N叉树的遍历

树的遍历

一棵二叉树可以按照前序、中序、后序或者层序来进行遍历。在这些遍历方法中，前序遍历、后序遍历和层序遍历同样可以运用到N叉树中。

> 回顾 - 二叉树的遍历
>
> 1. 前序遍历 - 首先访问根节点，然后遍历左子树，最后遍历右子树；
> 2. 中序遍历 - 首先遍历左子树，然后访问根节点，最后遍历右子树；
> 3. 后序遍历 - 首先遍历左子树，然后遍历右子树，最后访问根节点；
> 4. 层序遍历 - 按照从左到右的顺序，逐层遍历各个节点。

请注意，N叉树的中序遍历没有标准定义，中序遍历只有在二叉树中有明确的定义。尽管我们可以通过几种不同的方法来定义N叉树的中序遍历，但是这些描述都不是特别贴切，并且在实践中也不常用到，所以我们暂且跳过N叉树中序遍历的部分。

把上述关于二叉树遍历转换为N叉树遍历，我们只需把如下表述:

> 遍历左子树... 遍历右子树...

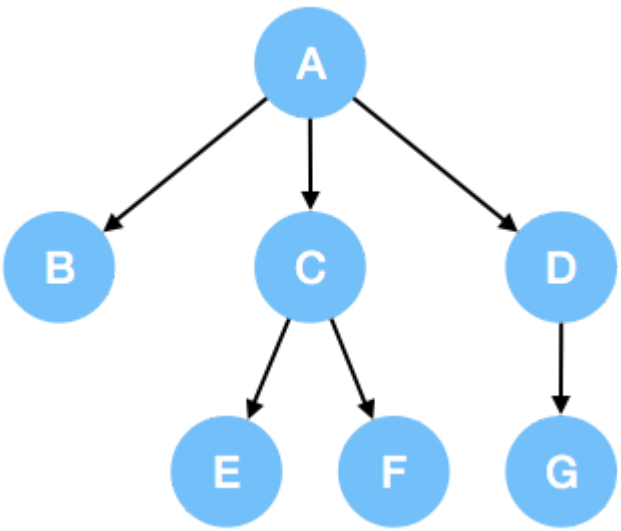
变为:

> 对于每个子节点: 通过递归地调用遍历函数来遍历以该子节点为根的子树

我们假设for循环将会按照各个节点在数据结构中的顺序进行遍历：通常按照从左到右的顺序，如下所示。

N叉树遍历示例

我们用如图所示的三叉树来举例说明:



1.前序遍历

在N叉树中，前序遍历指先访问根节点，然后逐个遍历以其子节点为根的子树。 例如，上述三叉树的前序遍历是: A->B->C->E->F->D->G.

2.后序遍历

在N叉树中，后序遍历指前先逐个遍历以根节点的子节点为根的子树，最后访问根节点。 例如，上述三叉树的后序遍历是: B->E->F->C->G->D->A.

3.层序遍历

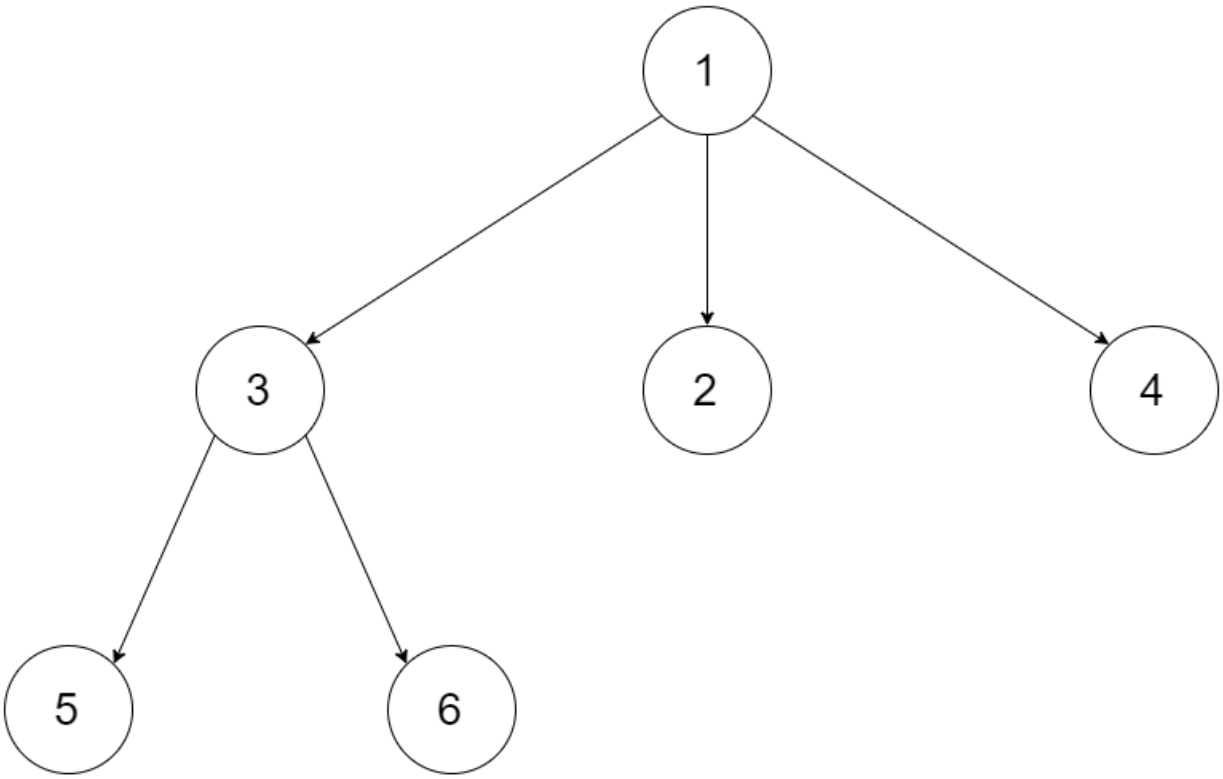
N叉树的层序遍历与二叉树的一致。通常，当我们在树中进行广度优先搜索时，我们将按层序的顺序进行遍历。 例如，上述三叉树的层序遍历是: A->B->C->D->E->F->G.

589.N叉树的前序遍历（简单）

1. 题目描述

   给定一个 N 叉树，返回其节点值的*前序遍历*。

   例如，给定一个 `3叉树` :



   返回其前序遍历: `[1,3,5,6,2,4]` 。

   说明: 递归法很简单，你可以使用迭代法完成此题吗?

2. 简单实现

```
/*
// Definition for a Node.
class Node {
public:
    int val;
```

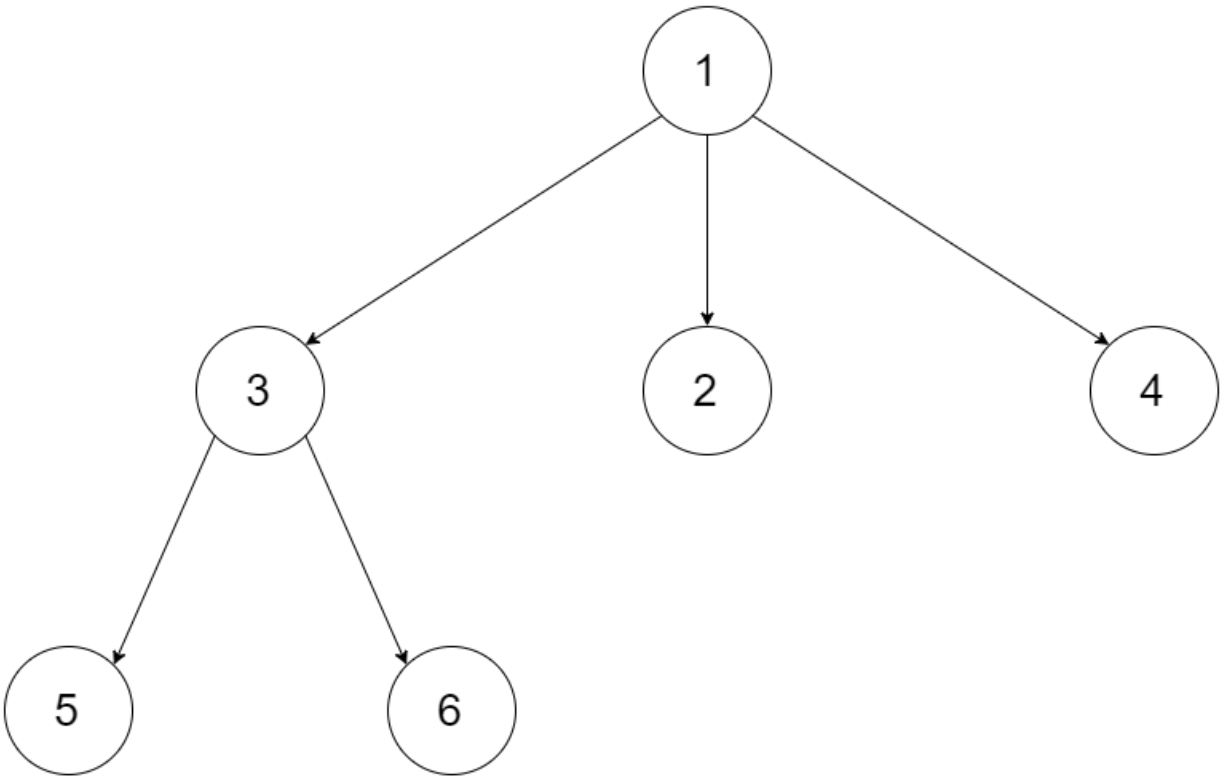
```
    vector<Node*> children;
    Node() {}
    Node(int _val) {
        val = _val;
    }
    Node(int _val, vector<Node*> _children) {
        val = _val;
        children = _children;
    }
};
*/
class Solution {
public:
    vector<int> preorder(Node* root) {
        vector<int> ans;
        if(!root) return ans;
        stack<Node*> s;
        s.push(root);
        while(!s.empty()){
            Node* cur = s.top();
            s.pop();
            ans.push_back(cur->val);
            for(int i = cur->children.size()-1; i >= 0; i--)
                s.push(cur->children[i]);
        }
        return ans;
    }
};
```

590.N叉树的后序遍历（简单）

1. 题目描述

   给定一个 N 叉树，返回其节点值的*后序遍历*。

   例如，给定一个 `3叉树` ：



返回其后序遍历: `[5,6,3,2,4,1]`.

说明: 递归法很简单，你可以使用迭代法完成此题吗?

2. 简单实现

增加pre指针用于判断是否遍历完所有孩子

```
class Solution {
public:
    vector<int> postorder(Node* root) {
        vector<int> ans;
        if(!root) return ans;
        stack<Node*> s;
        s.push(root);
        Node* pre = NULL;
        while(!s.empty()){
            Node* cur = s.top();
            if(cur->children.size() == 0 || pre == cur->children[cur-
>children.size()-1]){
                //叶节点 || 遍历完最后一个孩子
                s.pop();
                ans.push_back(cur->val);
                pre = cur;
            }
            else{//该节点的孩子们一个都没遍历过
                for(int i = cur->children.size()-1; i >= 0; i--)
                    s.push(cur->children[i]);
            }
        }
        return ans;
    }
```

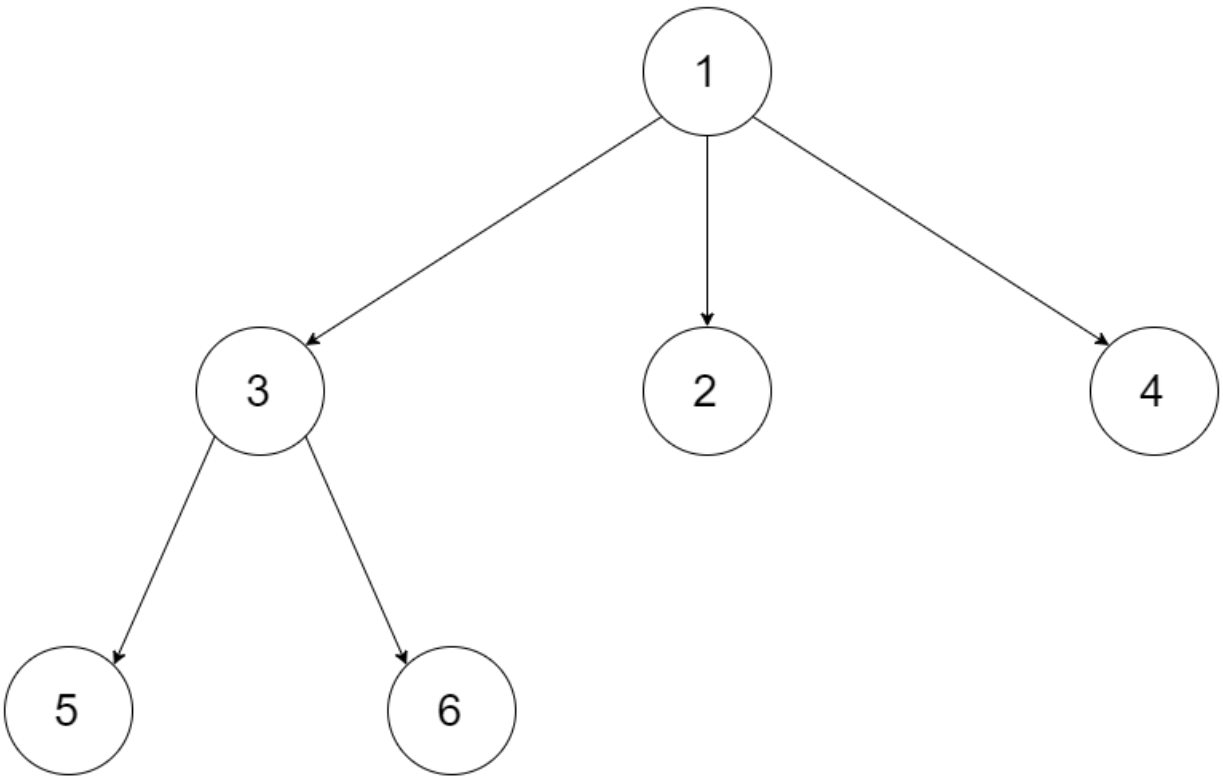
```
};
```

429. N叉树的层序遍历（中等）

1. 题目描述

   给定一个 N 叉树，返回其节点值的*层序遍历*。 (即从左到右，逐层遍历)。

   例如，给定一个 `3叉树` :



   返回其层序遍历:

```
[
     [1],
     [3,2,4],
     [5,6]
]
```

   说明:

   1. 树的深度不会超过 `1000` 。
   2. 树的节点总数不会超过 `5000` 。

2. 简单实现

```
class Solution {
public:
    vector<vector<int>> levelOrder(Node* root) {
        vector<vector<int>> ans;
        if(!root) return ans;
        queue<Node*> q;
```

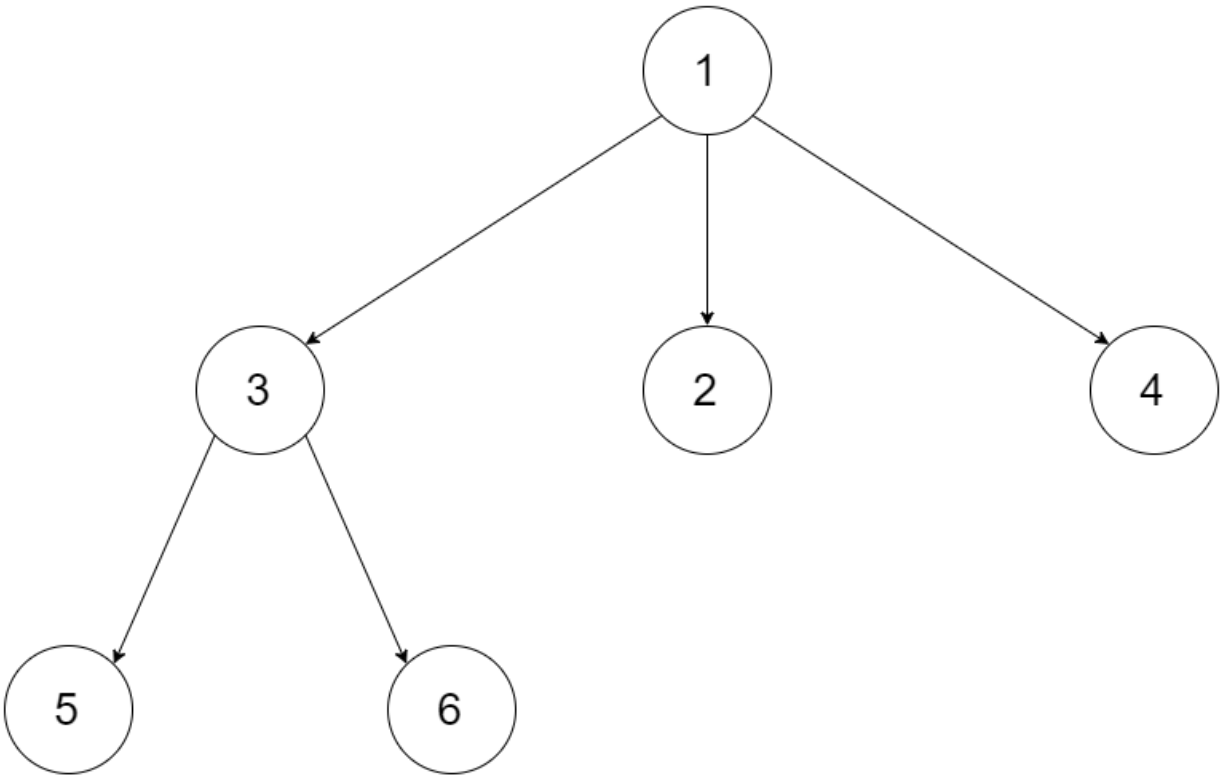
```
        q.push(root);
        while(!q.empty()){
            vector<int> temp;
            int size = q.size();
            for(int i = 0; i < size; i++){
                Node* cur = q.front();
                q.pop();
                temp.push_back(cur->val);
                for(int j = 0; j < cur->children.size(); j++){
                    q.push(cur->children[j]);
                }
            }
            ans.push_back(temp);
        }
        return ans;
    }
};
```

559.N叉树的最大深度（简单）

1. 题目描述

   给定一个 N 叉树，找到其最大深度。最大深度是指从根节点到最远叶子节点的最长路径上的节点总数。

   例如，给定一个 `3叉树` ：



   我们应返回其最大深度，3。

   说明:

   1. 树的深度不会超过 `1000` 。

2. 树的节点总不会超过 `5000`。

2. 简单实现

```
class Solution {
public:
    int maxDepth(Node* root) {
        if(!root) return 0;
        int len = root->children.size();
        int max = 0;
        for(int i = 0; i < len; i++){
            int cur = maxDepth(root->children[i]);
            if(cur > max)
                max = cur;
        }
        return max+1;
    }
};
```