第八周作业

为了保证测试用例的充足，本次作业的代码我都在leetcode上进行提交。

前序递归

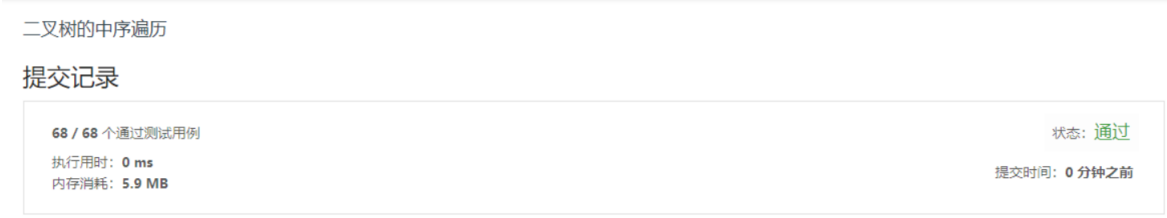
题目要求输入为按照数组形式的二叉树，输出为前序遍历数组，主要函数int* preorderTraversal(struct TreeNode* root, int* returnSize) 为固定格式，因此为了在层层迭代中形成数组，设置迭代函数 preorder，并引入参数arr和index。

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */


/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
#define MAX 1024

void preorder(struct TreeNode* root, int *arr, int* index){
    if(!root) return;
    arr[(*index)++] = root->val;
    preorder(root->left, arr, index);
    preorder(root->right, arr, index);
}

/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* preorderTraversal(struct TreeNode* root, int* returnSize){
    *returnSize = 0;
    int *arr = (int*)malloc(MAX * sizeof(int));
    preorder(root, arr, returnSize);
    return arr;
}
```

测试结果

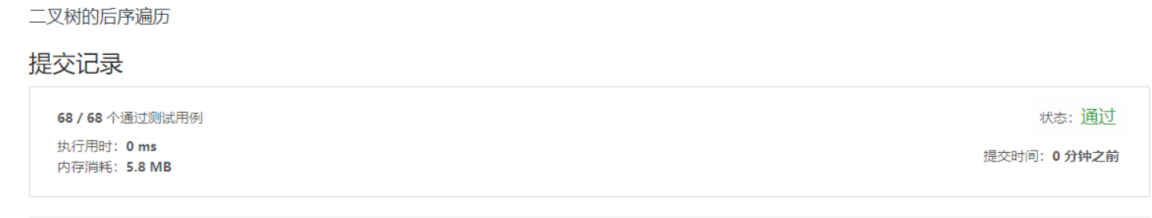
中序递归

中序的代码同理与前序递归，只是有两句代码更换顺序，此处只展示测试结果。



后序递归

后序递归同理，此处只展示测试结果。



前序迭代

完全按照老师的说法，复现了邓俊辉老师的两种想法。

注意不要忘记一开始的压栈！

1.指针法

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    vector<int> preorderTraversal(TreeNode* root) {
        std::stack<TreeNode*> stack;
```

```
        vector<int> vec;
        TreeNode* cur = root;
        stack.push(cur);
        while(!stack.empty()) {
        if(cur!=NULL){
            vec.push_back(cur->val);
            stack.push(cur->right);
            cur=cur->left;
        }
        else{
            cur=stack.top();
            stack.pop(); }
    }
        return vec;}
};
```

测试结果

2.朴素的第一种想法（就是左右子树都入栈那种）

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    vector<int> preorderTraversal(TreeNode* root) {
        std::stack<TreeNode*> stack;
        vector<int> vec;
        TreeNode* cur = root;
        stack.push(cur);
        while(cur!=NULL&&!stack.empty()) {
        cur=stack.top();
        stack.pop();
        vec.push_back(cur->val);
        if(cur->right!=NULL) stack.push(cur->right);
        if(cur->left!=NULL) stack.push(cur->left);
    }
        return vec;}
```

```
};
```

执行结果： 通过 显示详情 >

执行用时： 0 ms , 在所有 C++ 提交中击败了 100.00% 的用户

内存消耗： 8.5 MB , 在所有 C++ 提交中击败了 100.00% 的用户

炫耀一下：