Morris遍历——二叉树遍历

2022年8月15日 9:13

二叉树的遍历均可采用Morris遍历方法

Leetcode 144、94、145

方法三：Morris 遍历

思路与算法

有一种巧妙的方法可以在线性时间内，只占用常数空间来实现前序遍历。这种方法由 J. H. Morris 在 1979 年的论文「Traversing Binary Trees Simply and Cheaply」中首次提出，因此被称为 Morris 遍历。

Morris 遍历的核心思想是利用树的大量空闲指针，实现空间开销的极限缩减。其前序遍历规则总结如下：

1. 新建临时节点，令该节点为 `root`；
2. 如果当前节点的左子节点为空，将当前节点加入答案，并遍历当前节点的右子节点；
3. 如果当前节点的左子节点不为空，在当前节点的左子树中找到当前节点在中序遍历下的前驱节点：
    - 如果前驱节点的右子节点为空，将前驱节点的右子节点设置为当前节点。然后将当前节点加入答案，并将前驱节点的右子节点更新为当前节点。当前节点更新为当前节点的左子节点。
    - 如果前驱节点的右子节点为当前节点，将它的右子节点重新设为空。当前节点更新为当前节点的右子节点。
4. 重复步骤 2 和步骤 3，直到遍历结束。

这样我们利用 Morris 遍历的方法，前序遍历该二叉树，即可实现线性时间与常数空间的遍历。

```cpp
class Solution {
public:
    vector<int> preorderTraversal(TreeNode *root) {
        vector<int> res;
        if (root == nullptr) {
            return res;
        }

        TreeNode *p1 = root, *p2 = nullptr;

        while (p1 != nullptr) {
            p2 = p1->left;
            if (p2 != nullptr) {
                while (p2->right != nullptr && p2->right != p1) {
                    p2 = p2->right;
                }
                if (p2->right == nullptr) {
                    res.emplace_back(p1->val);
                    p2->right = p1;
                    p1 = p1->left;
                    continue;
                } else {
                    p2->right = nullptr;
                }
            } else {
                res.emplace_back(p1->val);
            }
            p1 = p1->right;
        }
        return res;
    }
```