144. Binary Tree Preorder Traversal

link

这个跟之前写的inOrder一样, 改变了存储的位置.

```
    public List<Integer> helper(TreeNode head){
        Stack<TreeNode> stack = new Stack();
        TreeNode current = head;
        List<Integer> res = new ArrayList();
        while(current != null || !stack.isEmpty()){
            while(current != null){
                stack.push(current);
                res.add(current.val);
                current = current.left;
            }
            current = stack.pop();
            current = current.right;
        }
        return res;
    }
```

这个只存右边的节点, 通过更改左边的节点实现

```
//Note that in this solution only right children are stored to stack.

public List<Integer> preorderTraversal(TreeNode node) {
    List<Integer> list = new LinkedList<Integer>();
    Stack<TreeNode> rights = new Stack<TreeNode>();
    while(node != null) {
        list.add(node.val);
        if (node.right != null) {
            rights.push(node.right);
        }
        node = node.left;
        if (node == null && !rights.isEmpty()) {
            node = rights.pop();
        }
    }
    return list;
}
```

更直接的方法

```
public List<Integer> preorderTraversal(TreeNode root) {
    List<Integer> result = new LinkedList<>();
    Deque<TreeNode> stack = new LinkedList<>();
    stack.push(root);
    while (!stack.isEmpty()) {
        TreeNode node = stack.pop();
        if (node != null) {
            result.add(node.val);
            stack.push(node.right);
            stack.push(node.left);
        }
    }
    return result;
}
```

递归

```
    public void helper1(TreeNode node, List res){
        if(node == null) return;
        res.add(node.val);
        helper1(node.left, res);
        helper1(node.right, res);
    }
```

Morris方法 O(1)遍历link

```
    public void morris(TreeNode node, List res){
        TreeNode rightNode = null;
        while(node != null){
            if(node.left != null){
                rightNode = node.left;
                while(rightNode.right != null && rightNode.right != node){
                    rightNode = rightNode.right;
                }
                if(rightNode.right == null){
                    rightNode.right = node;
                    res.add(node.val);
                    node = node.left;
                    continue;
                }else{
                    rightNode.right = null;
                    node = node.right;
                }
            }else{
                res.add(node.val);
                node = node.right;
            }
        }
```

```
}
```