Leetcode 总结版~链表

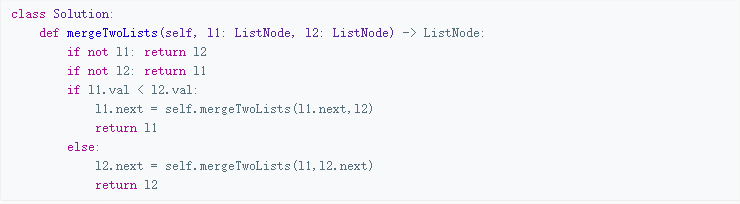
1. **合并两个有序链表**

将两个有序链表合并为一个新的有序链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

示例：

输入：1->2->4, 1->3->4

输出：1->1->2->3->4->4



1. **反转链表**

反转一个单链表。

**示例:**

**输入:** 1->2->3->4->5->NULL

**输出:** 5->4->3->2->1->NULL



1. **回文链表**

请判断一个链表是否为回文链表。

示例 1:

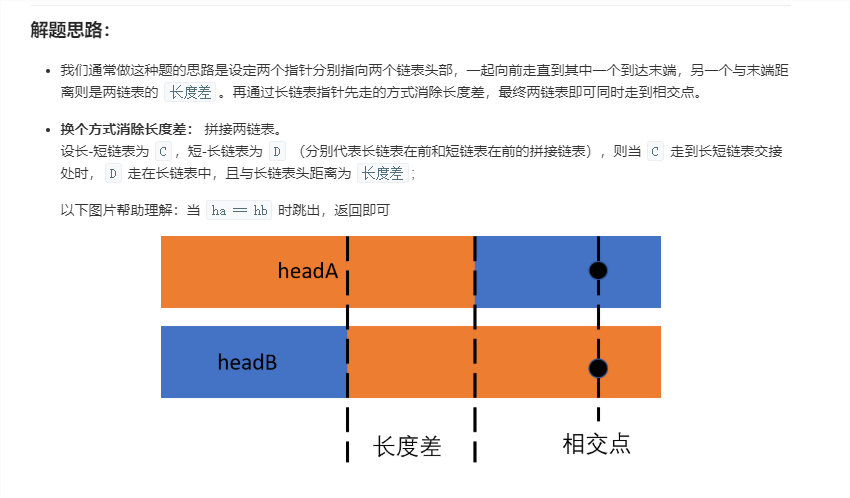
输入: 1->2

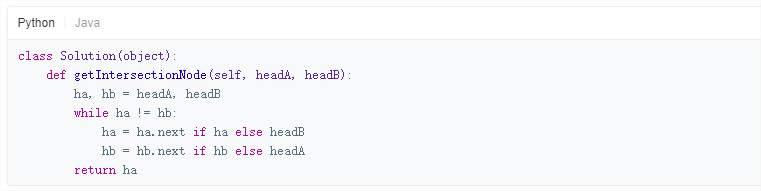
输出: false



1. **相交链表**

编写一个程序，找到两个单链表相交的起始节点。





1. **两数相加**

给出两个 非空 的链表用来表示两个非负的整数。其中，它们各自的位数是按照 逆序 的方式存储的，并且它们的每个节点只能存储 一位 数字。

如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。

您可以假设除了数字 0 之外，这两个数都不会以 0 开头。

示例：

输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)

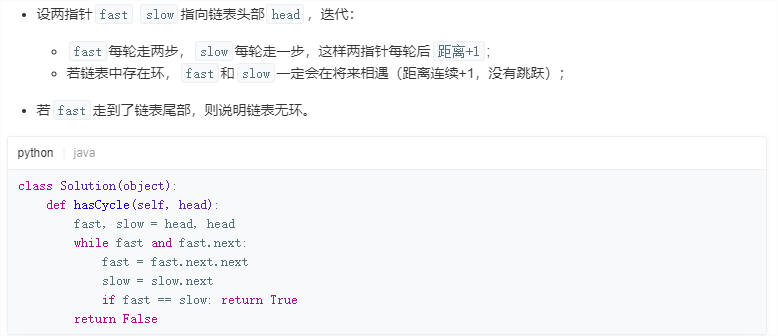
输出：7 -> 0 -> 8

原因：342 + 465 = 807



1. **环形链表**

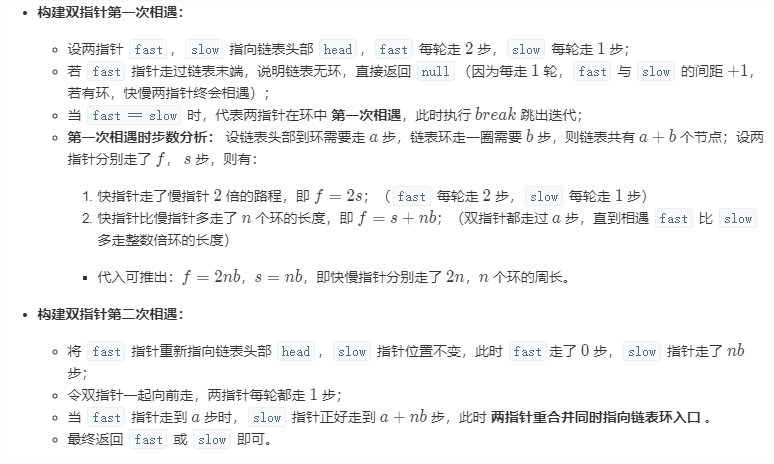
给定一个链表，判断链表中是否有环。为了表示给定链表中的环，我们使用整数 pos 来表示链表尾连接到链表中的位置（索引从 0 开始）。 如果 pos 是 -1，则在该链表中没有环。



1. **环形链表II**

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

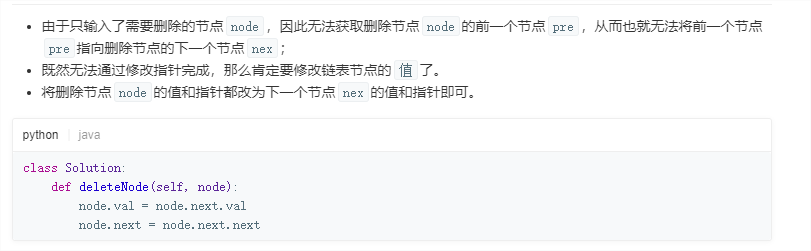
为了表示给定链表中的环，我们使用整数 pos 来表示链表尾连接到链表中的位置（索引从 0 开始）。 如果 pos 是 -1，则在该链表中没有环。





1. **删除链表的节点**

请编写一个函数，使其可以删除某个链表中给定的（非末尾）节点，你将只被给定要求被删除的节点。



1. **删除链表的倒数第K个节点**

给定一个链表，删除链表的倒数第 n 个节点，并且返回链表的头结点。

示例：

给定一个链表: 1->2->3->4->5, 和 n = 2.

当删除了倒数第二个节点后，链表变为 1->2->3->5.



1. **排序链表**

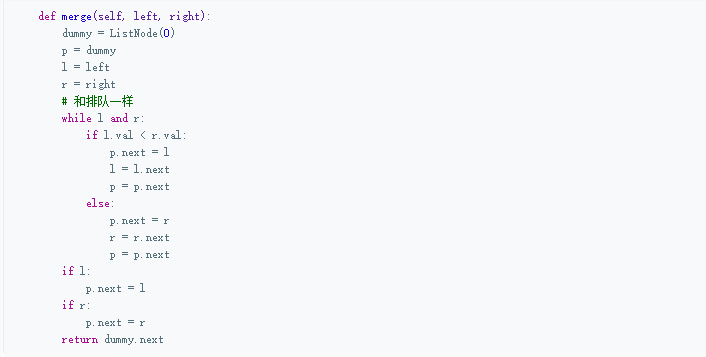
在 *O*(*n* log *n*) 时间复杂度和常数级空间复杂度下，对链表进行排序。

示例 1:

输入: 4->2->1->3

输出: 1->2->3->4

用归并排序：

附排序数组：



1. **旋转链表**

给定一个链表，旋转链表，将链表每个节点向右移动 k 个位置，其中 k 是非负数。

示例 1:

输入: 1->2->3->4->5->NULL, k = 2

输出: 4->5->1->2->3->NULL

解释:

向右旋转 1 步: 5->1->2->3->4->NULL

向右旋转 2 步: 4->5->1->2->3->NULL



1. **合并k个有序链表**

合并 k 个排序链表，返回合并后的排序链表。请分析和描述算法的复杂度。

示例:

输入:

[

  1->4->5,

  1->3->4,

  2->6

]

输出: 1->1->2->3->4->4->5->6

