（2）LC226.翻转二叉树

226. 翻转二叉树 - 力扣（LeetCode）

本质上是遍历每个节点，然后把每个结点的左右孩子翻转一下。

既然是遍历，可选择的就多了，pre-order，level-order，dfs等等均可以。

方法一：递归

代码：

```
class Solution{
public:
    void swap(TreeNode* &t1,TreeNode* &t2){
        TreeNode* tmp = t1;
        t1 = t2;
        t2 = tmp;
    }
    TreeNode* invertTree(TreeNode* root){
        if(root == NULL) return root;
        swap(root->left,root->right);
        invertTree(root->left);
        invertTree(root->right);
        return root;
    }
};
```

方法二：前序遍历迭代

思想：利用栈

代码：

```
class Solution {
public:
    TreeNode* invertTree(TreeNode* root) {
        if(root == NULL) return root;
        stack<TreeNode*> s;
        s.push(root);

        while(!s.empty()){
            TreeNode* tmp = s.top();
            s.pop();
            swap(tmp->left,tmp->right);
            if(tmp->left) s.push(tmp->left);
```

```
            if(tmp->right) s.push(tmp->right);
        }
        return root;
    }
};
```

方法三：level-order

思想：队列queue 以及层序遍历模板

代码：

```
class Solution {
public:
    TreeNode* invertTree(TreeNode* root) {
        if(root == NULL) return root;
        queue<TreeNode*> que;
        que.push(root);

        while(!que.empty()){
            int size = que.size();
            for(int i = 0;i < size;i++){
                TreeNode* tmp = que.front();
                que.pop();
                swap(tmp->left,tmp->right);
                if(tmp->left) que.push(tmp->left);
                if(tmp->right) que.push(tmp->right);
            }
        }
        return root;
    }
};
```