链表

> 题目：160. Intersection of Two Linked Lists
>
> 语言: python3
>
> 英文版链接：https://leetcode.com/problems/intersection-of-two-linked-lists/description/
>
> 中文版链接：https://leetcode-cn.com/problems/intersection-of-two-linked-lists/description/

题目分析

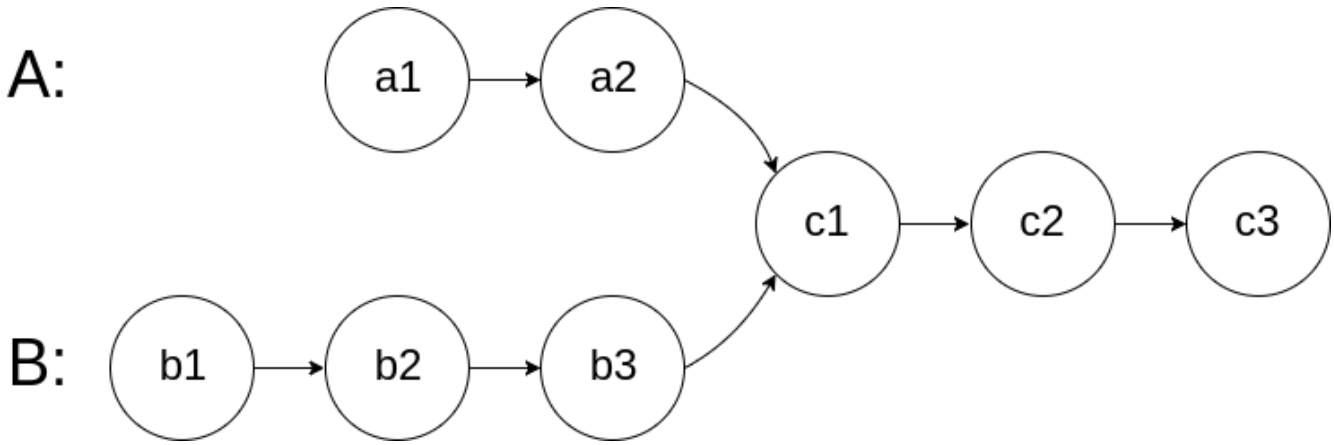
编写一个程序，找到两个单链表相交的起始节点。

要求：时间复杂度为 O(N)，空间复杂度为 O(1)

设 A 的长度为 a + c，B 的长度为 b + c，其中 c 为尾部公共部分长度，可知 a + c + b = b + c + a。

当访问 A 链表的指针访问到链表尾部时，令它从链表 B 的头部开始访问链表 B；同样地，当访问 B 链表的指针访问到链表尾部时，令它从链表 A 的头部开始访问链表 A。这样就能控制访问 A 和 B 两个链表的指针能同时访问到交点。

我们可以举个例子来说明：



我们从A访问再访问B，从B访问再访问A，将得到两个序列：

1. a1, a2, c1, c2, c3, b1, b2, b3, **c1, c2, c3**
2. b1, b2, b3, c1, c2, c3, a1, a2, **c1, c2, c3**

可以发现，如果链表A和B有相交的话，一定可以找到相交的节点。

答案

```
class Solution(object):
    def getIntersectionNode(self, headA, headB):
        """
        :type head1, head1: ListNode
        :rtype: ListNode
```

```
        """
        if headA is None or headB is None:
            return None
        l1 = headA
        l2 = headB
        while l1 != l2:
            if l1 is not None:
                l1 = l1.next
            else:
                l1 = headB
            if l2 is not None:
                l2 = l2.next
            else:
                l2 = headA
        return l1
```