        ans.push\_back(root->val);

        Dfs(root->right, ans);

    }

};

后序遍历：

class Solution {

public:

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

        if (root == nullptr) {

            return {};

        }

        vector<int> ans;

        Dfs(root, ans);

        return ans;

    }

    void Dfs(TreeNode\* root, vector<int>& ans)

    {

        if (root == nullptr) {

            return;

        }

        Dfs(root->left, ans);

        Dfs(root->right, ans);

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

    }

};

1. N叉树的递归模板

前序遍历：

class Solution {

public:

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

vector<int> ans;

PreOrderDfs(root, ans);

return ans;

}

void PreOrderDfs(Node\* root, vector<int>& ans)

{

if (root == nullptr) {

return;

}

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

for (const auto& child : root->children) {

PreOrderDfs(child, ans);

}

}

};

后序遍历：

class Solution {

public:

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

        vector<int> ans;

        PostOrderDfs(root, ans);

        return ans;

    }

    void PostOrderDfs(Node\* root, vector<int>& ans)

    {

        if (root == nullptr) {

            return;

        }

        for (const auto& child : root->children) {

            PostOrderDfs(child, ans);

        }

        ans.push\_back(root->val);

    }

};