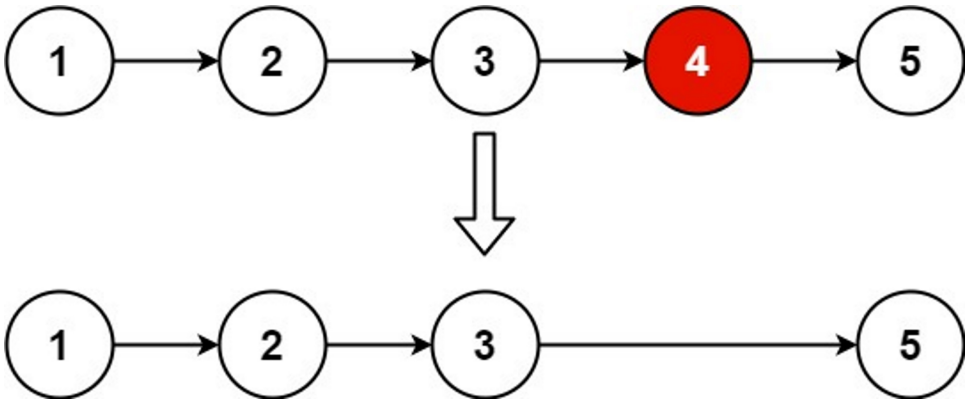
19. 删除链表的倒数第 N 个结点+面试题 02.07. 链表相交

Wednesday, July 13, 2022 9:29 AM

https://leetcode.cn/problems/remove-nth-node-from-end-of-list/

中等

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。



示例 1：

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：[]

解法：快慢指针

先设空节点res，res.next=head，不这样最后只能返回head，当head被删时，返回结果就不对了。

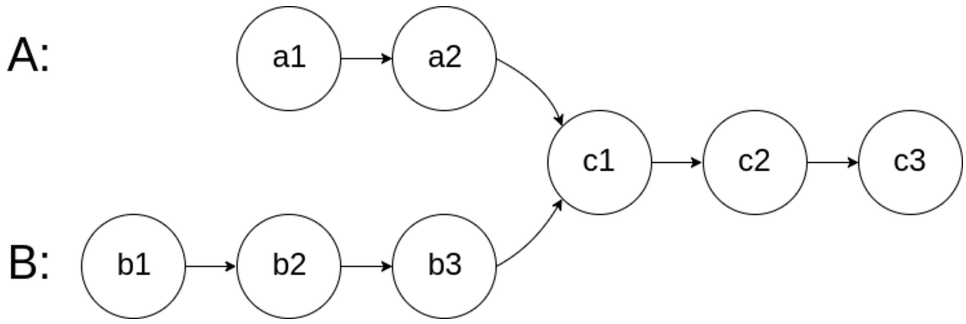
fast、slow均指向res，fast先走n步，然后再一起走，当fast指到null时，slow指向的就是要删除的节点。

```
class Solution {
    public ListNode removeNthFromEnd(ListNode head, int n) {
        if(head==null) return null;
        ListNode res=new ListNode(0);
        res.next=head;
        ListNode fast=res;
        ListNode slow=res;
        while(n>0){ //fast先走n步
            fast=fast.next;
            n--;
            if(n>0&&fast==null) //fast还没走完n步就到null了，说明n大于链表长度，直接返回。
                return head;
        }
        ListNode temp=null; //保存slow指针前一位
        while(fast!=null){ //同时走，直到fast遍历完。
            temp=slow;
```

```
            fast=fast.next;
            slow=slow.next;
        }
        temp.next=slow.next; //删除目标节点
        return res.next;
    }
}
```

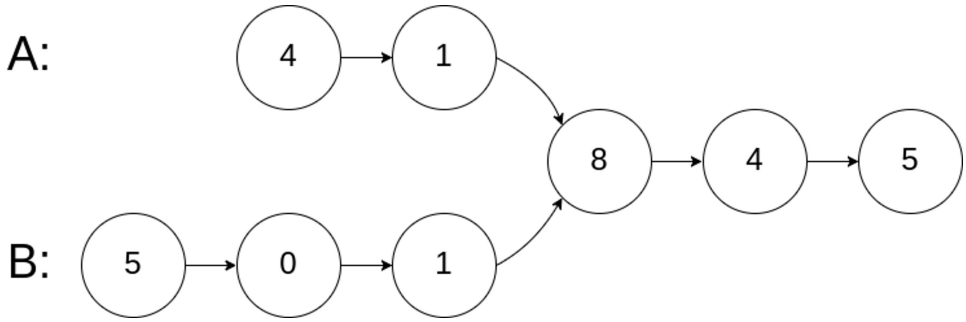
面试题 02.07. 链表相交
https://leetcode.cn/problems/intersection-of-two-linked-lists-lcci/
简单
给你两个单链表的头节点 headA 和 headB ， 请你找出并返回两个单链表相交的起始节点。如果两个链表没有交点，返回 null 。

图示两个链表在节点 c1 开始相交：



题目数据 保证 整个链式结构中不存在环。
注意，函数返回结果后，链表必须 保持其原始结构 。

示例 1：



输入：intersectVal = 8, listA = [4,1,8,4,5], listB = [5,0,1,8,4,5], skipA = 2, skipB = 3
输出：Intersected at '8'
解释：相交节点的值为 8 （注意，如果两个链表相交则不能为 0）。
从各自的表头开始算起，链表 A 为 [4,1,8,4,5]，链表 B 为 [5,0,1,8,4,5]。
在 A 中，相交节点前有 2 个节点；在 B 中，相交节点前有 3 个节点。

思路： 屁股对齐。谁长，谁先走长出来的步数，然后一起走，边走边比较。

```
public class Solution {
    public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
```

```
        if(headA==null || headB==null) return null;
        ListNode a=headA;
        ListNode b=headB;
        int countA=0;
        int countB=0;
        while(a!=null){ // 遍历保存A\B的长度
            a=a.next;
            countA++;
        }
        while(b!=null){
            b=b.next;
            countB++;
        }
        int count=countA-countB; // AB长度之差
        a=headA;
        b=headB;
        while(count>0){ // 以下的两个while只能进去一个，长度相等即count=0时一个都进不
去
            a=a.next; // 长的先走长出来的部分
            count--;
        }
        while(count<0){
            b=b.next;
            count++;
        }
        while(a!=null){ //此时，两个一样长了，while判断一个就够了。一起走
            if(a==b) return a; // 相等返回该节点。
            a=a.next;
            b=b.next;
        }
        return null; // 没找到，返回null
    }
}
```