0206. 反转链表

ITCharge 大约 3 分钟

- 标签：递归、链表
- 难度：简单

题目链接

- 0206. 反转链表 - 力扣

题目大意

描述：给定一个单链表的头节点 `head` 。

要求：将该单链表进行反转。可以迭代或递归地反转链表。

说明：

- 链表中节点的数目范围是 $[0, 5000]$
- $-5000 \leq Node.val \leq 5000$。

示例：

- 示例 1：

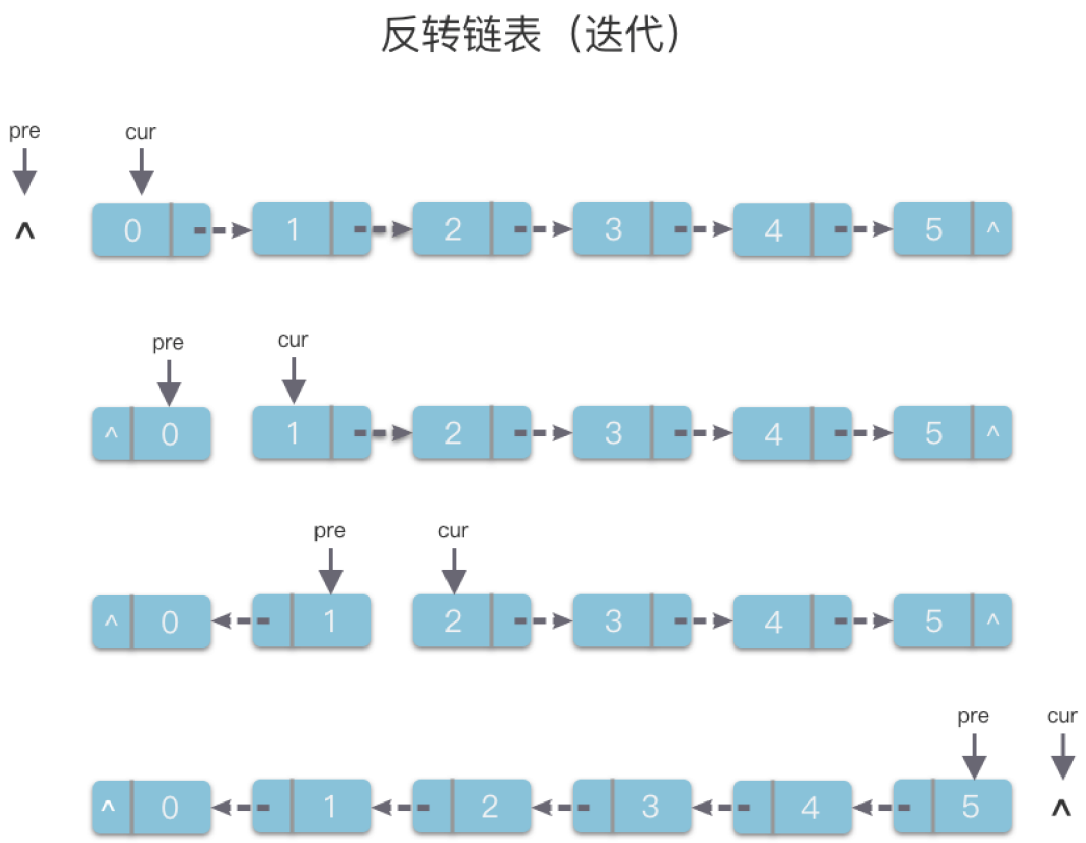
```py
输入：head = [1,2,3,4,5]
输出：[5,4,3,2,1]
解释：
翻转前    1->2->3->4->5->NULL
反转后    5->4->3->2->1->NULL
```

解题思路

思路 1：迭代

1. 使用两个指针 `cur` 和 `pre` 进行迭代。 `pre` 指向 `cur` 前一个节点位置。初始时， `pre` 指向 `None` ， `cur` 指向 `head` 。
2. 将 `pre` 和 `cur` 的前后指针进行交换，指针更替顺序为：
    1. 使用 `next` 指针保存当前节点 `cur` 的后一个节点，即 `next = cur.next` ；
    2. 断开当前节点 `cur` 的后一节点链接，将 `cur` 的 `next` 指针指向前一节点 `pre` ，即 `cur.next = pre` ；
    3. `pre` 向前移动一步，移动到 `cur` 位置，即 `pre = cur` ；
    4. `cur` 向前移动一步，移动到之前 `next` 指针保存的位置，即 `cur = next` 。
3. 继续执行第 2 步中的 1、2、3、4。
4. 最后等到 `cur` 遍历到链表末尾，即 `cur == None` ，时， `pre` 所在位置就是反转后链表的头节点，返回新的头节点 `pre` 。

使用迭代法反转链表的示意图如下所示：



思路 1：代码

```py
class Solution:
    def reverseList(self, head: ListNode) -> ListNode:
        pre = None
        cur = head
        while cur != None:
            next = cur.next
            cur.next = pre
            pre = cur
            cur = next
        return pre
```

思路 1：复杂度分析

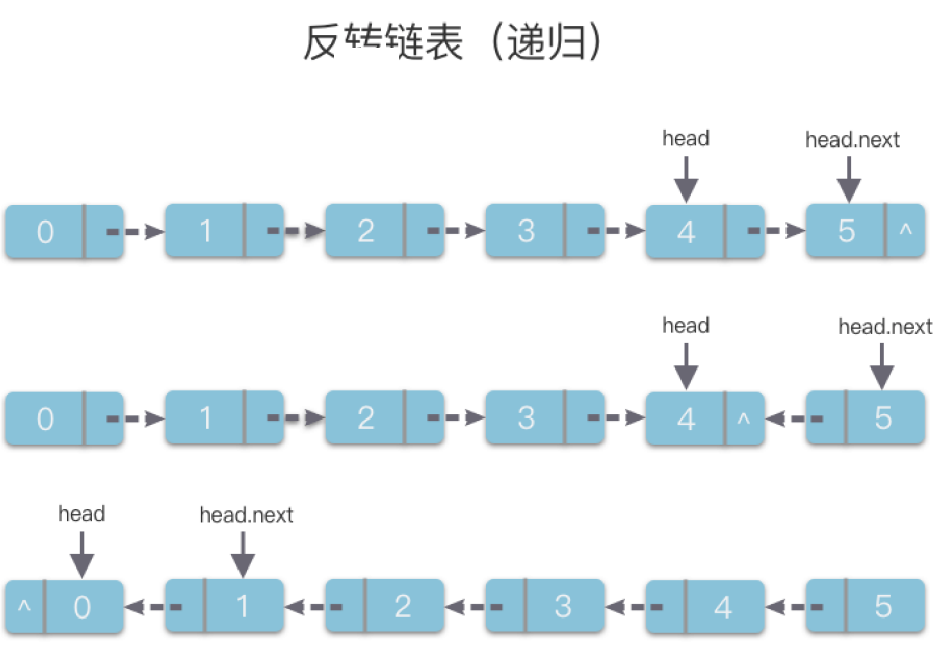
- **时间复杂度**：$O(n)$。
- **空间复杂度**：$O(1)$。

思路 2：递归

具体做法如下：

1. 首先定义递归函数含义为：将链表反转，并返回反转后的头节点。
2. 然后从 `head.next` 的位置开始调用递归函数，即将 `head.next` 为头节点的链表进行反转，并返回该链表的头节点。
3. 递归到链表的最后一个节点，将其作为最终的头节点，即为 `new_head` 。
4. 在每次递归函数返回的过程中，改变 `head` 和 `head.next` 的指向关系。也就是将 `head.next` 的 `next` 指针先指向当前节点 `head` ，即 `head.next.next = head` 。
5. 然后让当前节点 `head` 的 `next` 指针指向 `None` ，从而实现从链表尾部开始的局部反转。
6. 当递归从末尾开始顺着递归栈的退出，从而将整个链表进行反转。
7. 最后返回反转后的链表头节点 `new_head` 。

使用递归法反转链表的示意图如下所示：



思路 2：代码

```py
class Solution:
    def reverseList(self, head: ListNode) -> ListNode:
        if head == None or head.next == None:
            return head
        new_head = self.reverseList(head.next)
        head.next.next = head
        head.next = None
        return new_head
```

思路 2：复杂度分析

- **时间复杂度**：$O(n)$
- **空间复杂度**：$O(n)$。最多需要 n 层栈空间。

参考资料

- 【题解】反转链表 - 反转链表 - 力扣
- 【题解】【反转链表】：双指针，递归，妖魔化的双指针 - 反转链表 - 力扣（LeetCode）