# Leetcode题解

## 两数之和

通过减法和map来解决。遍历数组，用target减数组中的每个元素，然后判断元素是否在map中。如果不在则加入map，如果在则是要找的结果。

## **寻找两个有序数组的中位数--重做**

将两个数组分成左右两个部分，保证len(left) == len(right),max(left)<min(right)。

## 最长回文字符串

两个指针，一个指向字符串头部，一个指向字符串尾部。

两个for循环，第一层for循环尾指针向头移动，第二层for循环头指针向尾指针移动，判断头尾指针之间的字符串是否回文字符串，是则找到结果。

## 字符串转换整数

以非数字和‘+’“-”“ ”开头的都是非法的。

找到数字之前先判断正负号和空格，根据题目描述来判断是否合法。

对于数字的处理就是要判断是否超过Int32的范围。

判断方法对MaxInt32/10和MAXInt32%10，分别判断。

## 最长公共前缀

以字符串数组第一个成员作为标准，遍历其他字符串成员里的每个字符，找到公共前缀。

## 三数之和

先对数组进行排序。从数组第一个元素开始遍历数组到第len-3个数。

然后头尾指针，头指针从i+1开始，尾指针从len-1开始。计算a+b+c，由于数组是排序的。

如果三数之和比预期结果大，那么尾指针向头移动，如果和比预期结果要小，则头指针向尾移动。

把所有满足条件的结果保存下来即可。

## 最接近的三数之和

先数组排序，然后就是常规的三层循环遍历数组找到最接近的三数之和，需要注意的就是循环的时候要过滤掉相同的元素以提高循环效率。

## 有效的括号

用一个数组来保存左括号，碰到右括号的时候，就检查数组最后一个元素是不是对应左括号，是则将其移出数组，不是则直接返回错误。

## 删除排序数组的重复项

定义i、j，i从0开始，j从1开始，j遍历数组的所有成员，i则只有当array[i]!=array[j]的时候才才向前移动同时array[j] = array[i]。

## 盛水最多的容器

两个下标分别指向数组头尾。由于容器的大小是由最短高度来决定的。如果头下标的对应的高度小于尾下标对应的高度，那么头下标向尾下标移动。反过来尾下标到往头下标移动，直到相遇。

## 字符串相乘

模仿乘法运算的做法。创建一个字符数组用来保存结果。数组的长度就是两个字符串长度相加。

从字符串的最后一位开始算起，结果填入数组的最后一位。计算的过程要考虑到乘法进位和加法进位。

最后计算完成之后还需要把前面多于0去掉。

## 反转字符串

头尾指针对调。

## 反转字符串中的单词 IIII

调用splits的函数将字符串中单词提取出来，反转之后再用“ ”进行拼接。

## 除自身以外的乘积

从头到尾连乘，然后从尾到头连乘，把结果保存在数组对应的位置。除自身以外的乘积就是当前位置左边的元素乘右边的元素。

## 存在重复元素

哈希表解决。

## 螺旋矩阵

分四个循环，向右，向下，向左，向上。每个循环找到自己的边界即可。

## 螺旋矩阵II

跟上面的方法是一样的。

## 合并两个有序数组

由于题目中说明nums1有足够的空间保存nums2的元素，从后往前比较nums1和nums2的元素，将大的存入nums1的尾部。直到全部保存完成。

## 反转链表--重做

使用递归，找到数组末尾的元素，然后将next指向前一个节点即可。

## 两数相加

处理好进位即可。

## 合并两个有序链表

同时从头部遍历两个两链表，调整好顺序即可。

## 合并K个排序链表

思路和上题一样，每次都遍历所有链表的头部节点，找到最小的那一个，然后组成链表即可。

## 旋转链表

找到旋转右移次数和链表长度的关系，旋转多少次之后，链表会回到初始的样子。

## 环形链表

快慢指针，都从头结点开始，快指针一次走两步，慢指针一次走一步。如果有环，快指针一定会和慢指针重叠。

## 环形链表II

还是快慢指针，用上题的办法判断是否有环。如果有环，在两个指针相遇的时候，把慢指针重新指向链表头部。等到两个指针再次相遇的时候，就是环的起点。

## 相交链表

两个链表一长一短，相交部分肯定是一样长的。那么两个指针同时从两个链表的头部开始遍历。等到短链表指针走到链末尾的时候，长链表的指针剩余的距离刚好是两个链表的差值。

这时候引入第三个指针指向长链表的头部，然后和第一个长链表指针同时运动，等到第一个长链表指针走到链表末尾的时候，短链表指针从短链表头部开始于第三个指针一起运动，当两个指针相遇的时候，就是链表相交的地方。

## 删除链表中的节点

这题不能删除链表的尾节点。node.Val = node.next.Val; node.next = node.next.next即可。

## 整数反转--重做

直接%10和/10就行，就是要判断反转之后是否越界。

## 回文数--重做

所有负数直接out，正数判断高位半边和低位半边是否相等即可，不要把整数反转之后再判断，防止越界。

## 只出现一次的数字--重做

遍历整个数组，用0异或^数组中的所有元素，最后得到的结果，就是只出现了一次的数字。

## 求众数--重做

数组排序之后，直接取中间的那个成员就是结果。效率不一定最高，但是最简单。

或者用map计数，某个成员的计数超过len/2的时候就是结果。

## 2的次幂

1左移，然后跟target比较，如果相等就是，如果不等且1左移的值已经大于target那就不是。