**160. Intersection of Two Linked Lists**

Easy

2435243FavoriteShare

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

For example, the following two linked lists:

[](https://assets.leetcode.com/uploads/2018/12/13/160_statement.png)

begin to intersect at node c1.

**Example 1:**

[](https://assets.leetcode.com/uploads/2018/12/13/160_example_1.png)

**Input:** intersectVal = 8, listA = [4,1,8,4,5], listB = [5,0,1,8,4,5], skipA = 2, skipB = 3

**Output:** Reference of the node with value = 8

**Input Explanation:** The intersected node's value is 8 (note that this must not be 0 if the two lists intersect). From the head of A, it reads as [4,1,8,4,5]. From the head of B, it reads as [5,0,1,8,4,5]. There are 2 nodes before the intersected node in A; There are 3 nodes before the intersected node in B.

**Example 3:**

[](https://assets.leetcode.com/uploads/2018/12/13/160_example_3.png)

**Input:** intersectVal = 0, listA = [2,6,4], listB = [1,5], skipA = 3, skipB = 2

**Output:** null

**Input Explanation:** From the head of A, it reads as [2,6,4]. From the head of B, it reads as [1,5]. Since the two lists do not intersect, intersectVal must be 0, while skipA and skipB can be arbitrary values.

**Explanation:** The two lists do not intersect, so return null.

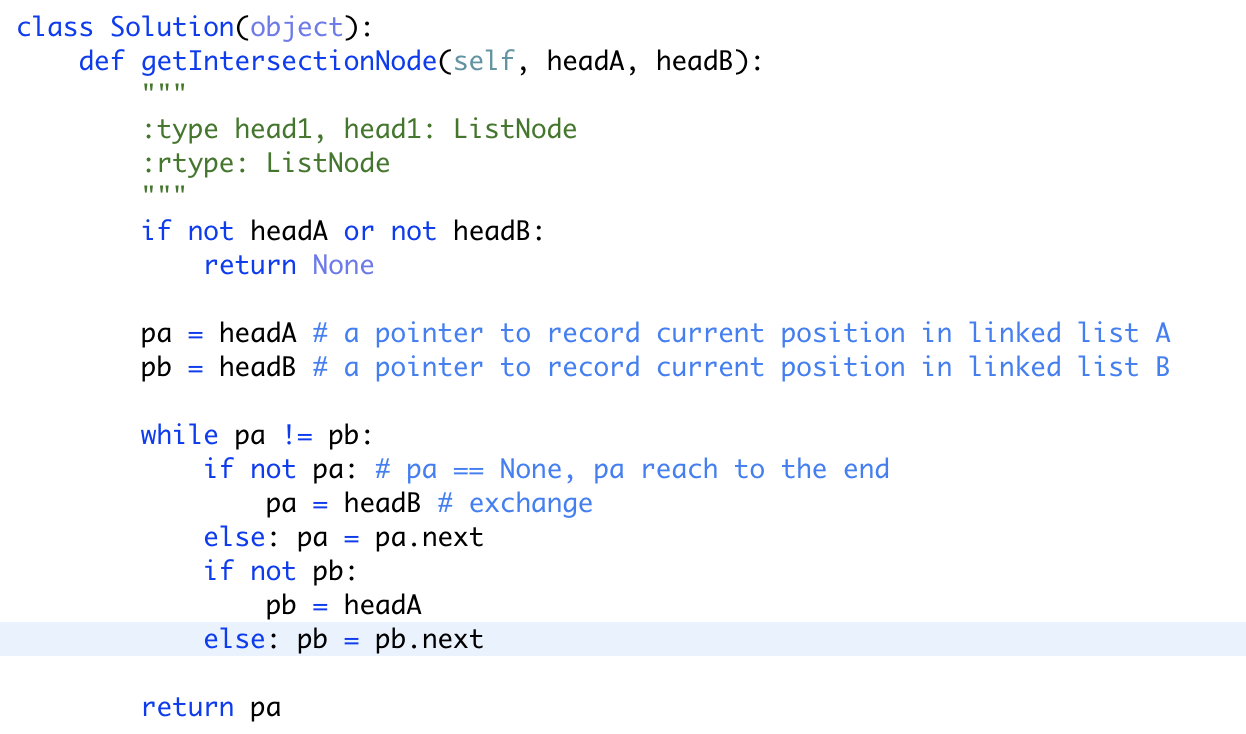
**Notes:**

* 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.

**看到这题，我第一眼是懵逼的，虽然是个easy题。首先不懂输入到底是个什么玩意，我当时就没懂testcase为什么会直接告诉我们第几个点merge，这不是直接告诉答案了吗，后来才意识到输入其实只有两个linked list的head node。然后我不太清楚如何判断在哪个位置merge，比如我之前一直认为当两个节点的值一样时 (nodeA.val == nodeB.val)，就是交叉点，后来才意识到不对，应该是到达了同一个节点才算是交叉点（nodeA == nodeB）。最后我一下子不能get这题的难点在哪，难点在于两个linked list长度是不一样的，如果他们同时移动，这必然导致他们会先后到达merge点。所以除了使用暴力解法，如何让他们同时到达merge点就成为了算法关注的重点。**

1. **交叉抵消长短不一**

**我们知道len(linked\_listA) 虽然不等于len(linked\_listB)，但是len(linked\_listA)+ len(linked\_listB)一定等于len(linked\_listB)+ len(linked\_listA)。所以当第一次循环分别使他们依次到底，不如让他们交换再遍历第二次，而这一次，他们一定会同时到达merge点，然后就可以使用（nodeA == nodeB）语句进行判断。**

****