/* 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. */

-
- 思想：
- (1) 首先分别遍历两个链表，得到两个链表的长度m,n；然后用两个指针分别指向链表头，较长的链表移动abs(m-n) 步，这样从此以后两个链表以后的部分拥有相同的长度，每次前进一步，如果两个链表有共同节点，那么它们就会至少有一次相等（不可能在前面的m-n步中存在共同节点，因为这样的话两个链表的长度将相同）

```
public ListNode getIntersectionNode(ListNode headA, ListNode headB) {

    int aLength = 0, bLength = 0;
    ListNode aPointer = headA, bPointer = headB;

    while ( aPointer != null ) {
        aLength++;
        aPointer = aPointer.next;
    }
    while ( bPointer != null ) {
        bLength++;
        bPointer = bPointer.next;
    }

    if ( aLength < bLength ) {
        return getIntersectionNodeHelper(headA, headB, bLength - aLength);
    } else {
        return getIntersectionNodeHelper(headB, headA, aLength - bLength);
    }

}
```