234 回文链表

```
Label：栈、双指针、链表
请判断一个链表是否为回文链表。
  输入: 1->2     输入: 1->2->2->1   输入: []
  输出: false    输出: true         输出: true

   输入: [1]     输入: [1,0,1]
   输出: true    输出: true
```

- 栈（用双向队列估计会更快）

```java
class Solution {
    public boolean isPalindrome(ListNode head) {
        Stack<Integer> stack = new Stack<>();

        if (head == null || head.next == null) {
            return true;
        }

        ListNode currNode = head;
        int count = 0;
        while (currNode != null) {
            stack.push(currNode.val);
            count++;
            currNode = currNode.next;
        }

        Stack<Integer> stack2 = new Stack<>();
        for (int i = 0; i < count/2; i++) {
            stack2.push(stack.pop());
        }

         if (count % 2 == 1) { // stack2 会多有一个
            stack.pop();
        }

        //对比
        while (!stack.empty()) {
            if (stack.pop().equals(stack2.pop())) {
                continue;
            } else {
                return false;
            }
        }
        return true;
    }
}
```

- 将值复制到数组中后用双指针法

```
class Solution {
    public boolean isPalindrome(ListNode head) {
       List<Integer> vals = new ArrayList<>();

        // 将list转换为数组
        ListNode currentNode = head;

        while (currentNode != null) {
            vals.add(currentNode.val);
            currentNode = currentNode.next;
        }

        int front = 0;
        int back = vals.size() - 1;

        while (front < back) {
            if (!vals.get(front).equals(vals.get(back))) {
                return false;
            }
            front++;
            back--;
        }
        return true;

    }
}
```

- 反转链表（避免使用额外的O（1）空间，但会改变结构）

```
class Solution {

    private ListNode findMid(ListNode head) { // 这种findmind的方法还是挺通用的
        ListNode fast = head, slow = head;
        while (fast != null && fast.next != null) {
            fast = fast.next.next;
            slow = slow.next;
        }
        return slow;
    }

    private ListNode reverse(ListNode head) {
        ListNode prev = null;
        ListNode cur = head;

        while (cur != null) {
            ListNode next = cur.next;
            cur.next = prev;
            prev = cur;
            cur = next;
        }
        return prev;
    }

    public boolean isPalindrome(ListNode head) {
        ListNode mid = findMid(head);
        ListNode newHead = reverse(mid); //将 mid 后面的反转
        while (newHead != null) {
            if (head.val != newHead.val) {
                return false;
            }
            head = head.next;
            newHead = newHead.next;
        }
        return true;
    }
}
```