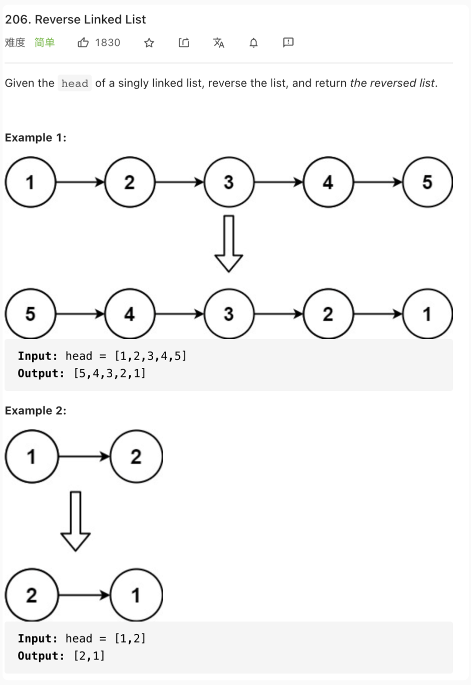
# 206. 反转链表

## 地址：反转链表

## 题目：

- English



206. Reverse Linked List

难度 简单 1830

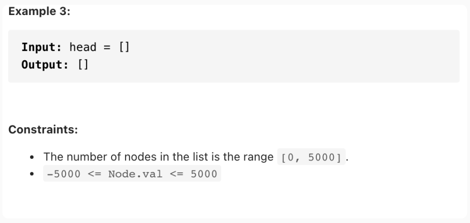
Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

**Example 1:**

```
Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]
```

**Example 2:**

```
Input: head = [1,2]
Output: [2,1]
```



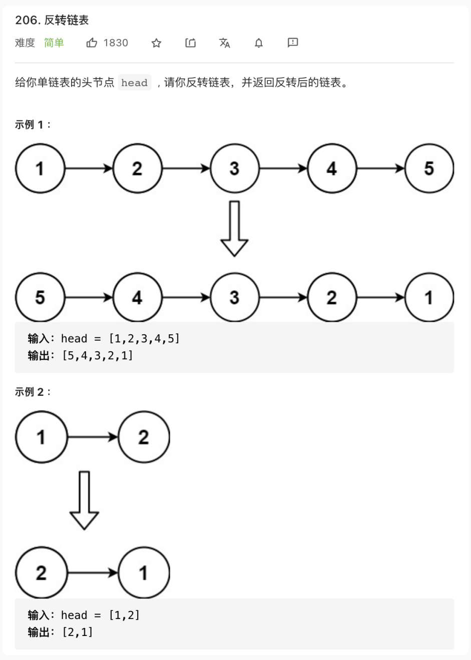
**Example 3:**

```
Input: head = []
Output: []
```

**Constraints:**

- The number of nodes in the list is the range `[0, 5000]`.
- `-5000 <= Node.val <= 5000`

- 中文



206. 反转链表

难度 简单 1830

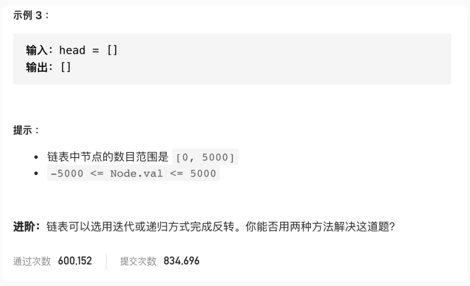
给你单链表的头节点 `head` ，请你反转链表，并返回反转后的链表。

**示例 1：**

```
输入：head = [1,2,3,4,5]
输出：[5,4,3,2,1]
```

**示例 2：**

```
输入：head = [1,2]
输出：[2,1]
```



**示例 3：**

```
输入：head = []
输出：[]
```

**提示：**

- 链表中节点的数目范围是 `[0, 5000]`
- `-5000 <= Node.val <= 5000`

**进阶：**链表可以选用迭代或递归方式完成反转。你能否用两种方法解决这道题？

通过次数 600,152 | 提交次数 834,696

## 思路1：双指针迭代法

### 分析

- ★ 双指针迭代赋值 :
    - ★ pre pointer - 迭代指针，用来迭代链表；
    - ★ cur pointer - 新建反转链表的头指针，倒叙存储当前节点。
- ★ 从链表开头迭代到尾部:
    - ★ 将当前节点连接到新建反转链表的头指针 cur pointer；
    - ★ 移动 cur pointer 作为 新建反转链表的头指针，倒叙存储当前节点。
- ★ 返回 反转链表 cur.

### 代码:

```java
// Java
// Time : 2021 - 07 - 12
public ListNode reverseList(ListNode head) {
    // head is empy list
    if (head == null) return head;


    ListNode cur = null;    // head pointer for new interated linked list
    ListNode pre = head;    // iterate the linked list


    while (pre != null) {   // iterate the linked list until list is end
      ListNode temp = pre;  // get the current node
      temp.next = cur;      // link the current node to new linked list
      cur = temp;           // move the cur pointer to the head of new linked list


      pre = pre.next;       // move pre pointer to next node
    }


    return cur;             // return the interate linked list
  }
```

```python
# Python
# Time : 2021 - 07 - 12

class Solution:
    def reverseList(self, head: ListNode) -> ListNode:

        if head == None:
            return head

        cur = None
        pre = head


        while pre:
            temp = pre
            pre = pre.next
            temp.next = cur
            cur = temp


        return cur
```

## 复杂度分析：

> - 时间复杂度：O(n)
> - 空间复杂度：O(1)