0203. 移除链表元素

ITCharge 大约 1 分钟

- 标签：递归、链表
- 难度：简单

题目链接

- 0203. 移除链表元素 - 力扣

题目大意

描述：给定一个链表的头节点 `head` 和一个值 `val` 。

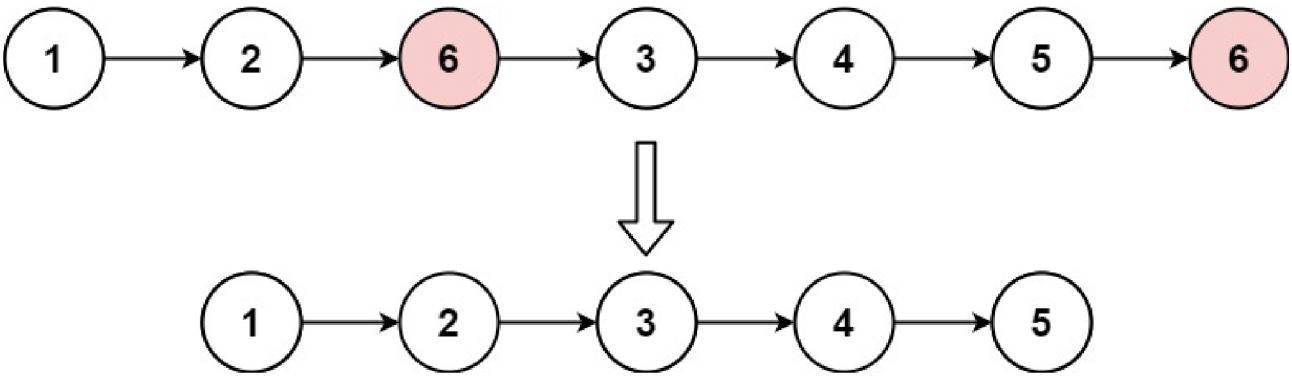
要求：删除链表中值为 `val` 的节点，并返回新的链表头节点。

说明：

- 列表中的节点数目在范围 $[0, 10^4]$ 内
- $1 \leq Node.val \leq 50$。
- $0 \leq val \leq 50$。

示例：

- 示例 1：




```py
输入：head = [1,2,6,3,4,5,6], val = 6
输出：[1,2,3,4,5]
```

- 示例 2：

```py
输入：head = [], val = 1
输出：[]
```

解题思路

思路 1：迭代

- 使用两个指针 `prev` 和 `curr` 。 `prev` 指向前一节点和当前节点， `curr` 指向当前节点。
- 从前向后遍历链表，遇到值为 `val` 的节点时，将 `prev` 的 `next` 指针指向当前节点的下一个节点，继续递归遍历。没有遇到则将 `prev` 指针向后移动一步。
- 向右移动 `curr` ，继续遍历。

需要注意的是：因为要删除的节点可能包含了头节点，我们可以考虑在遍历之前，新建一个头节点，让其指向原来的头节点。这样，最终如果删除的是头节点，则直接删除原头节点，然后最后返回新建头节点的下一个节点即可。

思路 1：代码

```py
class Solution:
    def removeElements(self, head: ListNode, val: int) -> ListNode:
        newHead = ListNode(0, head)
        newHead.next = head

        prev, curr = newHead, head
        while curr:
            if curr.val == val:
                prev.next = curr.next
            else:
                prev = curr
            curr = curr.next
        return newHead.next
```

思路 1：复杂度分析

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