youngyangyang04 / leetcode-master

Code Issues 6 Pull requests 9 Actions Projects Wiki Security

master

leetcode-master / problems / 0019.删除链表的倒数第N个节点.md

morningsky 添加 0019.删除链表的倒数第N个节点 python3版本 History

5 contributors

192 lines (156 sloc) 6.23 KB

PDF下载 代码随想录 刷题 微信群 B站 代码随想录 知识星球 代码随想录

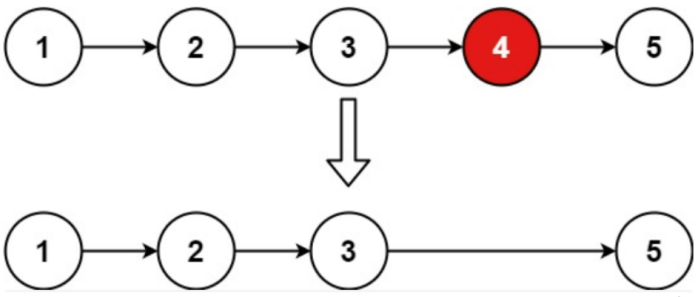
欢迎大家参与本项目，贡献其他语言版本的代码，拥抱开源，让更多学习算法的小伙伴们收益！

19.删除链表的倒数第N个节点

题目链接：https://leetcode-cn.com/problems/remove-nth-node-from-end-of-list/

给你一个链表，删除链表的倒数第 n 个结点，并且返回链表的头结点。

进阶：你能尝试使用一趟扫描实现吗?



示例 1：

19.删除链表的倒数第N个节点

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

输入：head = [1], n = 1 输出：[] 示例 3：

输入：head = [1,2], n = 1 输出：[1]

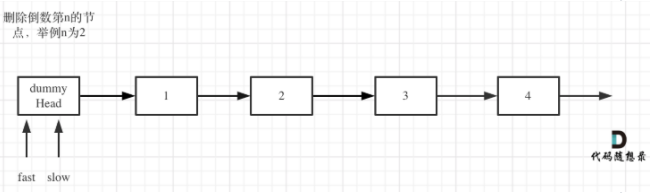
思路

双指针的经典应用，如果要删除倒数第n个节点，让fast移动n步，然后让fast和slow同时移动，直到fast指向链表末尾。删掉slow所指向的节点就可以了。

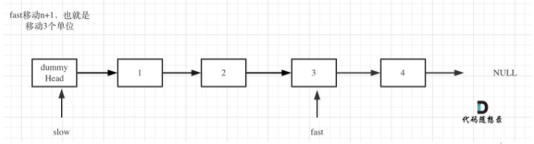
思路是这样的，但要注意一些细节。

分为如下几步：

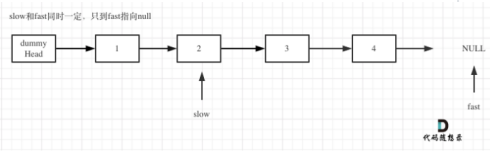
- 首先这里我推荐大家使用虚拟头结点，这样方面处理删除实际头结点的逻辑，如果虚拟头结点不清楚，可以看这篇： 链表：听说用虚拟头节点会方便很多？
- 定义fast指针和slow指针，初始值为虚拟头结点，如图：



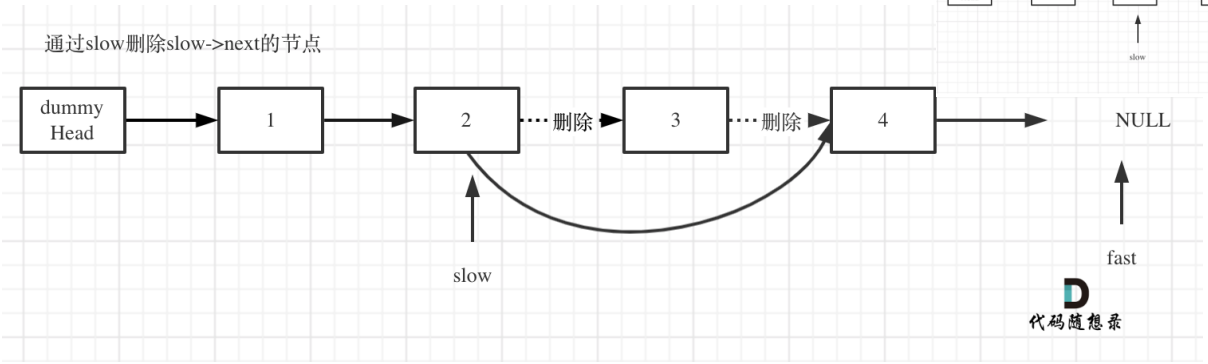
- fast首先走n + 1步 ，为什么是n+1呢，因为只有这样同时移动的时候slow才能指向删除节点的上一个节点（方便做删除操作），如图：



- fast和slow同时移动，之道fast指向末尾，如题：



- 删除slow指向的下一个节点，如图：



此时不难写出如下C++代码：

```
class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        ListNode* dummyHead = new ListNode(0);
        dummyHead->next = head;
        ListNode* slow = dummyHead;
        ListNode* fast = dummyHead;
        while(n-- && fast != NULL) { while(n-- && fast->next!=nullptr)
            fast = fast->next;
        }
        fast = fast->next; // fast再提前走一步，因为需要让slow指向删除节点的上一个节点
        while (fast != NULL) {    while(n-- && fast->next!=nullptr)
            fast = fast->next;
            slow = slow->next;
        }
        slow->next = slow->next->next;
        return dummyHead->next;
    }
};
```

```
ListNode* tmp = slow->next;
slow->next = slow->next->next;
delete tmp;
head = dummyHead->next;
delete dummyHead;
return head;
```

其他语言版本

java:

```
class Solution {
    public ListNode removeNthFromEnd(ListNode head, int n) {
        ListNode dummy = new ListNode(-1);
        dummy.next = head;
```

```
        ListNode slow = dummy;
        ListNode fast = dummy;
        while (n-- > 0) {
            fast = fast.next;
        }
        // 记住 待删除节点slow 的上一节点
        ListNode prev = null;
        while (fast != null) {
            prev = slow;
            slow = slow.next;
            fast = fast.next;
        }
        // 上一节点的next指针绕过 待删除节点slow 直接指向slow的下一节点
        prev.next = slow.next;
        // 释放 待删除节点slow 的next指针, 这句删掉也能AC
        slow.next = null;

        return dummy.next;
    }
}
```

Python:

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def removeNthFromEnd(self, head: ListNode, n: int) -> ListNode:
        head_dummy = ListNode()
        head_dummy.next = head

        slow, fast = head_dummy, head_dummy
        while(n!=0): #fast先往前走n步
            fast = fast.next
            n -= 1
        while(fast.next!=None):
            slow = slow.next
            fast = fast.next
        #fast 走到结尾后，slow的下一个节点为倒数第N个节点
        slow.next = slow.next.next #删除
        return head_dummy.next
```

Go:

```
/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *     Val int
 *     Next *ListNode
 * }
 */
func removeNthFromEnd(head *ListNode, n int) *ListNode {
```

```
    dummyHead := &ListNode{}
    dummyHead.Next = head
    cur := head
    prev := dummyHead
    i := 1
    for cur != nil {
        cur = cur.Next
        if i > n {
            prev = prev.Next
        }
        i++
    }
    prev.Next = prev.Next.Next
    return dummyHead.Next
}
```

JavaScript:

```
/**
 * @param {ListNode} head
 * @param {number} n
 * @return {ListNode}
 */
var removeNthFromEnd = function(head, n) {
    let ret = new ListNode(0, head),
        slow = fast = ret;
    while(n--) fast = fast.next;
    if(!fast) return ret.next;
    while (fast.next) {
        fast = fast.next;
        slow = slow.next
    };
    slow.next = slow.next.next;
    return ret.next;
};
```

- 作者微信：程序员Carl
- B站视频：代码随想录
- 知识星球：代码随想录