0160. 相交链表

ITCharge 大约 2 分钟

- 标签：哈希表、链表、双指针
- 难度：简单

题目链接

- 0160. 相交链表 - 力扣

题目大意

描述：给定 `listA` 、 `listB` 两个链表。

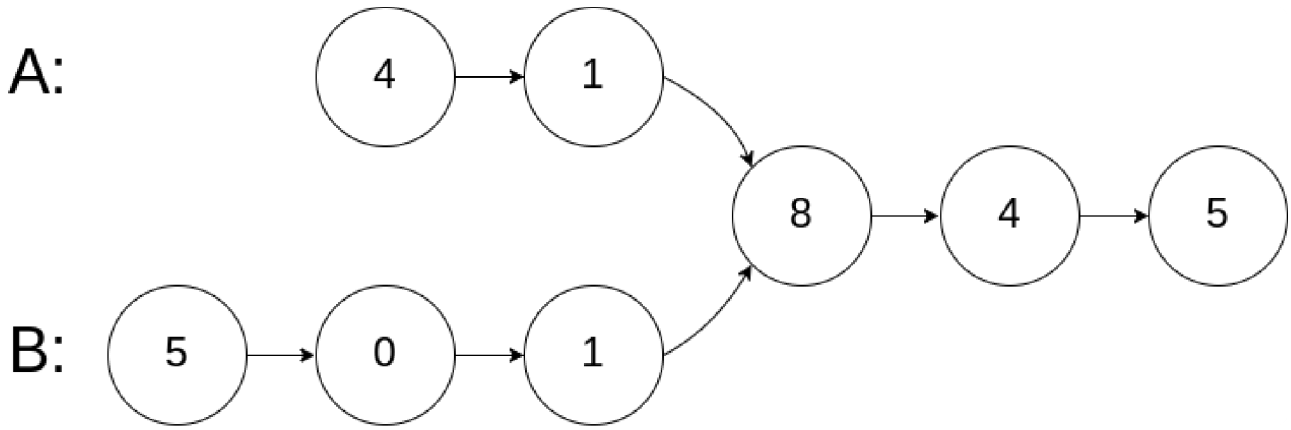
要求：判断两个链表是否相交，返回相交的起始点。如果不相交，则返回 `None` 。

说明：

- `listA` 中节点数目为 m。
- `listB` 中节点数目为 n。
- $1 \leq m, n \leq 3 * 10^4$。
- $1 \leq Node.val \leq 10^5$。
- $0 \leq skipA \leq m$。
- $0 \leq skipB \leq n$。
- 如果 `listA` 和 `listB` 没有交点， `intersectVal` 为 0。
- 如果 `listA` 和 `listB` 有交点， `intersectVal == listA[skipA] == listB[skipB]` 。

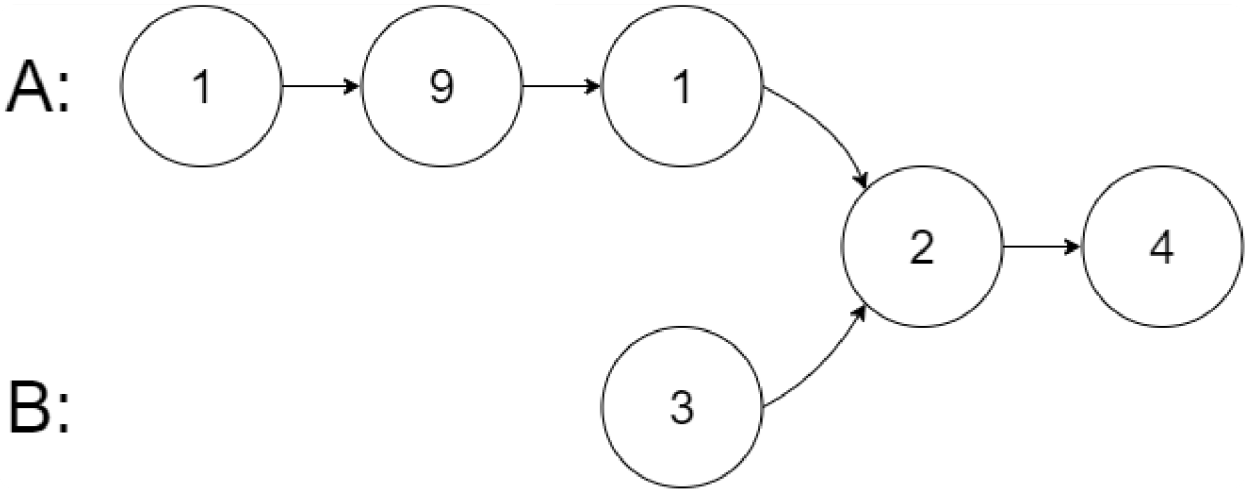
示例：

- 示例 1：




```py
输入：intersectVal = 8, listA = [4,1,8,4,5], listB = [5,6,1,8,4,5], skipA = 2, skipB = 3
输出：Intersected at '8'
解释：相交节点的值为 8 （注意，如果两个链表相交则不能为 0）。
从各自的表头开始算起，链表 A 为 [4,1,8,4,5]，链表 B 为 [5,6,1,8,4,5]。
在 A 中，相交节点前有 2 个节点；在 B 中，相交节点前有 3 个节点。
— 请注意相交节点的值不为 1，因为在链表 A 和链表 B 之中值为 1 的节点 (A 中第二个节点和 B 中第三个节点) 是不同的节点。换句话说，它们在内存中指向两个不同的位置，而链表 A 和链表 B 中值为 8 的节点 (A 中第三个节点，B 中第四个节点) 在内存中指向相同的位置。
```

- 示例 2:

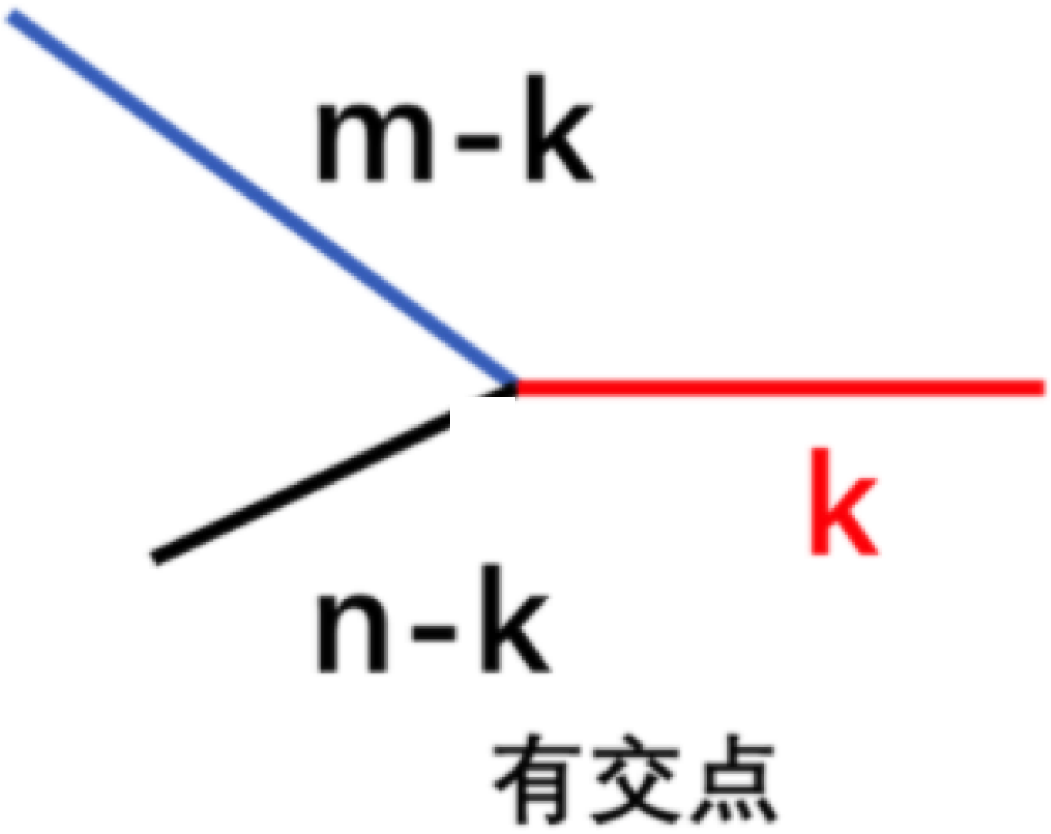



```py
输入：intersectVal = 2, listA = [1,9,1,2,4], listB = [3,2,4], skipA = 3, skipB = 1
输出：Intersected at '2'
解释：相交节点的值为 2 （注意，如果两个链表相交则不能为 0）。
从各自的表头开始算起，链表 A 为 [1,9,1,2,4]，链表 B 为 [3,2,4]。
在 A 中，相交节点前有 3 个节点；在 B 中，相交节点前有 1 个节点。
```

解题思路

思路 1：双指针

如果两个链表相交，那么从相交位置开始，到结束，必有一段等长且相同的节点。假设链表 `listA` 的长度为 m、链表 `listB` 的长度为 n，他们的相交序列有 k 个，则相交情况可以如下如所示：

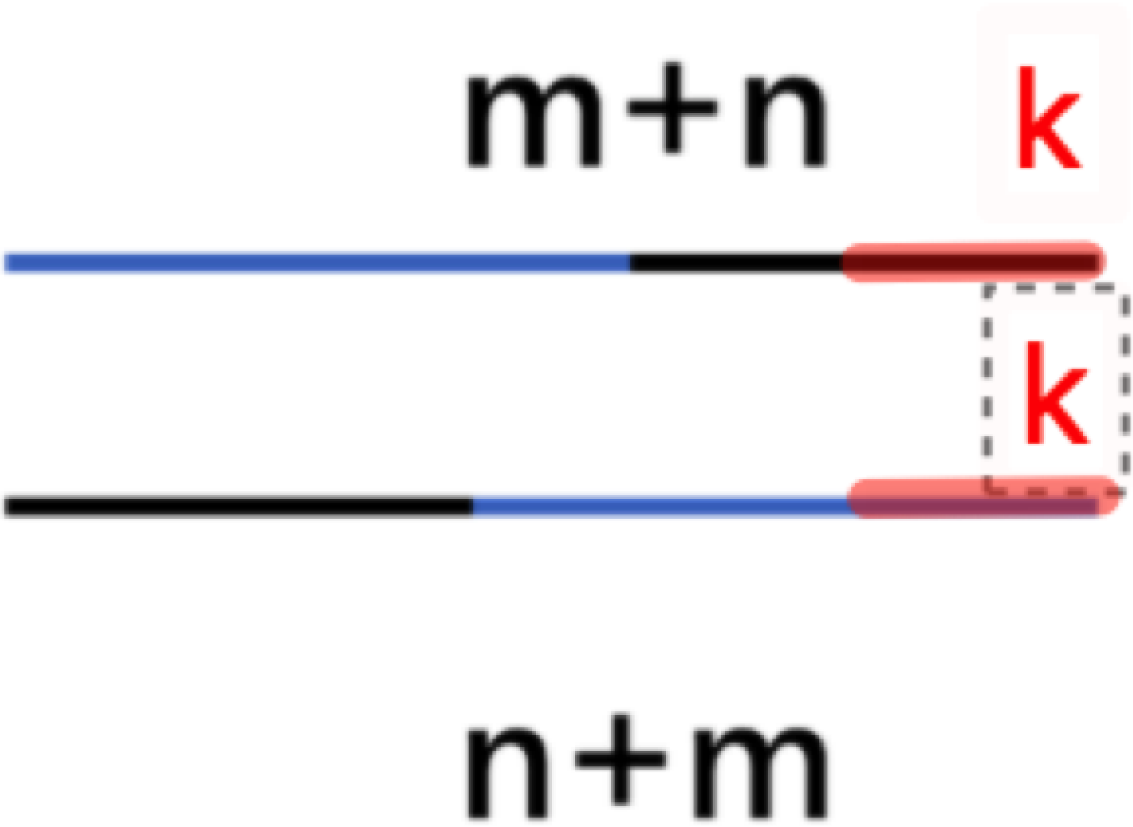


现在问题是如何找到 $m-k$ 或者 $n-k$ 的位置。

考虑将链表 `listA` 的末尾拼接上链表 `listB`，链表 `listB` 的末尾拼接上链表 `listA`。

然后使用两个指针 `pA` 、 `pB`，分别从链表 `listA`、链表 `listB` 的头节点开始遍历，如果走到共同的节点，则返回该节点。

否则走到两个链表末尾，返回 `None`。



思路 1：代码

```py
class Solution:
    def getIntersectionNode(self, headA: ListNode, headB: ListNode) -> ListNode:
        if headA == None or headB == None:
            return None
        pA = headA
        pB = headB
        while pA != pB:
            pA = pA.next if pA != None else headB
            pB = pB.next if pB != None else headA
        return pA
```

思路 1：复杂度分析

- **时间复杂度**：$O(m+n)$。
- **空间复杂度**：$O(1)$。