编程判断两个链表是否相交

给出两个单向链表的头指针（如图 3-8 所示），比如 h_1、h_2，判断这两个链表是否相交。这里为了简化问题，我们假设两个链表均不带环。

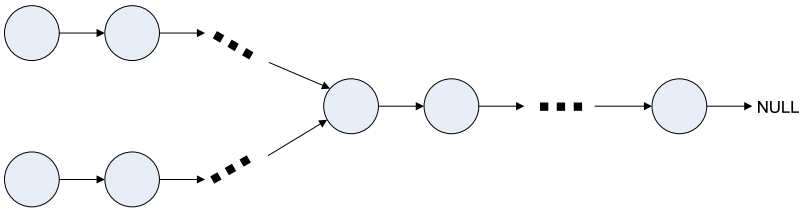

图 3-8　链表相交示意图

分析与解法

这样的一个问题，也许我们平时很少考虑。但在一个大的系统中，如果出现两个链表相交的情况，而且释放了其中一个链表的所有节点，那样就会造成信息的丢失，并且另一个与之相交的链表也会受到影响，这是我们不希望看到的。在特殊的情况下，的确需要出现相交的两个链表，我们希望在释放一个链表之前知道是否有其他链表跟当前这个链表相交。

【解法一】直观的想法

看到这个问题，我们的第一个想法估计都是，“不管三七二十一”，先判断第一个链表的每个节点是否在第二个链表中。这种方法的时间复杂度为 $O(\text{Length}(h_1) * \text{Length}(h_2))$。可见，这种方法很耗时间。

【解法二】利用计数的方法

很容易想到，如果两个链表相交，那么这两个链表就会有共同的节点。而节点地址又是节点的唯一标识。所以，如果我们能够判断两个链表中是否存在地址一致的节点，就可以知道这两个链表是否相交。一个简单的做法是对第一个链表的节点地址进行 hash 排序，建立 hash 表，然后针对第二个链表的每个节点的地址查询 hash 表，如果它在 hash 表中出现，那么说明第二个链表和第一个链表有共同的节点。这个方法的时间复杂度为 $O(\max(\text{Length}(h_1) + \text{Length}(h_2)))$。但是它同时需要附加 $O(\text{Length}(h_1))$的存储空间，以存储哈希表。虽然这样做减少了时间复杂度，但是是以增加存储空间为代价的。是否还有更好的方法呢，既能够以线性时间复杂度解决问题，又能减少存储空间？

【解法三】

由于两个链表都没有环，我们可以把第二个链表接在第一个链表后面，如果得到的链表有环，则说明这两个链表相交。否则，这两个链表不相交（如图 3-9 所示）。这样我们就把问题转化为判断一个链表是否有环。

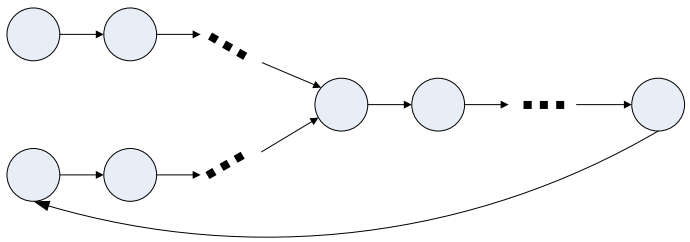

图 3-9　链表有环的情况

判断一个链表是否有环，也不是一个简单的问题，但是需要注意的是，在这里如果有环，则第二个链表的表头一定在环上，我们只需要从第二个链表开始遍历，看是否会回到起始点就可以判断出来。最后，当然可别忘了恢复原来的状态，去掉从第一个链表到第二个链表表头的指向。

这个方法总的时间复杂度也是线性的，但只需要常数的空间。

【解法四】

仔细观察题目中的图示，如果两个没有环的链表相交于某一节点的话，那么在这个节点之后的所有节点都是两个链表所共有的。那么我们能否利用这个特点简化我们的解法呢？困难在于我们并不知道哪个节点必定是两个链表共有的节点（如果它们相交的话）。进一步考虑“如果两个没有环的链表相交于某一节点的话，那么在这个节点之后的所有节点都是两个链表共有的”这个特点，我们可以知道，如果它们相交，则最后一个节点一定是共有的。而我们很容易能得到链表的最后一个节点，所以这成了我们简化解法的一个主要突破口。

先遍历第一个链表，记住最后一个节点。然后遍历第二个链表，到最后一个节点时和第一个链表的最后一个节点做比较，如果相同，则相交，否则，不相交。这样我们就得到了一个时间复杂度，它为 $O((\text{Length}(h_1) + \text{Length}(h_2))$，而且只用了一个额外的指针来存储最后一个节点。这个方法比解法三更胜一筹。

扩展问题

1. 如果链表可能有环呢?上面的方法需要怎么调整？

2. 如果我们需要求出两个链表相交的第一个节点呢？