# 剑指 offer 思路总结

本部分主要是笔者在练习剑指 offer 时所做的笔记，如果出现错误，希望大家指出！

## 题目

1. 二维数组中的查找

题目：  
在一个二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的  
一个二维数组和一个整数，判断数组中是否含有该整数。

思路：

（1）第一种方式是使用两层循环依次遍历，判断是否含有该整数。这一种方式最坏情况下的时间复杂度为 O(n^2)。

（2）第二种方式是利用递增序列的特点，我们可以从二维数组的右上角开始遍历。如果当前数值比所求的数要小，则将位置向下移动 ，再进行判断。如果当前数值比所求的数要大，则将位置向左移动，再进行判断。这一种方式最坏情况下的时间复杂度为 O(n)。

1. 替换空格

题目：  
  
请实现一个函数，将一个字符串中的空格替换成“%20”。例如，当字符串为 We Are Happy.则经过替换之后的字符串为 We%20  
Are%20Happy  
  
思路：  
  
使用正则表达式，结合字符串的 replace 方法将空格替换为 “%20”  
  
str.replace(/\s/g,"%20")

1. 从尾到头打印链表

题目：  
输入一个链表，从尾到头打印链表每个节点的值。  
  
思路：  
  
利用栈来实现，首先根据头结点以此遍历链表节点，将节点加入到栈中。当遍历完成后，再将栈中元素弹出并打印，以此来实现。栈的  
实现可以利用 Array 的 push 和 pop 方法来模拟。

1. 重建二叉树

题目：  
  
输入某二叉树的前序遍历和中序遍历的结果，请重建出该二叉树。假设输入的前序遍历和中序遍历的结果中都不含重复的数字。例如输  
入前序遍历序列 {1,2,4,7,3,5,6,8} 和中序遍历序列 {4,7,2,1,5,3,8,6}，则重建二叉树并返回。  
  
思路：  
  
利用递归的思想来求解，首先先序序列中的第一个元素一定是根元素。然后我们去中序遍历中寻找到该元素的位置，找到后该元素的左  
边部分就是根节点的左子树，右边部分就是根节点的右子树。因此我们可以分别截取对应的部分进行子树的递归构建。使用这种方式的  
时间复杂度为 O(n)，空间复杂度为 O(logn)。

1. 用两个栈实现队列

题目：  
  
用两个栈来实现一个队列，完成队列的 Push 和 Pop 操作。  
  
思路：  
  
队列的一个基本特点是，元素先进先出。通过两个栈来模拟时，首先我们将两个栈分为栈 1 和栈 2。当执行队列的 push 操作时，直接  
将元素 push 进栈 1 中。当队列执行 pop 操作时，首先判断栈 2 是否为空，如果不为空则直接 pop 元素。如果栈 2 为空，则将栈 1 中  
的所有元素 pop 然后 push 到栈 2 中，然后再执行栈 2 的 pop 操作。  
  
扩展：  
  
当使用两个长度不同的栈来模拟队列时，队列的最大长度为较短栈的长度的两倍。

1. 旋转数组的最小数字

题目：  
  
把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。 输入一个非递减排序的数组的一个旋转，输出旋转数组的  
最小元素。 例如数组{3,4,5,1,2}为{1,2,3,4,5}的一个旋转，该数组的最小值为 1。 NOTE：给出的所有元素都大于 0，若数组大  
小为 0，请返回 0。  
  
思路：  
  
（1）我们输入的是一个非递减排序的数组的一个旋转，因此原始数组的值递增或者有重复。旋转之后原始数组的值一定和一个值相  
邻，并且不满足递增关系。因此我们就可以进行遍历，找到不满足递增关系的一对值，后一个值就是旋转数组的最小数字。  
  
（2）二分法

相关资料可以参考： [《旋转数组的最小数字》](https://www.cnblogs.com/edisonchou/p/4746561.html)

1. 斐波那契数列

题目：  
  
大家都知道斐波那契数列，现在要求输入一个整数 n，请你输出斐波那契数列的第 n 项。 n<=39  
  
思路：  
  
斐波那契数列的规律是，第一项为 0，第二项为 1，第三项以后的值都等于前面两项的和，因此我们可以通过循环的方式，不断通过叠  
加来实现第 n 项值的构建。通过循环而不是递归的方式来实现，时间复杂度降为了 O(n)，空间复杂度为 O(1)。

1. 跳台阶

题目：  
  
一只青蛙一次可以跳上 1 级台阶，也可以跳上 2 级。求该青蛙跳上一个 n 级的台阶总共有多少种跳法。  
  
思路：  
  
跳台阶的问题是一个动态规划的问题，由于一次只能够跳 1 级或者 2 级，因此跳上 n 级台阶一共有两种方案，一种是从 n-1 跳上，一  
种是从 n-2 级跳上，因此 f(n) = f(n-1) + f(n-2)。  
  
和斐波那契数列类似，不过初始两项的值变为了 1 和 2，后面每项的值等于前面两项的和。

1. 变态跳台阶

题目：

一只青蛙一次可以跳上 1 级台阶，也可以跳上 2 级……它也可以跳上 n 级。求该青蛙跳上一个 n 级的台阶总共有多少种跳法。

思路：

变态跳台阶的问题同上一个问题的思考方案是一样的，我们可以得到一个结论是，每一项的值都等于前面所有项的值的和。

f(1) = 1 f(2) = f(2-1) + f(2-2) //f(2-2) 表示 2 阶一次跳 2 阶的次数。 f(3) = f(3-1) + f(3-2) + f(3-3) ... f(n) = f(n-1) + f(n-2) + f(n-3) + ... + f(n-(n-1)) + f(n-n)

再次总结可得

| 1 ,(n=0 )  
f(n) = | 1 ,(n=1 )  
 | 2\\*f(n-1),(n>=2)

1. 矩形覆盖

题目：  
  
我们可以用 2\*1 的小矩形横着或者竖着去覆盖更大的矩形。请问用 n 个 2\*1 的小矩形无重叠地覆盖一个 2\\*n 的大矩形，总共  
有多少种方法？  
  
思路：  
  
依旧是斐波那契数列的应用

1. 二进制中 1 的个数

题目：  
  
输入一个整数，输出该数二进制表示中 1 的个数。其中负数用补码表示。  
  
思路：  
  
一个不为 0 的整数的二进制表示，一定会有一位为 1。我们找到最右边的一位 1，当我们将整数减去 1 时，最右边的一位 1 变为 0，它后  
面的所有位都取反，因此将减一后的值与原值相与，我们就会能够消除最右边的一位 1。因此判断一个二进制中 1 的个数，我们可以判  
断这个数可以经历多少次这样的过程。  
  
如：1100&1011=1000

1. 数值的整数次方

题目：  
  
给定一个 double 类型的浮点数 base 和 int 类型的整数 exponent。求 base 的 exponent 次方。  
  
思路：  
  
首先我们需要判断 exponent 正负和零取值三种情况，根据不同的情况通过递归来实现。

1. 调整数组顺序使奇数位于偶数前面

题目：  
  
输入一个整数数组，实现一个函数来调整该数组中数字的顺序，使得所有的奇数位于数组的前半部分，所有的偶数位于位于数组的后半  
部分，并保证奇数和奇数，偶数和偶数之间的相对位置不变。  
  
思路：  
  
由于需要考虑到调整之后的稳定性，因此我们可以使用辅助数组的方式。首先对数组中的元素进行遍历，每遇到一个奇数就将它加入到  
奇数辅助数组中，每遇到一个偶数，就将它将入到偶数辅助数组中。最后再将两个数组合并。这一种方法的时间复杂度为 O(n)，空间  
复杂度为 O(n)。

1. 链表中倒数第 k 个节点

题目：  
  
输入一个链表，输出该链表中倒数第 k 个结点。  
  
思路：  
  
使用两个指针，先让第一个和第二个指针都指向头结点，然后再让第二个指针走 k-1 步，到达第 k 个节点。然后两个指针同时向后  
移动，当第二个指针到达末尾时，第一个指针指向的就是倒数第 k 个节点了。

1. 反转链表

题目：  
  
输入一个链表，反转链表后，输出链表的所有元素。  
  
思路：  
  
通过设置三个变量 pre、current 和 next，分别用来保存前继节点、当前节点和后继结点。从第一个节点开始向后遍历，首先将当  
前节点的后继节点保存到 next 中，然后将当前节点的后继节点设置为 pre，然后再将 pre 设置为当前节点，current 设置为 ne  
xt 节点，实现下一次循环。

1. 合并两个排序的链表

题目：  
  
输入两个单调递增的链表，输出两个链表合成后的链表，当然我们需要合成后的链表满足单调不减规则。  
  
思路：  
  
通过递归的方式，依次将两个链表的元素递归进行对比。

1. 树的子结构

题目：  
  
输入两棵二叉树 A、B，判断 B 是不是 A 的子结构。（ps：我们约定空树不是任意一个树的子结构）  
  
思路：  
  
通过递归的思想来解决  
  
第一步首先从树 A 的根节点开始遍历，在左右子树中找到和树 B 根结点的值一样的结点 R 。  
第二步两棵树同时从 R 节点和根节点以相同的遍历方式进行遍历，依次比较对应的值是否相同，当树 B 遍历结束时，结束比较。

1. 二叉树的镜像

题目：  
  
操作给定的二叉树，将其变换为源二叉树的镜像。  
  
思路：  
  
从根节点开始遍历，首先通过临时变量保存左子树的引用，然后将根节点的左右子树的引用交换。然后再递归左右节点的子树交换。

1. 顺时针打印矩阵

题目：  
  
输入一个矩阵，按照从外向里以顺时针的顺序依次打印出每一个数字，  
例如，如果输入如下矩阵： 1 2 3 4  
5 6 7 8  
9 10 11 12  
13 14 15 16  
则依次打印出数字 1,2,3,4,8,12,16,15,14,13,9,5,6,7,11,10  
  
思路：  
  
（1）根据左上角和右下角可以定位出一次要旋转打印的数据。一次旋转打印结束后，往对角分别前进和后退一个单位，可以确定下一  
次需要打印的数据范围。  
  
（2）使用模拟魔方逆时针解法，每打印一行，则将矩阵逆时针旋转 90 度，打印下一行，依次重复。

1. 定义一个栈，实现 min 函数

题目：  
  
定义栈的数据结构，请在该类型中实现一个能够得到栈最小元素的 min 函数。  
  
思路：  
  
使用一个辅助栈，每次将数据压入数据栈时，就把当前栈里面最小的值压入辅助栈当中。这样辅助栈的栈顶数据一直是数据栈中最小  
的值。

1. 栈的压入弹出

题目：  
  
输入两个整数序列，第一个序列表示栈的压入顺序，请判断第二个序列是否为该栈的弹出顺序。假设压入栈的所有数字均不相等。例如  
序列 1,2,3,4,5 是某栈的压入顺序，序列 4,5,3,2,1 是该压栈序列对应的一个弹出序列，但 4,3,5,1,2 就不可能是该压栈序  
列的弹出序列。（注意：这两个序列的长度是相等的）  
  
思路：  
  
我们可以使用一个辅助栈的方式来实现，首先遍历压栈顺序，依次将元素压入辅助栈中，每次压入元素后我们首先判断该元素是否与出  
栈顺序中的此刻位置的元素相等，如果不相等，则将元素继续压栈，如果相等，则将辅助栈中的栈顶元素出栈，出栈后，将出栈顺序中  
的位置后移一位继续比较。当压栈顺序遍历完成后，如果辅助栈不为空，则说明该出栈顺序不正确。

1. 从上往下打印二叉树

题目：  
  
从上往下打印出二叉树的每个节点，同层节点从左至右打印。  
  
思路：  
  
本质上是二叉树的层序遍历，可以通过队列来实现。首先将根节点入队。然后对队列进行出队操作，每次出队时，将出队元素的左右子  
节点依次加入到队列中，直到队列长度变为 0 时，结束遍历。

1. 二叉搜索树的后序遍历

题目：  
  
输入一个整数数组，判断该数组是不是某二叉搜索树的后序遍历的结果。如果是则输出 Yes，否则输出 No。假设输入的数组的任意两  
个数字都互不相同。  
  
思路：  
  
对于一个合法而二叉树的后序遍历来说，最末尾的元素为根元素。该元素前面的元素可以划分为两个部分，一部分为该元素的左子树，  
所有元素的值比根元素小，一部分为该元素的右子树，所有的元素的值比该根元素大。并且每一部分都是一个合法的后序序列，因此我  
们可以利用这些特点来递归判断。

1. 二叉树中和为某一值路径

题目：  
  
输入一颗二叉树和一个整数，打印出二叉树中结点值的和为输入整数的所有路径。路径定义为从树的根结点开始往下一直到叶结点所经  
过的结点形成一条路径。  
  
思路：  
  
通过对树进行深度优先遍历，遍历时保存当前节点的值并判断是否和期望值相等，如果遍历到叶节点不符合要求则回退处理。

1. 复杂链表的复制

题目：  
  
输入一个复杂链表（每个节点中有节点值，以及两个指针，一个指向下一个节点，另一个特殊指针指向任意一个节点），返回结果为  
复制后复杂链表的 head。（注意，输出结果中请不要返回参数中的节点引用，否则判题程序会直接返回空）  
  
思路：  
  
（1）第一种方式，首先对原有链表每个节点进行复制，通过 next 连接起来。然后当链表复制完成之后，再来设置每个节点的 ra  
ndom 指针，这个时候每个节点的 random 的设置都需要从头结点开始遍历，因此时间的复杂度为 O(n^2)。  
  
（2）第二种方式，首先对原有链表每个节点进行复制，并且使用 Map 以键值对的方式将原有节点和复制节点保存下来。当链表复  
制完成之后，再来设置每个节点的 random 指针，这个时候我们通过 Map 中的键值关系就可以获取到对应的复制节点，因此  
不必再从头结点遍历，将时间的复杂度降低为了 O(n)，但是空间复杂度变为了 O(n)。这是一种以空间换时间的做法。  
  
（3）第三种方式，首先对原有链表的每个节点进行复制，并将复制后的节点加入到原有节点的后面。当链表复制完成之后，再进行  
random 指针的设置，由于每个节点后面都跟着自己的复制节点，因此我们可以很容易的获取到 random 指向对应的复制节点  
。最后再将链表分离，通过这种方法我们也能够将时间复杂度降低为 O(n)。

1. 二叉搜索树与双向链表

题目：  
  
输入一棵二叉搜索树，将该二叉搜索树转换成一个排序的双向链表。要求不能创建任何新的结点，只能调整树中结点指针的指向。  
  
思路：  
  
需要生成一个排序的双向列表，那么我们应该通过中序遍历的方式来调整树结构，因为只有中序遍历，返回才是一个从小到大的排序  
序列。  
  
基本的思路是我们首先从根节点开始遍历，先将左子树调整为一个双向链表，并将左子树双向链表的末尾元素的指针指向根节点，并  
将根节点的左节点指向末尾节点。再将右子树调整为一个双向链表，并将右子树双向链表的首部元素的指针指向根元素，再将根节点  
的右节点指向首部节点。通过对左右子树递归调整，因此来实现排序的双向链表的构建。

1. 字符串的排列

题目：  
  
输入一个字符串，按字典序打印出该字符串中字符的所有排列。例如输入字符串 abc，则打印出由字符 a,b,c 所能排列出来的所有  
字符串 abc,acb,bac,bca,cab 和 cba。输入描述：输入一个字符串，长度不超过 9（可能有字符重复），字符只包括大小写字母。  
  
思路：  
  
我们可以把一个字符串看做是两个部分，第一部分为它的第一个字符，第二部分是它后面的所有字符。求整个字符串的一个全排列，可  
以看做两步，第一步是求所有可能出现在第一个位置的字符，即把第一个字符和后面的所有字符交换。第二步就是求后面所有字符的一  
个全排列。因此通过这种方式，我们可以以递归的思路来求出当前字符串的全排列。

详细资料可以参考： [《字符串的排列》](https://wiki.jikexueyuan.com/project/for-offer/question-twenty-eight.html)

1. 数组中出现次数超过一半的数字

题目：  
  
数组中有一个数字出现的次数超过数组长度的一半。请找出这个数字。例如输入一个长度为 9 的数组{1,2,3,2,2,2,5,4,2}。由于数  
字 2 在数组中出现了 5 次，超过数组长度的一半，因此输出 2。如果不存在则输出 0。  
  
思路：  
  
（1）对数组进行排序，排序后的中位数就是所求数字。这种方法的时间复杂度取决于我们采用的排序方法的时间复杂度，因此最快为  
O(nlogn)。  
  
（2）由于所求数字的数量超过了数组长度的一半，因此排序后的中位数就是所求数字。因此我们可以将问题简化为求一个数组的中  
位数问题。其实数组并不需要全排序，只需要部分排序。我们通过利用快排中的 partition 函数来实现，我们现在数组中随  
机选取一个数字，而后通过 partition 函数返回该数字在数组中的索引 index，如果 index 刚好等于 n/2，则这个数字  
便是数组的中位数，也即是要求的数，如果 index 大于 n/2，则中位数肯定在 index 的左边，在左边继续寻找即可，反之  
在右边寻找。这样可以只在 index 的一边寻找，而不用两边都排序，减少了一半排序时间，这种方法的时间复杂度为 O(n)。  
  
（3）由于该数字的出现次数比所有其他数字出现次数的和还要多，因此可以考虑在遍历数组时保存两个值：一个是数组中的一个数  
字，一个是次数。当遍历到下一个数字时，如果下一个数字与之前保存的数字相同，则次数加 1，如果不同，则次数减 1，如果  
次数为 0，则需要保存下一个数字，并把次数设定为 1。由于我们要找的数字出现的次数比其他所有数字的出现次数之和还要大，  
则要找的数字肯定是最后一次把次数设为 1 时对应的数字。该方法的时间复杂度为 O(n)，空间复杂度为 O(1)。

详细资料可以参考： [《出现次数超过一半的数字》](https://blog.csdn.net/ns_code/article/details/26957383)

1. 最小的 K 个数

题目：  
  
输入 n 个整数，找出其中最小的 K 个数。例如输入 4,5,1,6,2,7,3,8 这 8 个数字，则最小的 4 个数字是 1,2,3,4 。  
  
思路：  
  
（1）第一种思路是首先将数组排序，排序后再取最小的 k 个数。这一种方法的时间复杂度取决于我们选择的排序算法的时间复杂  
度，最好的情况下为 O(nlogn)。  
  
（2）第二种思路是由于我们只需要获得最小的 k 个数，这 k 个数不一定是按序排序的。因此我们可以使用快速排序中的 part  
ition 函数来实现。每一次选择一个枢纽值，将数组分为比枢纽值大和比枢纽值小的两个部分，判断枢纽值的位置，如果该枢  
纽值的位置为 k-1 的话，那么枢纽值和它前面的所有数字就是最小的 k 个数。如果枢纽值的位置小于 k-1 的话，假设枢  
纽值的位置为 n-1，那么我们已经找到了前 n 小的数字了，我们就还需要到后半部分去寻找后半部分 k-n 小的值，进行划  
分。当该枢纽值的位置比 k-1 大时，说明最小的 k 个值还在左半部分，我们需要继续对左半部分进行划分。这一种方法的平  
均时间复杂度为 O(n)。  
  
（3）第三种方法是维护一个容量为 k 的最大堆。对数组进行遍历时，如果堆的容量还没有达到 k ，则直接将元素加入到堆中，这  
就相当于我们假设前 k 个数就是最小的 k 个数。对 k 以后的元素遍历时，我们将该元素与堆的最大值进行比较，如果比最  
大值小，那么我们则将最大值与其交换，然后调整堆。如果大于等于堆的最大值，则继续向后遍历，直到数组遍历完成。这一  
种方法的平均时间复杂度为 O(nlogk)。

详细资料可以参考： [《寻找最小的 k 个数》](https://www.kancloud.cn/kancloud/the-art-of-programming/41579)

1. 连续子数组的最大和

题目：  
  
HZ 偶尔会拿些专业问题来忽悠那些非计算机专业的同学。今天测试组开完会后，他又发话了:在古老的一维模式识别中，常常需要计  
算连续子向量的最大和,当向量全为正数的时候，问题很好解决。但是，如果向量中包含负数，是否应该包含某个负数，并期望旁边的  
正数会弥补它呢？例如：{6,-3,-2,7,-15,1,2,2}，连续子向量的最大和为 8（从第 0 个开始，到第 3 个为止）。你会不会被他忽悠  
住？（子向量的长度至少是 1）  
  
思路：  
  
（1）第一种思路是直接暴力求解的方式，先以第一个数字为首往后开始叠加，叠加的过程中保存最大的值。然后再以第二个数字为首  
往后开始叠加，并与先前保存的最大的值进行比较。这一种方法的时间复杂度为 O(n^2)。  
  
（2）第二种思路是，首先我们观察一个最大和的连续数组的规律，我们可以发现，子数组一定是以正数开头的，中间包含了正负数。  
因此我们可以从第一个数开始向后叠加，每次保存最大的值。叠加的值如果为负数，则将叠加值初始化为 0，因为后面的数加上负  
数只会更小，因此需要寻找下一个正数开始下一个子数组的判断。一直往后判断，直到这个数组遍历完成为止，得到最大的值。  
使用这一种方法的时间复杂度为 O(n)。

详细资料可以参考： [《连续子数组的最大和》](http://wiki.jikexueyuan.com/project/for-offer/question-thirty-one.html)

1. 整数中 1 出现的次数（待深入理解）

题目：  
  
求出 1~13 的整数中 1 出现的次数，并算出 100~1300 的整数中 1 出现的次数？为此他特别数了一下 1~13 中包含 1 的数字有 1、10、11、  
12、13 因此共出现 6 次，但是对于后面问题他就没辙了。ACMer 希望你们帮帮他，并把问题更加普遍化，可以很快的求出任意非负整  
数区间中 1 出现的次数。  
  
思路：  
  
（1）第一种思路是直接遍历每个数，然后将判断每个数中 1 的个数，一直叠加。  
  
（2）第二种思路是求出 1 出现在每位上的次数，然后进行叠加。

详细资料可以参考： [《从 1 到 n 整数中 1 出现的次数：O(logn)算法》](https://blog.csdn.net/yi_Afly/article/details/52012593)

1. 把数组排成最小的数

题目：  
  
输入一个正整数数组，把数组里所有数字拼接起来排成一个数，打印能拼接出的所有数字中最小的一个。例如输入数组{3，32，321  
}，则打印出这三个数字能排成的最小数字为 321323。  
  
思路：  
  
（1）求出数组的全排列，然后对每个排列结果进行比较。  
  
（2）利用排序算法实现，但是比较时，比较的并不是两个元素的大小，而是两个元素正序拼接和逆序拼接的大小，如果逆序拼接的  
结果更小，则交换两个元素的位置。排序结束后，数组的顺序则为最小数的排列组合顺序。

详细资料可以参考： [《把数组排成最小的数》](http://wiki.jikexueyuan.com/project/for-offer/question-thirty-three.html)

1. 丑数（待深入理解）

题目：  
  
把只包含质因子 2、3 和 5 的数称作丑数。例如 6、8 都是丑数，但 14 不是，因为它包含因子 7。 习惯上我们把 1 当做是第一个丑数。求  
按从小到大的顺序的第 N 个丑数。  
  
思路：  
  
（1）判断一个数是否为丑数，可以判断该数不断除以 2，最后余数是否为 1。判断该数不断除以 3，最后余数是否为 1。判断不断除以  
5，最后余数是否为 1。在不考虑时间复杂度的情况下，可以依次遍历找到第 N 个丑数。  
  
（2）使用一个数组来保存已排序好的丑数，后面的丑数由前面生成。

1. 第一个只出现一次的字符

题目：  
  
在一个字符串（1<=字符串长度<=10000，全部由大写字母组成）中找到第一个只出现一次的字符，并返回它的位置。  
  
思路：  
  
（1）第一种思路是，从前往后遍历每一个字符。每遍历一个字符，则将字符与后边的所有字符依次比较，判断是否含有相同字符。这  
一种方法的时间复杂度为 O(n^2)。  
  
（2）第二种思路是，首先对字符串进行一次遍历，将字符和字符出现的次数以键值对的形式存储在 Map 结构中。然后第二次遍历时  
，去 Map 中获取对应字符出现的次数，找到第一个只出现一次的字符。这一种方法的时间复杂度为 O(n)。

1. 数组中的逆序对

题目：  
  
在数组中的两个数字，如果前面一个数字大于后面的数字，则这两个数字组成一个逆序对。输入一个数组，求出这个数组中的逆序对  
的总数 P。  
  
思路：  
  
（1）第一种思路是直接求解的方式，顺序扫描整个数组。每扫描到一个数字的时候，逐个比较该数字和它后面的数字的大小。如果  
后面的数字比它小，则这两个数字就组成了一个逆序对。假设数组中含有 n 个数字。由于每个数字都要和 O(n）个数字作比  
较，因此这个算法的时间复杂度是 O(n^2)。  
  
（2）第二种方式是使用归并排序的方式，通过利用归并排序分解后进行合并排序时，来进行逆序对的统计，这一种方法的时间复杂  
度为 O(nlogn)。

详细资料可以参考： [《数组中的逆序对》](http://wiki.jikexueyuan.com/project/for-offer/question-thirty-six.html)

1. 两个链表的第一个公共结点

题目：  
  
输入两个链表，找出它们的第一个公共结点。  
  
思路：  
  
（1）第一种方法是在第一个链表上顺序遍历每个结点，每遍历到一个结点的时候，在第二个链表上顺序遍历每个结点。如果在第二  
个链表上有一个结点和第一个链表上的结点一样，说明两个链表在这个结点上重合，于是就找到了它们的公共结点。如果第一  
个链表的长度为 m，第二个链表的长度为 n。这一种方法的时间复杂度是 O(mn）。  
  
（2）第二种方式是利用栈的方式，通过观察我们可以发现两个链表的公共节点，都位于链表的尾部，以此我们可以分别使用两个栈  
，依次将链表元素入栈。然后在两个栈同时将元素出栈，比较出栈的节点，最后一个相同的节点就是我们要找的公共节点。这  
一种方法的时间复杂度为 O(m+n)，空间复杂度为 O(m+n)。  
  
（3）第三种方式是，首先分别遍历两个链表，得到两个链表的长度。然后得到较长的链表与较短的链表长度的差值。我们使用两个  
指针来分别对两个链表进行遍历，首先将较长链表的指针移动 n 步，n 为两个链表长度的差值，然后两个指针再同时移动，  
判断所指向节点是否为同一节点。这一种方法的时间复杂度为 O(m+n)，相同对于上一种方法不需要额外的空间。

详细资料可以参考： [《两个链表的第一个公共结点》](http://wiki.jikexueyuan.com/project/for-offer/question-thirty-seven.html)

1. 数字在排序数组中出现的次数

题目：  
  
统计一个数字：在排序数组中出现的次数。例如输入排序数组｛ 1, 2, 3, 3, 3, 3, 4, 5｝和数字 3 ，由于 3 在这个数组中出  
现了 4 次，因此输出 4 。  
  
思路：  
  
（1）第一种方法是直接对数组顺序遍历的方式，通过这种方法来统计数字的出现次数。这种方法的时间复杂度为 O(n)。  
  
（2）第二种方法是使用二分查找的方法，由于数组是排序好的数组，因此相同数字是排列在一起的。统计数字出现的次数，我们需要  
去找到该段数字开始和结束的位置，以此来确定数字出现的次数。因此我们可以使用二分查找的方式来确定该数字的开始和结束  
位置。如果我们第一次我们数组的中间值为 k ，如果 k 值比所求值大的话，那么我们下一次只需要判断前面一部分就行了，如  
果 k 值比所求值小的话，那么我们下一次就只需要判断后面一部分就行了。如果 k 值等于所求值的时候，我们则需要判断该值  
是否为开始位置或者结束位置。如果是开始位置，那么我们下一次需要到后半部分去寻找结束位置。如果是结束位置，那么我们  
下一次需要到前半部分去寻找开始位置。如果既不是开始位置也不是结束位置，那么我们就分别到前后两个部分去寻找开始和结  
束位置。这一种方法的平均时间复杂度为 O(logn)。

1. 二叉树的深度

题目：  
  
输入一棵二叉树，求该树的深度。从根结点到叶结点依次经过的结点（含根、叶结点）形成树的一条路径，最长路径的长度为树的深  
度。  
  
思路：  
  
根节点的深度等于左右深度较大值加一，因此可以通过递归遍历来实现。

1. 平衡二叉树

题目：  
  
输入一棵二叉树，判断该二叉树是否是平衡二叉树。  
  
思路：  
  
（1）在遍历树的每个结点的时候，调用函数得到它的左右子树的深度。如果每个结点的左右子树的深度相差都不超过 1 ，那么它  
就是一棵平衡的二叉树。使用这种方法时，节点会被多次遍历，因此会造成效率不高的问题。  
  
（2）在求一个节点的深度时，同时判断它是否平衡。如果不平衡则直接返回 -1，否则返回树高度。如果一个节点的一个子树的深  
度为-1，那么就直接向上返回 -1 ，该树已经是不平衡的了。通过这种方式确保了节点只能够被访问一遍。

1. 数组中只出现一次的数字

题目：  
  
一个整型数组里除了两个数字之外，其他的数字都出现了两次。请写程序找出这两个只出现一次的数字。  
  
思路：  
  
（1）第一种方式是依次遍历数组，记录下数字出现的次数，从而找出两个只出现一次的数字。  
  
（2）第二种方式，根据位运算的异或的性质，我们可以知道两个相同的数字异或等于 0，一个数和 0 异或还是它本身。由于数组中  
的其他数字都是成对出现的，因此我们可以将数组中的所有数依次进行异或运算。如果只有一个数出现一次的话，那么最后剩下  
的就是落单的数字。如果是两个数只出现了一次的话，那么最后剩下的就是这两个数异或的结果。这个结果中的 1 表示的是 A 和  
B 不同的位。我们取异或结果的第一个 1 所在的位数，假如是第 3 位，接着通过比较第三位来将数组分为两组，相同数字一定会  
被分到同一组。分组完成后再按照依次异或的思路，求得剩余数字即为两个只出现一次的数字。

1. 和为 S 的连续正数序列

题目：  
  
小明很喜欢数学，有一天他在做数学作业时，要求计算出 9~16 的和，他马上就写出了正确答案是 100。但是他并不满足于此，他在想究  
竟有多少种连续的正数序列的和为 100（至少包括两个数）。没多久，他就得到另一组连续正数和为 100 的序列：18,19,20,21,22。  
现在把问题交给你，你能不能也很快的找出所有和为 S 的连续正数序列？Good Luck!输出描述：输出所有和为 S 的连续正数序列。序  
列内按照从小至大的顺序，序列间按照开始数字从小到大的顺序。  
  
思路：  
  
维护一个正数序列数组，数组中初始只含有值 1 和 2，然后从 3 依次往后遍历，每遍历到一个元素则将这个元素加入到序列数组中，然后  
判断此时序列数组的和。如果序列数组的和大于所求值，则将第一个元素（最小的元素弹出）。如果序列数组的和小于所求值，则继续  
往后遍历，将元素加入到序列中继续判断。当序列数组的和等于所求值时，打印出此时的正数序列，然后继续往后遍历，寻找下一个连  
续序列，直到数组遍历完成终止。

详细资料可以参考： [《和为 s 的连续正数序列》](http://wiki.jikexueyuan.com/project/for-offer/question-forty-one.html)

1. 和为 S 的两个数字

题目：  
  
输入一个递增排序的数组和一个数字 S，在数组中查找两个数，是的他们的和正好是 S，如果有多对数字的和等于 S，输出两个数  
的乘积最小的。输出描述：对应每个测试案例，输出两个数，小的先输出。  
  
思路：  
  
首先我们通过规律可以发现，和相同的两个数字，两个数字的差值越大，乘积越小。因此我们只需要从数组的首尾开始找到第一对和  
为 s 的数字对进行了。因此我们可以使用双指针的方式，左指针初始指向数组的第一个元素，右指针初始指向数组的最后一个元素  
。然后首先判断两个指针指向的数字的和是否为 s ，如果为 s ，两个指针指向的数字就是我们需要寻找的数字对。如果两数的和  
比 s 小，则将左指针向左移动一位后继续判断。如果两数的和比 s 大，则将右指针向右移动一位后继续判断。

详细资料可以参考： [《和为 S 的字符串》](https://www.cnblogs.com/wuguanglin/p/FindNumbersWithSum.html)

1. 左旋转字符串

题目：  
  
汇编语言中有一种移位指令叫做循环左移（ROL），现在有个简单的任务，就是用字符串模拟这个指令的运算结果。对于一个给定的  
字符序列 S，请你把其循环左移 K 位后的序列输出。例如，字符序列 S=”abcXYZdef”，要求输出循环左移 3 位后的结果，即 “X  
YZdefabc”。是不是很简单？OK，搞定它！  
  
思路：  
  
字符串裁剪后拼接

1. 翻转单词顺序列

题目：  
  
牛客最近来了一个新员工 Fish，每天早晨总是会拿着一本英文杂志，写些句子在本子上。同事 Cat 对 Fish 写的内容颇感兴趣，有  
一天他向 Fish 借来翻看，但却读不懂它的意思。例如，“student. a am I”。后来才意识到，这家伙原来把句子单词的顺序翻转了  
，正确的句子应该是“I am a student.”。Cat 对一一的翻转这些单词顺序可不在行，你能帮助他么？  
  
思路：  
  
通过空格将单词分隔，然后将数组反序后，重新拼接为字符串。

1. 扑克牌的顺子

题目：  
  
LL 今天心情特别好，因为他去买了一副扑克牌，发现里面居然有 2 个大王，2 个小王（一副牌原本是 54 张^\\_^）...他随机从中抽出  
了 5 张牌，想测测自己的手气，看看能不能抽到顺子，如果抽到的话，他决定去买体育彩票，嘿嘿！！“红心 A，黑桃 3，小王，大王  
，方片 5”，“Oh My God!”不是顺子..... LL 不高兴了，他想了想，决定大\小王可以看成任何数字，并且 A 看作 1，J 为 11，  
Q 为 12，K 为 13。上面的 5 张牌就可以变成“1,2,3,4,5”（大小王分别看作 2 和 4），“So Lucky!”。LL 决定去买体育彩票啦。  
现在，要求你使用这幅牌模拟上面的过程，然后告诉我们 LL 的运气如何。为了方便起见，你可以认为大小王是 0。  
  
思路：  
  
首先判断 5 个数字是不是连续的，最直观的方法是把数组排序。值得注意的是，由于 0 可以当成任意数字，我们可以用 0 去补满数  
组中的空缺。如果排序之后的数组不是连续的，即相邻的两个数字相隔若干个数字，但只要我们有足够的。可以补满这两个数字的空  
缺，这个数组实际上还是连续的。  
  
于是我们需要做 3 件事情：首先把数组排序，再统计数组中 0 的个数，最后统计排序之后的数组中相邻数字之间的空缺总数。如  
果空缺的总数小于或者等于 0 的个数，那么这个数组就是连续的：反之则不连续。最后，我们还需要注意一点：如果数组中的非 0  
数字重复出现，则该数组不是连续的。换成扑克牌的描述方式就是如果一副牌里含有对子，则不可能是顺子。

详细资料可以参考： [《扑克牌的顺子》](http://wiki.jikexueyuan.com/project/for-offer/question-forty-four.html)

1. 圆圈中最后剩下的数字（约瑟夫环问题）

题目：  
  
0, 1, … , n-1 这 n 个数字排成一个圈圈，从数字 0 开始每次从圆圏里删除第 m 个数字。求出这个圈圈里剩下的最后一个数  
字。  
  
思路：  
  
（1）使用环形链表进行模拟。  
  
（2）根据规律得出（待深入理解）

详细资料可以参考： [《圆圈中最后剩下的数字》](http://wiki.jikexueyuan.com/project/for-offer/question-forty-five.html)

1. 1+2+3+...+n

题目：  
  
求 1+2+3+...+n，要求不能使用乘除法、for、while、if、else、switch、case 等关键字及条件判断语句（A?B:C）。  
  
思路：  
  
由于不能使用循环语句，因此我们可以通过递归来实现。并且由于不能够使用条件判断运算符，我们可以利用 && 操作符的短路特  
性来实现。

1. 不用加减乘除做加法

题目：  
  
写一个函数，求两个整数之和，要求在函数体内不得使用 ＋、－、×、÷ 四则运算符号。  
  
思路：  
  
通过位运算，递归来实现。

1. 把字符串转换成整数。

题目：  
  
将一个字符串转换成一个整数，要求不能使用字符串转换整数的库函数。数值为 0 或者字符串不是一个合法的数值则返回 0。输入描  
述：输入一个字符串，包括数字字母符号，可以为空。输出描述：如果是合法的数值表达则返回该数字，否则返回 0。  
  
思路：  
  
首先需要进行符号判断，其次我们根据字符串的每位通过减 0 运算转换为整数和，依次根据位数叠加。

1. 数组中重复的数字

题目：  
  
在一个长度为 n 的数组里的所有数字都在 0 到 n-1 的范围内。数组中某些数字是重复的，但不知道有几个数字重复了，也不知  
道每个数字重复了几次。请找出数组中任意一个重复的数字。  
  
思路：  
  
（1）首先将数组排序，排序后再进行判断。这一种方法的时间复杂度为 O(nlogn)。  
  
（2）使用 Map 结构的方式，依次记录下每一个数字出现的次数，从而可以判断是否出现重复数字。这一种方法的时间复杂度为 O  
(n)，空间复杂度为 O(n)。  
  
（3）从数组首部开始遍历，每遍历一个数字，则将该数字和它的下标相比较，如果数字和下标不等，则将该数字和它对应下标的值  
交换。如果对应的下标值上已经是正确的值了，那么说明当前元素是一个重复数字。这一种方法相对于上一种方法来说不需要  
额外的内存空间。

1. 构建乘积数组

题目：  
  
给定一个数组 A[0,1,...,n-1]，请构建一个数组 B[0,1,...,n-1]，其中 B 中的元素 B[i]=A[0]\_A[1]\_...\*A[i-1]\*A  
[i+1]\*...\*A[n-1]。不能使用除法。  
  
思路：  
  
（1） C[i]=A[0]×A[1]×...×A[i-1]=C[i-1]×A[i-1]  
 D[i]=A[i+1]×...×A[n-1]=D[i+1]×A[i+1]  
 B[i]=C[i]×D[i]  
 将乘积分为前后两个部分，分别循环求出后，再进行相乘。  
  
（2）上面的方法需要额外的内存空间，我们可以引入中间变量的方式，来降低空间复杂度。（待深入理解）

详细资料可以参考： [《构建乘积数组》](https://zhuanlan.zhihu.com/p/34804711)

1. 正则表达式的匹配

题目：  
  
请实现一个函数用来匹配包括'.'和'\_'的正则表达式。模式中的字符'.'表示任意一个字符，而'\_'表示它前面的字符可以出现任  
意次（包含 0 次）。 在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"a.a"和"ab\*ac\*a"匹配，  
但是与"aa.a"和"ab\\*a"均不匹配。  
  
思路：  
  
（1）状态机思路（待深入理解）

详细资料可以参考： [《正则表达式匹配》](http://wiki.jikexueyuan.com/project/for-offer/question-fifty-three.html)

1. 表示数值的字符串

题目：  
  
请实现一个函数用来判断字符串是否表示数值（包括整数和小数）。例如，字符串"+100","5e2","-123","3.1416"和"-1E-  
16"都表示数值。 但是"12e","1a3.14","1.2.3","+-5"和"12e+4.3"都不是。、  
  
思路：  
  
利用正则表达式实现

1. 字符流中第一个不重复的字符

题目：  
  
请实现一个函数用来找出字符流中第一个只出现一次的字符。例如，当从字符流中只读出前两个字符 "go" 时，第一个只出现一次  
的字符是 "g" 。当从该字符流中读出前六个字符 "google" 时，第一个只出现一次的字符是 "l"。 输出描述：如果当前字符流  
没有存在出现一次的字符，返回#字符。  
  
思路：  
  
同第 34 题

1. 链表中环的入口结点

题目：  
  
一个链表中包含环，如何找出环的入口结点？  
  
思路：  
  
首先使用快慢指针的方式我们可以判断链表中是否存在环，当快慢指针相遇时，说明链表中存在环。相遇点一定存在于环中，因此我  
们可以从使用一个指针从这个点开始向前移动，每移动一个点，环的长度加一，当指针再次回到这个点的时候，指针走了一圈，因此  
通过这个方法我们可以得到链表中的环的长度，我们将它记为 n 。  
  
然后我们设置两个指针，首先分别指向头结点，然后将一个指针先移动 n 步，然后两个指针再同时移动，当两个指针相遇时，相遇  
点就是环的入口节点。

详细资料可以参考： [《链表中环的入口结点》](http://wiki.jikexueyuan.com/project/for-offer/question-fifty-six.html) [《《剑指 offer》——链表中环的入口结点》](https://blog.csdn.net/shansusu/article/details/50285735)

1. 删除链表中重复的结点

题目：  
  
在一个排序的链表中，存在重复的结点，请删除该链表中重复的结点，重复的结点不保留，返回链表头指针。例如，链表 1->2->3-  
  
> 3->4->4->5 处理后为 1->2->5  
  
思路：  
  
解决这个问题的第一步是确定删除的参数。当然这个函数需要输入待删除链表的头结点。头结点可能与后面的结点重复，也就是说头  
结点也可能被删除，所以在链表头额外添加一个结点。  
  
接下来我们从头遍历整个链表。如果当前结点的值与下一个结点的值相同，那么它们就是重复的结点，都可以被删除。为了保证删除  
之后的链表仍然是相连的而没有中间断开，我们要把当前的前一个结点和后面值比当前结点的值要大的结点相连。我们要确保 prev  
要始终与下一个没有重复的结点连接在一起。

1. 二叉树的下一个结点

题目：  
  
给定一棵二叉树和其中的一个结点，如何找出中序遍历顺序的下一个结点？树中的结点除了有两个分别指向左右子结点的指针以外，  
还有一个指向父节点的指针。  
  
思路：  
  
这个问题我们可以分为三种情况来讨论。  
  
第一种情况，当前节点含有右子树，这种情况下，中序遍历的下一个节点为该节点右子树的最左子节点。因此我们只要从右子节点  
出发，一直沿着左子节点的指针，就能找到下一个节点。  
  
第二种情况是，当前节点不含有右子树，并且当前节点为父节点的左子节点，这种情况下中序遍历的下一个节点为当前节点的父节  
点。  
  
第三种情况是，当前节点不含有右子树，并且当前节点为父节点的右子节点，这种情况下我们沿着父节点一直向上查找，直到找到  
一个节点，该节点为父节点的左子节点。这个左子节点的父节点就是中序遍历的下一个节点。

1. 对称二叉树

题目：  
  
请实现一个函数来判断一棵二叉树是不是对称的。如果一棵二叉树和它的镜像一样，那么它是对称的。  
  
思路：  
  
我们对一颗二叉树进行前序遍历的时候，是先访问左子节点，然后再访问右子节点。因此我们可以定义一种对称的前序遍历的方式  
，就是先访问右子节点，然后再访问左子节点。通过比较两种遍历方式最后的结果是否相同，以此来判断该二叉树是否为对称二叉  
树。

1. 按之字形顺序打印二叉树（待深入理解）

题目：  
  
请实现一个函数按照之字形顺序打印二叉树，即第一行按照从左到右的顺序打印，第二层按照从右到左的顺序打印，即第一行按照  
从左到右的顺序打印，第二层按照从右到左顺序打印，第三行再按照从左到右的顺序打印，其他以此类推。  
  
思路：  
  
按之字形顺序打印二叉树需要两个栈。我们在打印某一行结点时，把下一层的子结点保存到相应的栈里。如果当前打印的是奇数层  
，则先保存左子结点再保存右子结点到一个栈里；如果当前打印的是偶数层，则先保存右子结点再保存左子结点到第二个栈里。每  
一个栈遍历完成后进入下一层循环。

详细资料可以参考： [《按之字形顺序打印二叉树》](https://www.cnblogs.com/wuguanglin/p/Print.html)

1. 从上到下按层打印二叉树，同一层结点从左至右输出。每一层输出一行。

题目：  
  
从上到下按层打印二叉树，同一层的结点按从左到右的顺序打印，每一层打印一行。  
  
思路：  
  
用一个队列来保存将要打印的结点。为了把二叉树的每一行单独打印到一行里，我们需要两个变量：一个变量表示在当前的层中还  
没有打印的结点数，另一个变量表示下一次结点的数目。

1. 序列化二叉树（带深入理解）

题目：  
  
请实现两个函数，分别用来序列化和反序列化二叉树。  
  
思路：  
  
数组模拟

1. 二叉搜索树的第 K 个节点

题目：  
  
给定一颗二叉搜索树，请找出其中的第 k 小的结点。  
  
思路：  
  
对一颗树首先进行中序遍历，在遍历的同时记录已经遍历的节点数，当遍历到第 k 个节点时，这个节点即为第 k 大的节点。

1. 数据流中的中位数（待深入理解）

题目：  
  
如何得到一个数据流中的中位数？如果从数据流中读出奇数个数值，那么中位数就是所有值排序之后位于中间的数值。如果数据  
流中读出偶数个数值，那么中位数就是所有数值排序之后中间两个数的平均值。

1. 滑动窗口中的最大值（待深入理解）

题目：  
  
给定一个数组和滑动窗口的大小，找出所有滑动窗口里数值的最大值。例如，如果输入数组{2,3,4,2,6,2,5,1}及滑动窗口的  
大小 3，那么一共存在 6 个滑动窗口，他们的最大值分别为{4,4,6,6,6,5}； 针对数组{2,3,4,2,6,2,5,1}的滑动窗口有以下  
6 个： {[2,3,4],2,6,2,5,1}， {2,[3,4,2],6,2,5,1}， {2,3,[4,2,6],2,5,1}， {2,3,4,[2,6,2],5,1}， {2  
,3,4,2,[6,2,5],1}， {2,3,4,2,6,[2,5,1]}。  
  
思路：  
  
使用队列的方式模拟

1. 矩阵中的路径（待深入理解）

题目：  
  
请设计一个函数，用来判断在一个矩阵中是否存在一条包含某字符串所有字符的路径。路径可以从矩阵中的任意一个格子开始，每  
一步可以在矩阵中向左，向右，向上，向下移动一个格子。如果一条路径经过了矩阵中的某一个格子，则该路径不能再进入该格子  
。例如 a b c e s f c s a d e e 矩阵中包含一条字符串"bcced"的路径，但是矩阵中不包含"abcb"路径，因为字符串的  
第一个字符 b 占据了矩阵中的第一行第二个格子之后，路径不能再次进入该格子。

1. 机器人的运动范围（待深入理解）

题目：  
  
地上有一个 m 行和 n 列的方格。一个机器人从坐标 0,0 的格子开始移动，每一次只能向左，右，上，下四个方向移动一格，但是不能  
进入行坐标和列坐标的数位之和大于 k 的格子。 例如，当 k 为 18 时，机器人能够进入方格（35,37），因为 3+5+3+7 = 18。但是  
，它不能进入方格（35,38），因为 3+5+3+8 = 19。请问该机器人能够达到多少个格子？

剑指 offer 相关资料可以参考：

* [《剑指 offer 题目练习及思路分析》](https://blog.csdn.net/zzl819954692/article/details/79648054)
* [《JS 版剑指 offer》](https://www.cnblogs.com/wuguanglin/p/code-interview.html)
* [《剑指 Offer 学习心得》](https://wiki.jikexueyuan.com/project/for-offer/)

## 相关算法题

1. 明星问题 ``` 题目：

有 n 个人，其中一个明星和 n-1 个群众，群众都认识明星，明星不认识任何群众，群众和群众之间的认识关系不知道，现有一个 函数 foo(A, B)，若 A 认识 B 返回 true，若 A 不认识 B 返回 false，试设计一种算法找出明星，并给出时间复杂度。

思路：

（1）第一种方法我们可以直接使用双层循环遍历的方式，每一个人都和其他人进行判断，如果一个人谁都不认识，那么他就是明星。 这一种方法的时间复杂度为 O(n^2)。

（2）上一种方法没有充分利用题目所给的条件，其实我们每一次比较，都可以排除一个人的可能。比如如果 A 认识 B，那么说明 A 就不会是明星，因此 A 就可以从数组中移除。如果 A 不认识 B，那么说明 B 不可能是明星，因此 B 就可以从数组中移 除。因此每一次判断都能够减少一个可能性，我们只需要从数组从前往后进行遍历，每次移除一个不可能的人，直到数组中只剩 一人为止，那么这个人就是明星。这一种方法的时间复杂度为 O(n)。 ``` 详细资料可以参考： [《一个明星和 n-1 个群众》](https://www.nowcoder.com/questionTerminal/fd3caff374c945fb9ea5f107016bcd4f)

1. 正负数组求和 ``` 题目：

有两个数组，一个数组里存放的是正整数，另一个数组里存放的是负整数，都是无序的，现在从两个数组里各拿一个，使得它们的和 最接近零。

思路：

（1）首先我们可以对两个数组分别进行排序，正数数组按从小到大排序，负数数组按从大到小排序。排序完成后我们使用两个指针分 别指向两个数组的首部，判断两个指针的和。如果和大于0，则负数指针往后移动一个位置，如果和小于0，则正数指针往后移动 一个位置，每一次记录和的值，和当前保存下来的最小值进行比较。 ```