Problem # 160: Intersection of two Linked Lists (Easy)

------------------------------------------------------------------------

<https://leetcode.com/problems/intersection-of-two-linked-lists/>

My Solution:

Runtime beats 94.80%

1. Find the length of the two linked lists by traversing the nodes from the head and counting the nodes from the head taking head as starting point.

2. Find the difference between the length.

3. If out which list is longer, and traverse as many nodes as the difference from the head. Now both lists have equal number of nodes left to get to the intersection point.

4. Advance the pointer for both list one step at a time and check if they are both at the same node. If so, return the pointer for one of the lists.

-----------------------------------------------

# Definition for singly-linked list.

# class ListNode:

# def \_\_init\_\_(self, x):

# self.val = x

# self.next = None

def getLength(head):

if not head:

return(0)

length = 0

current = head

while current:

current = current.next

length += 1

return(length)

class Solution:

def getIntersectionNode(self, headA: ListNode, headB: ListNode) -> ListNode:

lenA = getLength(headA)

lenB = getLength(headB)

#print("lenA = ", lenA)

#print("lenB = ", lenB)

diff = abs(lenA - lenB)

#print("diff = ", diff)

p = headA

q = headB

if lenA > lenB:

for i in range(diff):

p = p.next

elif lenB > lenA:

for i in range(diff):

q = q.next

while p:

if p == q:

return(p)

p = p.next

q = q.next

return(None)