题目：

Given a linked list, remove the *n*th node from the end of list and return its head.

For example,

Given linked list: **1->2->3->4->5**, and ***n* = 2**.

After removing the second node from the end, the linked list becomes **1->2->3->5**.

**Note:**  
Given *n* will always be valid.  
Try to do this in one pass.

问题描述就是删除单向链表的倒数第n个链单元。

方法一：

直接删，但是需要先遍历一遍链表记录链表的长度length。然后锁定倒数第n个链节的位置为length-n+1。但是链表要删除的时候一定是从要删除的位置的前一个链节删（p.next=p.next.next），所以最终锁定的链节是length-n。

方法先需要遍历一遍链表，然后再次遍历到length-n。所以方法的时间复杂度比较大O(n\*n)

基本代码如下：

public static ListNode removeNthFromEnd(ListNode head, int n) {

int count=-1;

int i=0;

ListNode p =head;

while(p!=null){ //遍历计算链表总长度

p=p.next;

count+=1;

}

p=head;

while(p!=null){ //第二次遍历

if(i==count-n){ //锁定删除节点的上一个位置

p.next=p.next.next; //删除

}

i+=1;

p=p.next;

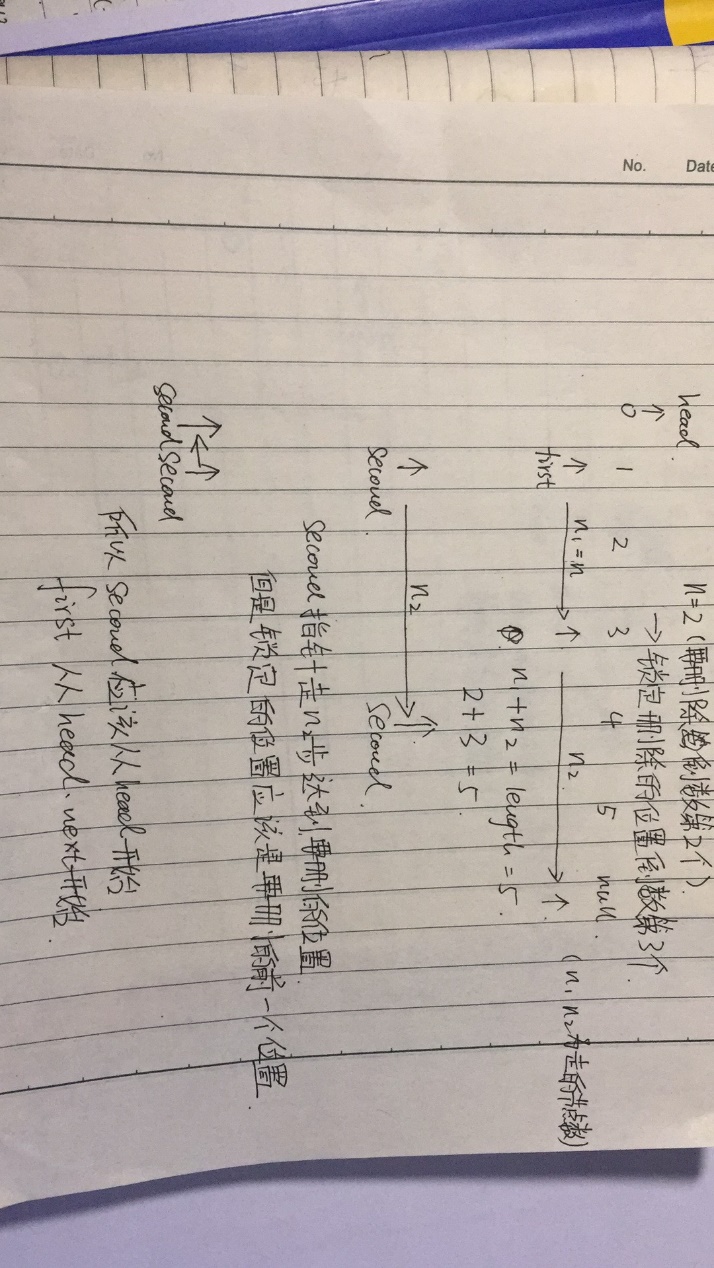
}

return head;

}

方法二：

建立两个指针，第一个指针确定删除的位置，第二个指针删除。



步骤如下：

（1）first指针先走n步

（2）以first！=null条件first和second一起走

（3）second走到的位置正好是要删除的位置的前一位，删除。

second.next=second.next.next;

具体代码如下：

public static ListNode removeNthFromEnd2(ListNode head, int n) {

ListNode first = head.next;

ListNode second = head;

for(int i=0;i<n;i++){

first=first.next;

}

while (first!=null){

first=first.next;

second=second.next;

}

second.next=second.next.next;

return head;

}