/\*

Write a program to find the node at which the intersection of two singly linked lists begins.

For example, the following two linked lists:

A: a1 → a2

↘

c1 → c2 → c3

↗

B: b1 → b2 → b3

begin to intersect at node c1.

Notes:

If the two linked lists have no intersection at all, return null.

The linked lists must retain their original structure after the function returns.

You may assume there are no cycles anywhere in the entire linked structure.

Your code should preferably run in O(n) time and use only O(1) memory.

计算长度法。假设链表A和B相交，则交点及之后的部分长度是相同的，相差的是交点前的部分。计算A和B的长度差n，较长的那个先走N步，然后和较短的那个同时遍历。如果指针指向同一元素，则为交点，否则A和B 没有交点。该方法能够满足线性时间复杂度和常数空间复杂度的要求。

\*/

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution {

public:

ListNode \*getIntersectionNode(ListNode \*headA, ListNode \*headB)

{

int lenA=0,lenB=0;

ListNode \*p=headA;

while(p)

{

p=p->next;

lenA++;

}

p=headB;

while(p)

{

p=p->next;

lenB++;

}

if(lenA>=lenB)

{

p=headA;

for(int i=0;i<lenA-lenB;i++)

p=p->next;

ListNode \*q=headB;

while(p && q)

{

if(p==q)

return p;

p=p->next;

q=q->next;

}

return NULL;

}

else

{

p=headB;

for(int i=0;i<lenB-lenA;i++)

p=p->next;

ListNode \*q=headA;

while(p && q)

{

if(p==q)

return p;

p=p->next;

q=q->next;

}

return NULL;

}

}

};