题目描述

题目：遍历二叉树

解题思路

遍历二叉树分为前序遍历、中序遍历、后序遍历、层级遍历

代码实现

```
// 前序遍历
public List<Integer> preorderTraversal(TreeNode root) {
   List<Integer> list = new ArrayList<>();
   // 用来存放右节点的栈
   Deque<TreeNode> rights = new LinkedList<>();
   TreeNode cur = root;
   while (cur != null || !rights.isEmpty()) {
      list.add(cur.val);
      if (cur.right != null) rights.offerFirst(cur.right);
      cur = cur.left != null ? cur.left : rights.pollFirst();
   }
   return list;
}
// 中序遍历
public List<Integer> inorderTraversal(TreeNode root) {houxubianli

   List<Integer> list = new ArrayList<>();
   Deque<TreeNode> nodeStack = new LinkedList<>();
   TreeNode cur = root;
   while (cur != null || !nodeStack.isEmpty()) {

      while (cur != null) {
         nodeStack.offerFirst(cur);
         cur = cur.left;
      }

      cur = nodeStack.pollFirst();
      list.add(cur.val);
      cur = cur.right;
   }

   return list;
}
// 后序遍历
public List<Integer> postorderTraversal(TreeNode root) {
   LinkedList<Integer> list = new LinkedList<>();
   Deque<TreeNode> lefts = new LinkedList<>();
   TreeNode cur = root;
   while (cur != null || !lefts.isEmpty()) {
      list.addFirst(cur.val);
      if (cur.left != null) lefts.offerFirst(cur.left);
      cur = cur.right != null ? cur.right : lefts.pollFirst();
   }
   return list;
}
// 层级遍历
public List<List<Integer>> levelOrder(TreeNode root) {
   if(root == null) return new ArrayList<>();

   List<List<Integer>> resList = new ArrayList<>();
   Queue<TreeNode> queue = new LinkedList<>();
   TreeNode cur = root;
   queue.offer(cur);

   while(!queue.isEmpty()){
      int nodeNums = queue.size();
      List<Integer> subList = new ArrayList<>();
      for(int i=0;i<nodeNums;i++){
         cur = queue.poll();
         subList.add(cur.val);
         if(cur.left != null) queue.offer(cur.left);
         if(cur.right != null) queue.offer(cur.right);
      }
      resList.add(subList);
   }

   return resList;
}
```

题目描述

题目：反转链表

代码实现

```
public ListNode reverseList(ListNode head) {
    if(head==null) return head;
    ListNode newList = null;
    ListNode cur = head;
    while(cur != null){
        ListNode next = cur.next;
        cur.next = newList;
        newList = cur;
        cur = next;
    }

    return newList;
}
```