**Remove Nth Node From End of List**

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

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

简单题， 使用快慢指针解决此题，需要注意最后删除的是否为头节点。让快指针先走n步，直至快指针走到终点，找到需要删除节点之前的一个节点，改变node->next域即可。

class Solution {

public:

ListNode\* removeNthFromEnd(ListNode\* head, int n) {

if (!head->next) return NULL;

ListNode \*pre = head, \*cur = head;

for (int i = 0; i < n; ++i) cur = cur->next;

if (!cur) return head->next;

while (cur->next) {

cur = cur->next;

pre = pre->next;

}

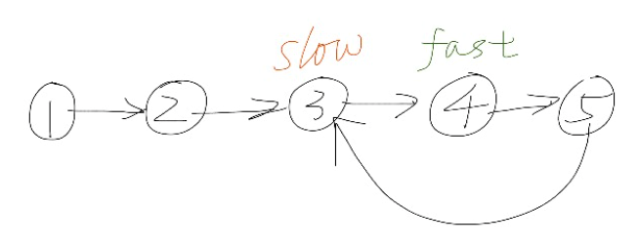
pre->next = pre->next->next;

return head;

}

};

链表环：快慢指针



public class Solution {

public boolean hasCycle(ListNode head) {

if (head == null || head.next == null) {

return false;

}

ListNode slow = head;

ListNode fast = head;

while (fast.next != null && fast.next.next != null) {

slow = slow.next;

fast = fast.next.next;

if (slow == fast) {

return true;

}

}

return false;

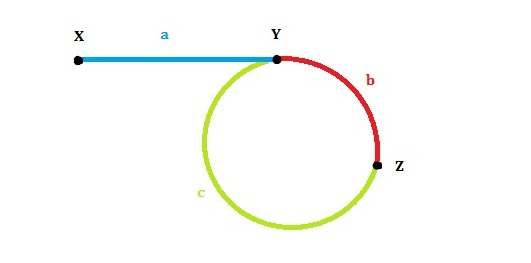
}

}

**扩展问题https://www.cnblogs.com/hiddenfox/p/3408931.html**

在网上搜集了一下这个问题相关的一些问题，思路开阔了不少，总结如下：

1. 环的长度是多少？



第一次相遇时slow走过的距离：a+b，fast走过的距离：a+b+c+b。

**因为fast的速度是slow的两倍，所以fast走的距离是slow的两倍，有 2(a+b) = a+b+c+b，可以得到a=c（这个结论很重要！）。**

也就是说，从一开始到二者第一次相遇，慢指针的步数就等于环的长度。

1. 如何找到环中第一个节点（即Linked List Cycle II）？

我们已经得到了结论a=c，那么让两个指针分别从X和Z开始走，每次走一步，那么正好会在Y相遇！也就是环的第一个节点。

1. 如何将有环的链表变成单链表（解除环）？

将c段中Y点之前的那个节点与Y的链接切断即可。

4. 如何判断两个单链表是否有交点？如何找到第一个相交的节点？

首先我们看下面这张图：

**判断两个单链表是否相交并求出相交的第一结点**

将一个链表首尾相接，然后判断另外一个链表是否有环，如果有环，则两个链表相交。那么求第一个交点则求出有环的的那个链表的环结点即是。