1. 两数和

查表考虑哈希表

4、 寻找两个数组的中位数

如何把时间复杂度降低到 O(\log(m+n))O(log(m+n)) 呢？如果对时间复杂度的要求有 \log log，通常都需要用到二分查找，这道题也可以通过二分查找实现。

8、字符串转换整数

（3）在判断一个数是否不在环境的存储范围时，不能计算好再判断，这样会可能越界，导致判断错误。如果当前的值是正数，应该在加法之前先判断当前的结果是否已经大于INT\_MAX/10，如果大于，当前值乘10一定会越界，负数同样的道理。 （4）以正数为例，进行步骤（3）的判断后还不完整，如果在加法之前判断当前的结果等于INT\_MAX/10，那么要判断接下来要读到的数是否大于INT\_MAX的最后一位数，如果大于，直接返回INT\_MAX。

9、回文数

回文数的判断可以反转一半

[10. 正则表达式匹配](https://leetcode-cn.com/problems/regular-expression-matching/)

其实是个DP， 对可以匹配0-无数次末尾字符的符号“\*”构建状态转移方程

19 删除列表最后的第n个节点

方法二：一次遍历算法

算法

上述算法可以优化为只使用一次遍历。我们可以使用两个指针而不是一个指针。第一个指针从列表的开头向前移动 n+1 步，而第二个指针将从列表的开头出发。现在，这两个指针被 nn 个结点分开。我们通过同时移动两个指针向前来保持这个恒定的间隔，直到第一个指针到达最后一个结点。此时第二个指针将指向从最后一个结点数起的第 n 个结点。我们重新链接第二个指针所引用的结点的 next 指针指向该结点的下下个结点。

[23. 合并K个排序链表](https://leetcode-cn.com/problems/merge-k-sorted-lists/)

 这个问题在面试中常常出现，为了达到空间代价是 O(1)，我们的宗旨是「原地调整链表元素的 next 指针完成合并」

优先队列 分治

24 交换链表节点————递归 O(N), O(N)

迭代 O(N), O(1)

27、移除元素 查找的时候可以用双指针

28、 给定一个 haystack 字符串和一个 needle 字符串，在 haystack 字符串中找出 needle 字符串出现的第一个位置 (从0开始)。如果不存在，则返回  -1。

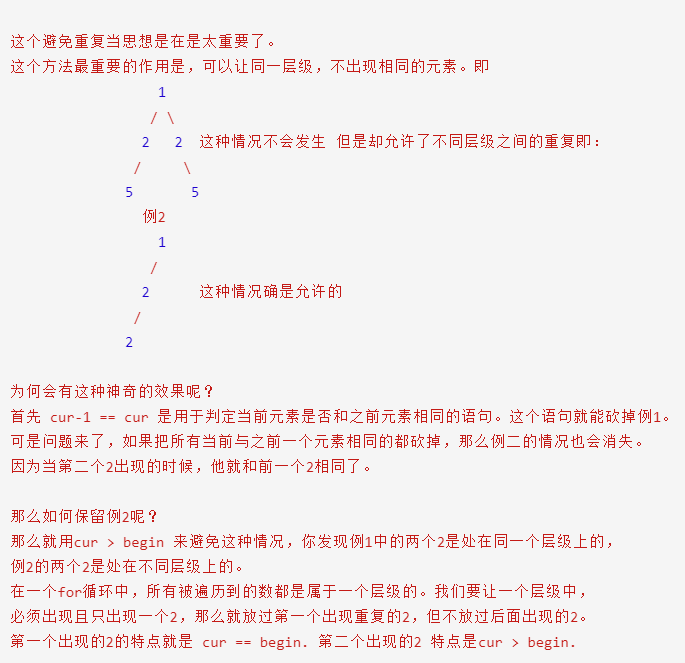
滑动窗口比较：O((N - L)L)

用长度为L的哈希码比对，或者KMP——计算匹配字符的相同前缀，实现文本中指针不回退，回退的是待匹配字符中的状态

32、最长有效括号——栈/DP

39、组合总和——回溯去重（规定搜索的顺序，向前/向后搜索）





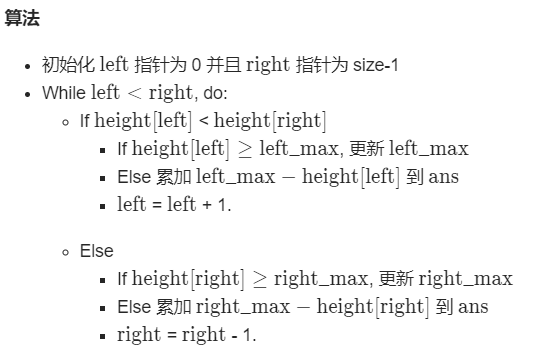
40、缺失的第一个正数

1-N的数组变成哈希

或者方法二：置换

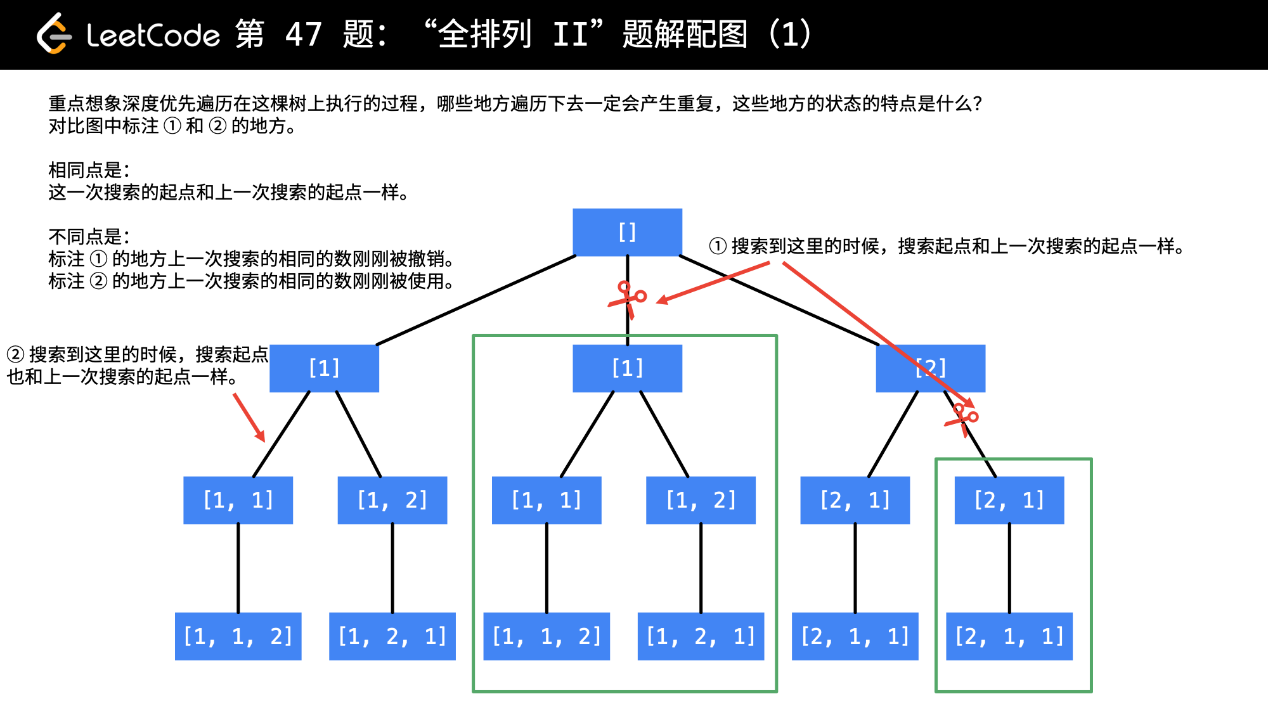
除了打标记以外，我们还可以使用置换的方法，将给定的数组「恢复」，然后寻找“错误”的位置

41、储水： 枚举，DP（存left/right），栈，双指针



46、全排列——回溯

47、全排列不可重复——排序+剪枝



50. Pow(x,n) : 可以用2的n次方做减半运算——结合二进制