```java
package leetcode.com.pickup1.hard;

import leetcode.com.util.TreeNode;

import java.util.Stack;

/**
 * Created by tclresearchamerica on 9/4/16.
 * ***************************************************************
 * Location:
 * https://leetcode.com/problems/serialize-and-deserialize-binary-tree/
 * ***************************************************************
 * Description:
 * Serialization is the process of converting a data structure or object into a
sequence of bits so that it can
 * be stored in a file or memory buffer, or transmitted across a network connection
link to be reconstructed later
 * in the same or another computer environment.
 * Design an algorithm to serialize and deserialize a binary tree. There is no
restriction on how your
 * serialization/deserialization algorithm should work. You just need to ensure that
a binary tree can be serialized
 * to a string and this string can be deserialized to the original tree structure.
 * For example, you may serialize the following tree
 * 1
 * / \
 * 2   3
 * / \
 * 4   5
 * as "[1,2,3,null,null,4,5]", just the same as how LeetCode OJ serializes a binary
tree. You do not necessarily need to
 * follow this format, so please be creative and come up with different approaches
yourself.
 * Note: Do not use class member/global/static variables to store states. Your
serialize and deserialize algorithms
 * should be stateless.
 * ***************************************************************
 * time: 50 mins
 * Beat: 50%
 * Bug: 2
 * ***************************************************************
 * Hindsight:
 * Binart Tree 的非递归解法,没有记牢,之前在递归解法上浪费了太多的时间啦
 */
public class No297 Serialize and Deserialize Binary Tree {}
class Codec {
    // Encodes a tree to a single string.
    public String serialize(TreeNode root) {
        //zero node
        if (root == null) return null;
```

```
        StringBuilder sb = new StringBuilder();
        helper(root, sb);
        //how to deal with fisrt ,
        sb.deleteCharAt(0);
        return sb.toString();
    }
    private void helper(TreeNode root, StringBuilder sb) {
        if (root == null) {
            sb.append(",null");
            return;
        }
        sb.append(",");
        sb.append(root.val);
        helper(root.left, sb);
        helper(root.right, sb);
    }
    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        if (data == null) return null;
        String[] nodes = data.split(",");
        //放弃递归调用改用while，可以解决一些不太方便的问题
        int index = 0;
        Stack<TreeNode> stack = new Stack<>();
        TreeNode root = new TreeNode(Integer.parseInt(nodes[index++]));
        stack.push(root);
        boolean leftFlg = true;
        while (index < nodes.length) {
            TreeNode node = stack.peek();
            if (leftFlg) {
                if ("null".equals(nodes[index])) {
                    node.left = null;
                    leftFlg = false;
                } else {
                    node.left = new TreeNode(Integer.parseInt(nodes[index]));
                    stack.push(node.left);
                }
            } else {
                stack.pop();
                if ("null".equals(nodes[index])) {
                    node.right = null;
                } else {
                    node.right = new TreeNode(Integer.parseInt(nodes[index]));
                    stack.push(node.right);
                    leftFlg = true;
                }
            }
            index++;
        }
        return root;
    }
}
```