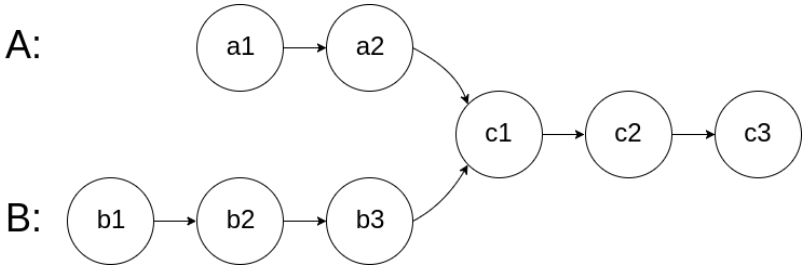
相交链表

给你两个单链表的头节点 headA 和 headB ，请你找出并返回两个单链表相交的起始节点。如果两个链表不存在相交节点，返回 null 。

图示两个链表在节点 c1 开始相交：



题目数据 **保证** 整个链式结构中不存在环。

注意，函数返回结果后，链表必须 **保持其原始结构** 。

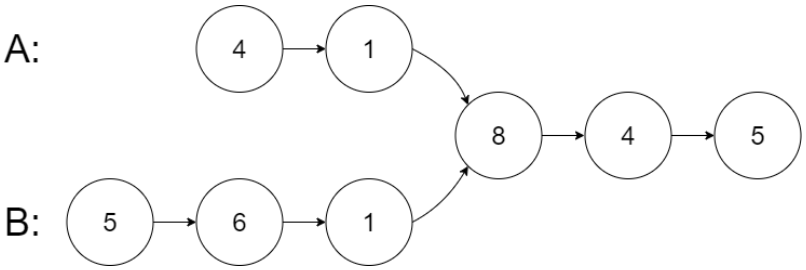
自定义评测：

评测系统 的输入如下（你设计的程序 **不适用** 此输入）：

- intersectVal - 相交的起始节点的值。如果不存在相交节点，这一值为 0
- listA - 第一个链表
- listB - 第二个链表
- skipA - 在 listA 中（从头节点开始）跳到交叉节点的节点数
- skipB - 在 listB 中（从头节点开始）跳到交叉节点的节点数

评测系统将根据这些输入创建链式数据结构，并将两个头节点 headA 和 headB 传递给你的程序。如果程序能够正确返回相交节点，那么你的解决方案将被 **视作正确答案** 。

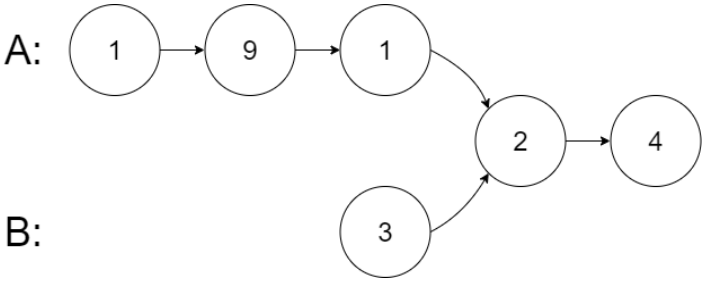
示例 1：



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

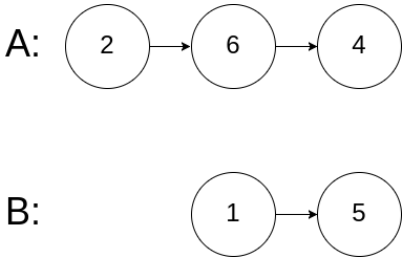
和 B 中第三个节点）是不同的节点。换句话说，它们在内存中指向两个不同的位置，而链表 A 和链表 B 中值为 8 的节点（A 中第三个节点，B 中第四个节点）在内存中指向相同的位置。

示例 2：



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。

示例 3：



输入： intersectVal = 0, listA = [2,6,4], listB = [1,5], skipA = 3, skipB = 2
输出： null
解释： 从各自的表头开始算起，链表 A 为 [2,6,4]，链表 B 为 [1,5]。
由于这两个链表不相交，所以 intersectVal 必须为 0，而 skipA 和 skipB 可以是任意值。
这两个链表不相交，因此返回 null 。

提示：

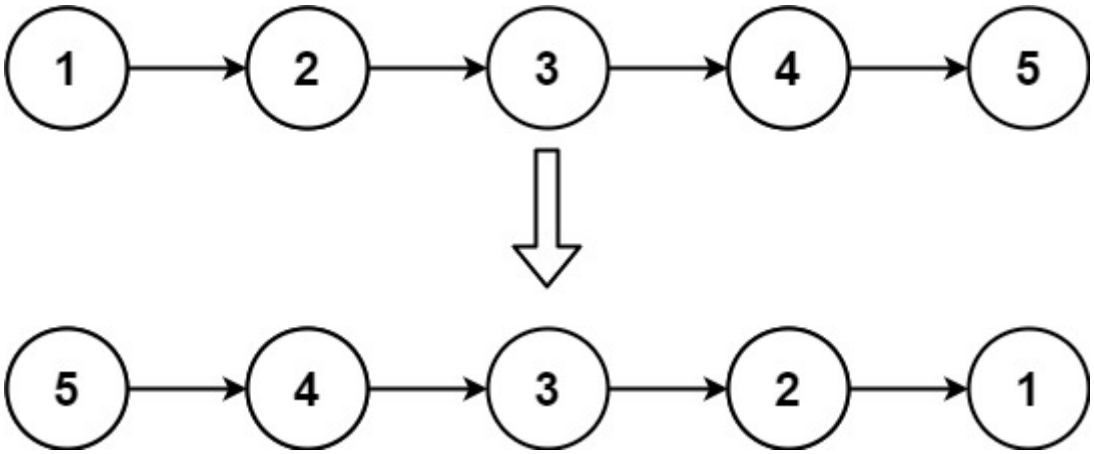
- listA 中节点数目为 m
- listB 中节点数目为 n
- $1 <= m, n <= 3 * 10^4$
- $1 <= Node.val <= 10^5$
- 0 <= skipA <= m
- 0 <= skipB <= n
- 如果 listA 和 listB 没有交点，intersectVal 为 0
- 如果 listA 和 listB 有交点，intersectVal == listA[skipA] == listB[skipB]

进阶：你能否设计一个时间复杂度 O(m + n)、仅用 O(1) 内存的解决方案?

反转链表

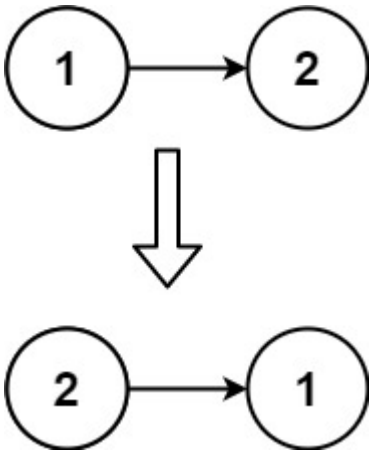
给你单链表的头节点 head ，请你反转链表，并返回反转后的链表。

示例 1：



输入： head = [1,2,3,4,5]
输出： [5,4,3,2,1]

示例 2：



输入： head = [1,2]
输出： [2,1]

示例 3：

输入： head = []
输出： []

提示：

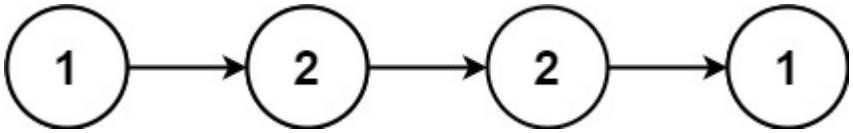
- 链表中节点的数目范围是 [0, 5000]
- -5000 <= Node.val <= 5000

进阶：链表可以选用迭代或递归方式完成反转。你能否用两种方法解决这道题？

回文链表

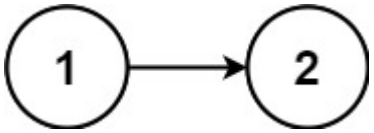
给你一个单链表的头节点 `head` ， 请你判断该链表是否为回文链表。如果是，返回 `true` ； 否则，返回 `false` 。

示例 1：



输入： `head = [1,2,2,1]`
输出： `true`

示例 2：



输入： `head = [1,2]`
输出： `false`

提示：

- 链表中节点数目在范围`[1, 10^5]` 内
- `0 <= Node.val <= 9`

进阶： 你能否用 `O(n)` 时间复杂度和 `O(1)` 空间复杂度解决此题?

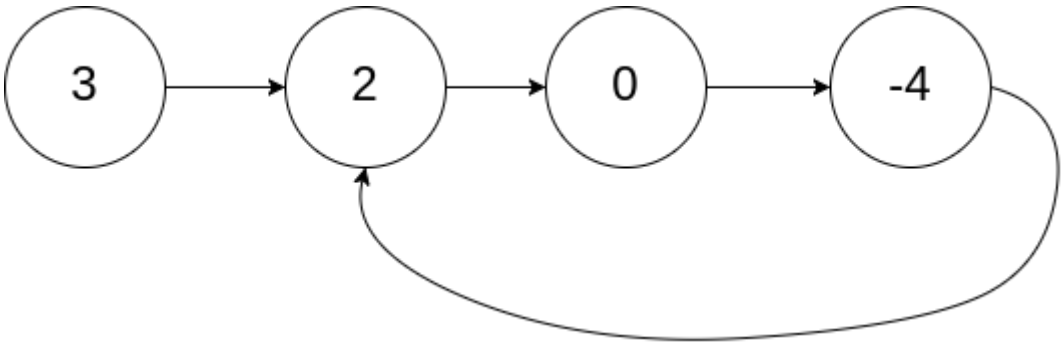
环形链表

给你一个链表的头节点 head ，判断链表中是否有环。

如果链表中有某个节点，可以通过连续跟踪 next 指针再次到达，则链表中存在环。 为了表示给定链表中的环，评测系统内部使用整数 pos 来表示链表尾连接到链表中的位置（索引从 0 开始）。**注意：pos 不作为参数进行传递** 。仅仅是为了标识链表的实际情况。

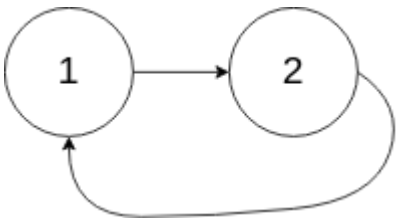
如果链表中存在环 ，则返回 true 。 否则，返回 false 。

示例 1：



输入：head = [3,2,0,-4], pos = 1
输出：true
解释：链表中有一个环，其尾部连接到第二个节点。

示例 2：



输入：head = [1,2], pos = 0
输出：true
解释：链表中有一个环，其尾部连接到第一个节点。

示例 3：

输入：head = [1], pos = -1
输出：false
解释：链表中没有环。

提示：

- 链表中节点的数目范围是 `[0, 10^4]`
- `-10^5 <= Node.val <= 10^5`
- `pos` 为 `-1` 或者链表中的一个 **有效索引** 。

进阶：你能用 `O(1)`（即，常量）内存解决此问题吗？

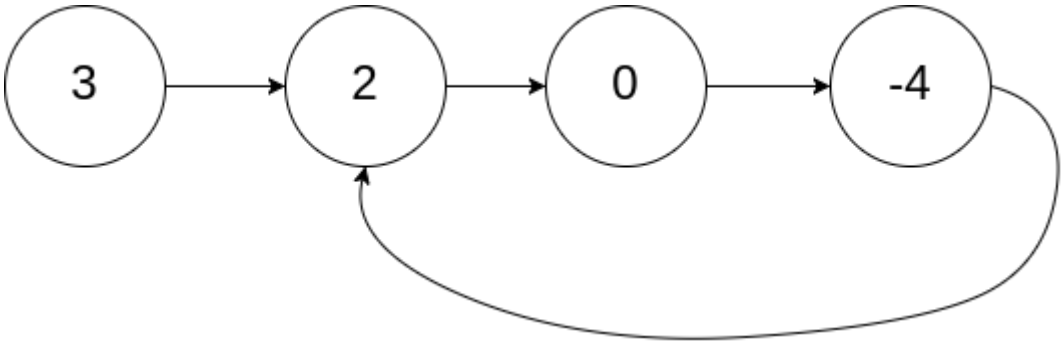
环形链表 II

给定一个链表的头节点 `head` ，返回链表开始入环的第一个节点。 如果链表无环，则返回 *`null`*。

如果链表中有某个节点，可以通过连续跟踪 `next` 指针再次到达，则链表中存在环。 为了表示给定链表中的环，评测系统内部使用整数 `pos` 来表示链表尾连接到链表中的位置（**索引从 0 开始**）。如果 `pos` 是 `-1`，则在该链表中没有环。**注意：pos 不作为参数进行传递**，仅仅是为了标识链表的实际情况。

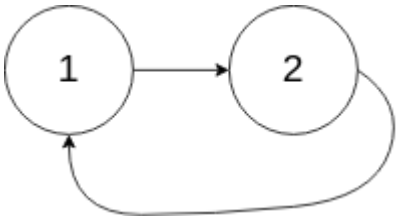
不允许修改 链表。

示例 1：



输入：`head = [3,2,0,-4], pos = 1`
输出：返回索引为 `1` 的链表节点
解释：链表中有一个环，其尾部连接到第二个节点。

示例 2：



输入：`head = [1,2], pos = 0`
输出：返回索引为 `0` 的链表节点
解释：链表中有一个环，其尾部连接到第一个节点。

示例 3：

输入：`head = [1], pos = -1`
输出：返回 `null`
解释：链表中没有环。

提示：

- 链表中节点的数目范围在范围 `[0, 10^4]` 内
- `-10^5 <= Node.val <= 10^5`
- `pos` 的值为 `-1` 或者链表中的一个有效索引

进阶：你是否可以使用 `O(1)` 空间解决此题？

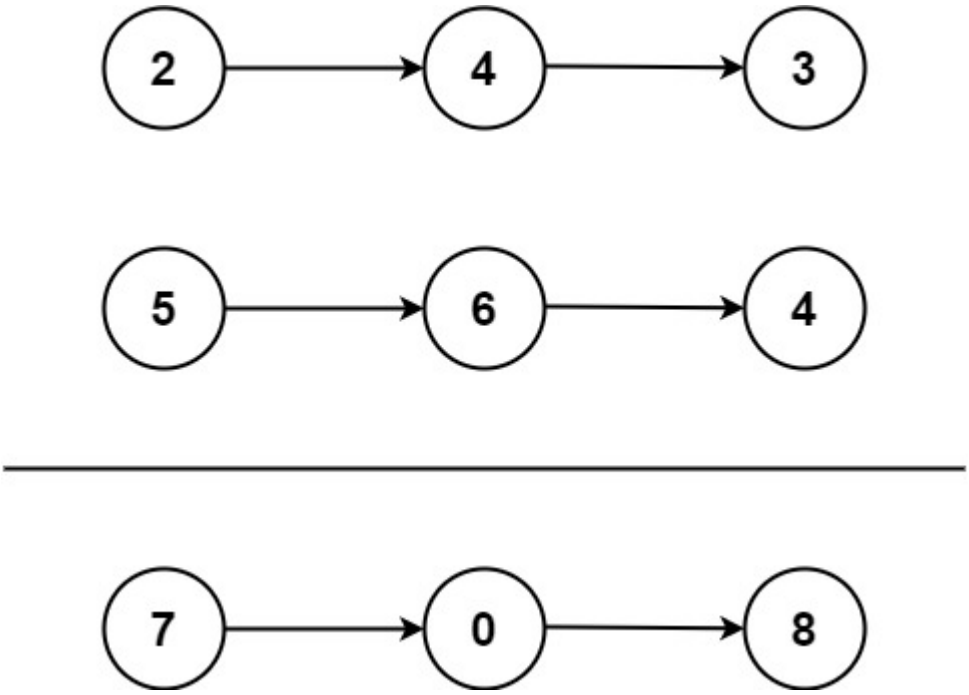
两数相加

给你两个 **非空** 的链表，表示两个非负的整数。它们每位数字都是按照 **逆序** 的方式存储的，并且每个节点只能存储 **一位** 数字。

请你将两个数相加，并以相同形式返回一个表示和的链表。

你可以假设除了数字 0 之外，这两个数都不会以 0 开头。

示例 1：



输入： l1 = [2,4,3], l2 = [5,6,4]
输出： [7,0,8]
解释： 342 + 465 = 807.

示例 2：

输入： l1 = [0], l2 = [0]
输出： [0]

示例 3：

输入： l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]
输出： [8,9,9,9,0,0,0,1]

提示：

- 每个链表中的节点数在范围 [1, 100] 内
- 0 <= Node.val <= 9

· 题目数据保证列表表示的数字不含前导零