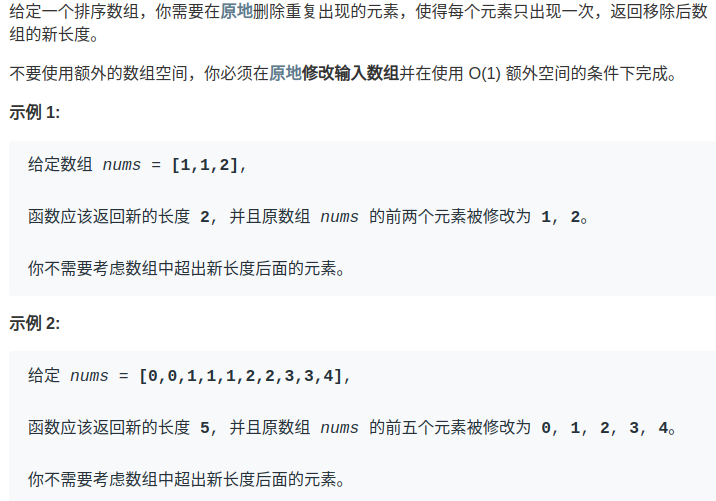
高效对有序数组/链表去重？

我们知道对于数组来说，在尾部插入、删除元素是比较高效的，时间复杂度是 O(1)，但是如果在中间或者开头插入、删除元素，就会涉及数据的搬移，时间复杂度为 O(N)，效率较低。

所以对于一般处理数组的算法问题，我们要尽可能只对数组尾部的元素进行操作，以避免额外的时间复杂度。

这篇文章讲讲如何对一个有序数组去重，先看下题目：



显然，由于数组已经排序，所以重复的元素一定连在一起，找出它们并不难，但如果毎找到一个重复元素就立即删除它，就是在数组中间进行删除操作，整个时间复杂度是会达到 O(N^2)。而且题目要求我们原地修改，也就是说不能用辅助数组，空间复杂度得是 O(1)。

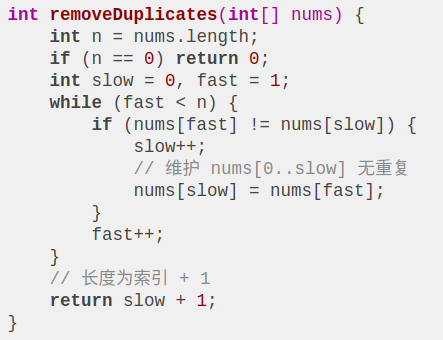
其实，**对于数组相关的算法问题，有一个通用的技巧：要尽量避免在中间删除元素，那我就先想办法把这个元素换到最后去**。

这样的话，最终待删除的元素都拖在数组尾部，一个一个 pop 掉就行了，每次操作的时间复杂度也就降低到 O(1) 了。

按照这个思路呢，又可以衍生出解决类似需求的通用方式：双指针技巧。具体一点说，应该是快慢指针。

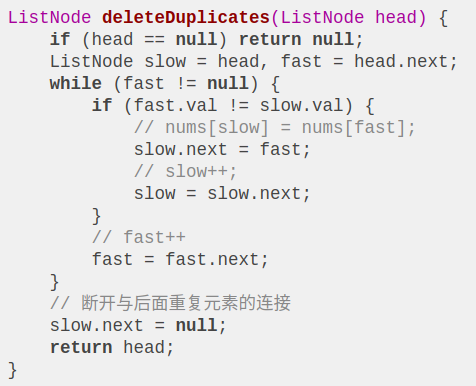
我们让慢指针slow走左后面，快指针fast走在前面探路，找到一个不重复的元素就告诉slow并让slow前进一步。

这样当fast指针遍历完整个数组nums后，**nums[0..slow]就是不重复元素，之后的所有元素都是重复元素**。



看下算法执行的过程（见当前目录gif文件）：

再简单扩展一下，如果给你一个有序链表，如何去重呢？其实和数组是一模一样的，唯一的区别是把数组赋值操作变成操作指针而已：



对于链表去重，算法执行的过程是这样的（见当前目录gif文件）：

最后，近期准备写写一些简单实用的数组/链表技巧。那些稍困难的技巧（比如动态规划）虽然秀，但毕竟在现实生活中不容易遇到。恰恰是一些简单常用的技巧，能够在不经意间，让人发现你是个高手 ^\_^。