094. Binary Tree Inorder Traversal

094 Binary Tree Inorder Traversal

- **Hash Table** + tree
- **Stack** + tree

Description

Given a binary tree, return the *inorder* traversal of its nodes' values.

For example:
Given binary tree `[1,null,2,3]`,

```
   1
    \
     2
    /
   3
```

return `[1,3,2]`.

1. Thought line

(1) Stack

inorder: left → root → right.

Stack 思路:

1. 对 ∀ node
   ↳ 找到最左点, 另将一路经过的点加入 toVisit
   ↳ 当最左点为空时, 取 toVisit.top(), 压入. 去看其右子树

⇒ Cur: starts from Root   rootJumped: starts from empty.

⇒ To each node: Cur

   (1) If Cur = nullptr:

   - Visit rootJumped.top()   // Go to Cur's parent node
   - Replace Cur = rootJumped.top()
   - Push Cur → val into Result
   - Go to Right Child to start over // Inorder: left child → node → right child

   (2) If Cur != NULL

   - Push Cur into rootJumped   // Cur is exist as a root node
   - Go to Cur' left child to start over

2. Stack+tree

```
/**
 * 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> inorderTraversal(TreeNode* root) {
        stack<TreeNode*> rootJumped;
        vector<int> result;
        TreeNode* cur = root;
        while(cur || !rootJumped.empty()){
            if (cur){
                rootJumped.push(cur);
                cur = cur->left;
            }
            else{
                cur = rootJumped.top();
                rootJumped.pop();
                result.push_back(cur->val);
                cur = cur->right;
```

```
            }
        }
        return result;
    }
};
```