剑指 Offer 52. 两个链表的第一个公共节点

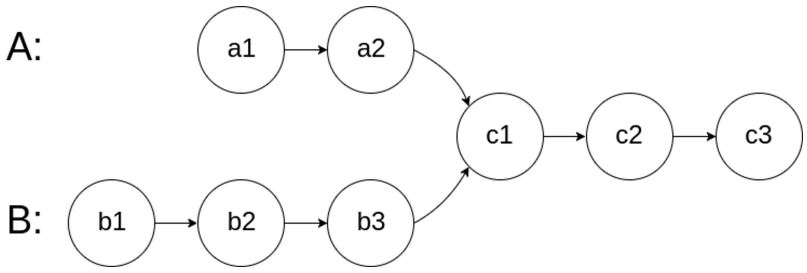
Friday, June 10, 2022 11:14 AM

https://leetcode.cn/problems/liang-ge-lian-biao-de-di-yi-ge-gong-gong-jie-dian-lcof/

简单

输入两个链表，找出它们的第一个公共节点。

如下面的两个链表：



在节点 c1 开始相交。

注意：

如果两个链表没有交点，返回 null.

在返回结果后，两个链表仍须保持原有的结构。

可假定整个链表结构中没有循环。

程序尽量满足 O(n) 时间复杂度，且仅用 O(1) 内存。

我的思路：（有相交则屁股一样长，先对齐屁股，再去找）

1.先同时遍历，直到 其中一个遍历完 停下

2.若A非空，继续遍历A，lenA表示其剩下的长度。若B非空，lenB同理。

3.求得的lenA或lenB 即 表示比对方长出来的长度。

4.长的先走 比对方长出来的长度。

5.此时两者离末尾长度一致，同时遍历，若有**相同（不是值相等）**节点则返回该节点。

遍历结束都没有则返回null。

code:

```
public class Solution {
    public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
        // 判空
        if(headA==null||headB==null) return null;
        // 要求不改变结构 故使用 cur指针
        ListNode curA=headA;
        ListNode curB=headB;
        // 表示其 比对方长的长度
        int lenA=0;
        int lenB=0;
        // 其中谁遍历完即退出
        while(curA!=null&&curB!=null){
            curA=curA.next;
            curB=curB.next;
        }
```

```
        // A没完的情况
        while(curA!=null){
            lenA+=1;
            curA=curA.next;
        }
        // B没完
        while(curB!=null){
            lenB+=1;
            curB=curB.next;
        }
        // 从头再来
        curA=headA;
        curB=headB;
        // 长的先走长出来的长度
        if(lenA>0){
            while(lenA>0){
                curA=curA.next;
                lenA--;
            }
        }else if(lenB>0){
            while(lenB>0){
                curB=curB.next;
                lenB--;
            }
        }
        // 到这里 AB剩下的一样长了，遍历，有相同的就是第一个相交点
        while(curA!=null&&curB!=null){
            if(curA==curB){
                return curA;
            }else{
                curA=curA.next;
                curB=curB.next;
            }
        }
        return null;
    }
}
```

优化：（脑筋急转弯，仅减少了代码量）

A长a B长b 相交部分长c

a + (b - c) = b + (a - c)

做法：curA curB同时分别在 A\B 上走，

curA 在 A 上走完就到 B 上走，curB在B上走完就到A上走。

1. 有相交的情况

   curA走过的路： a + b-c

   curB走过的路： b + a-c

   此时 curA与curB在相交点相遇(curA=curB)。

2. 没有相交的情况

   curA走过的路： a + b

curB走过的路： b + a

此时 curA与curB在null相遇(curA==curB)。

故：循环跳出条件：curA==curB

code：

```
public class Solution {
    public ListNode getIntersectionNode(ListNode headA, ListNode headB) {
        if(headA==null||headB==null) return null;
        ListNode curA = headA;
        ListNode curB = headB;
        // 循环跳出条件
        while (curA != curB) {
            // A上走完跳到B上
            if( curA == null){
                curA = headB;
            }else{
                curA = curA.next;
            }
            // B上走完跳到A上
            if( curB == null){
                curB = headA;
            }else{
                curB = curB.next;
            }
        }
        // while循环出来 curA==curB,不是null就是相遇的第一个节点
        return curA;
        }
}
```