Given a linked list, return the node where the cycle begins. If there is no cycle, return null.

Note: Do not modify the linked list.

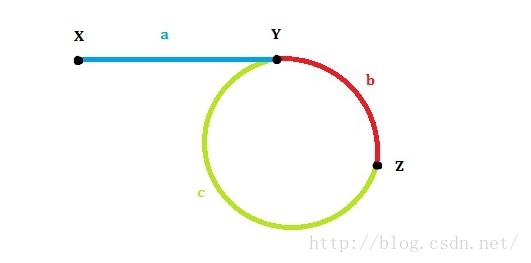
<http://blog.csdn.net/xy010902100449/article/details/48981185>

【分析】本文是找到链表环的第一个节点。

一开始使用了复杂度O(n^2)的方法，使用两个指针a, b。a从表头开始一步一步往前走，遇到null则说明没有环，返回false；a每走一步，b从头开始走，如果遇到b==a.next，则说明有环true，如果遇到b==a，则说明暂时没有环，继续循环。  
  
后来找到了复杂度O(n)的方法，使用两个指针slow,fast。两个指针都从表头开始走，slow每次走一步，fast每次走两步，如果fast遇到null，则说明没有环，返回false；如果slow==fast，说明有环，并且此时fast超了slow一圈，返回true。  
  
**为什么有环的情况下二者一定会相遇呢？因为fast先进入环，在slow进入之后，如果把slow看作在前面，fast在后面每次循环都向slow靠近1，所以一定会相遇，而不会出现fast直接跳过slslow的情况。**  
  
扩展问题  
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？
2. 如何找到环中第一个节点（即Linked List Cycle II）？
3. 如何将有环的链表变成单链表（解除环）？
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：



设：链表头是X，环的第一个节点是Y，slow和fast第一次的交点是Z。各段的长度分别是a,b,c，如图所示。环的长度是L。slow和fast的速度分别是qs,qf。  
下面我们来挨个问题分析。  
 方法一（网上都是这个答案）：  
第一次相遇后，让slow,fast继续走，记录到下次相遇时循环了几次。因为当fast第二次到达Z点时，fast走了一圈，slow走了半圈，而当fast第三次到达Z点时，fast走了两圈，slow走了一圈，正好还在Z点相遇。  
方法二：  
第一次相遇后，让fast停着不走了，slow继续走，记录到下次相遇时循环了几次。  
**方法三（最简单）：**  
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！）。**  
  
我们发现L=b+c=a+b，也就是说，

1. 从一开始到二者第一次相遇，循环的次数就等于环的长度。
2. 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。
3. 在上一个问题的最后，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？先判断两个链表是否有环，如果一个有环一个没环，肯定不相交；如果两个都没有环，判断两个列表的尾部是否相等；如果两个都有环，判断一个链表上的Z点是否在另一个链表上。

代码如下

**[cpp]** [view plain](https://blog.csdn.net/xy010902100449/article/details/48995255) [copy](https://blog.csdn.net/xy010902100449/article/details/48995255)

1. /\*\*
2. \* Definition for singly-linked list.
3. \*\*/
4. **struct** ListNode {
5. **int** val;
6. **struct** ListNode \*next;
7. };
9. **struct** ListNode \*detectCycle(**struct** ListNode \*head)
10. {
11. **if**(!head)
12. **return** NULL;
13. **struct** ListNode \*fast = head;
14. **struct** ListNode \*slow = head;
16. **while**(fast!=NULL&&fast->next!=NULL) {
18. fast = fast->next->next;
19. slow = slow->next;
20. **if**(fast==slow)
21. **break**;
22. }
23. **if**(fast==slow&&fast->next!=NULL) {
24. slow = head;
25. **while**(slow!=fast) {
26. slow = slow->next;
27. fast = fast->next;
28. }
29. **return** slow;
30. }
31. **return** NULL;
32. }

感觉上面写的思路还是不够清晰，首先整理一下判断链表是否有环的代码

**[cpp]** [view plain](https://blog.csdn.net/xy010902100449/article/details/48995255) [copy](https://blog.csdn.net/xy010902100449/article/details/48995255)

1. /\*Improved version of the program\*/
2. **bool** hasCycle(**struct** ListNode \*head) {
3. **if**(!head)
4. **return** **false**;
5. **struct** ListNode \*fast = head;
6. **struct** ListNode \*slow = head;
8. **do** {
9. **if**(!fast||!slow)
10. **return** **false**;/\*If LinkNode exist NULL node,then return false\*/
11. fast = fast->next;
12. slow = slow->next;
13. **if**(fast)
14. fast = fast->next;
15. **else**
16. **return** **false**;/\*If LinkNode exist NULL node,then return false\*/
17. }**while**(fast!=slow);
19. **return** **true**;
20. }

判断环的第一个相交节点改进代码如下：

**[cpp]** [view plain](https://blog.csdn.net/xy010902100449/article/details/48995255) [copy](https://blog.csdn.net/xy010902100449/article/details/48995255)

1. /\*Improved version of the program\*/
2. **struct** ListNode \*detectCycle(**struct** ListNode \*head)
3. {
4. **if**(!head)
5. **return** NULL;
6. **struct** ListNode \*fast = head;
7. **struct** ListNode \*slow = head;
9. **do** {
10. **if**(!fast||!slow)
11. **return** NULL;/\*If LinkNode exist NULL node,then return NULL\*/
12. fast = fast->next;
13. slow = slow->next;
14. **if**(fast)
15. fast = fast->next;
16. **else**
17. **return** NULL;/\*If LinkNode exist NULL node,then return NULL\*/
18. }**while**(fast!=slow);
20. slow = head;
21. **while**(slow!=fast) {
22. fast = fast->next;
23. slow = slow->next;
24. }
25. **return** slow;
26. }