数据结构：

数组、队列、栈、链表（单向和双向链表）、二叉树、哈希表、字符串、树、图、堆（最大堆和最小堆）

算法：

二分查找、动态规划、分治思想、贪心算法、递归、快慢指针、深度优先遍历、广度优先遍历、快速排序、归并排序、计数排序、回溯、滑动窗口、双指针

评判标准：

可读性、时间复杂度、空间复杂度

语言：

c++、Python

选题：

面试精典150题

做题思路：

模仿GitHub cheat sheet

记录：

题目序号、考察的知识点、解题思路

1. 合并两个有序数组

题目信息：两个非递减顺序排列的整数数组，两个整数分别对应其元素数目，合并2到1中，结果1同样按照非递减顺序排列，注意合并后存储在1中，故1的初始长度为2者长度之和，初始空闲位置的元素均为0。可以存在空数组，但不同时空，结果最长200，数值在正负10^9，题目进阶要求时间复杂度O(m+n)

考察知识点：数组、排序、双指针

解题思路：从条件来看数值在int32表示范围内，数组长度最长200，最短1。第一反应可想到的方法是申请一个新的空间，遍历两个数组，将元素依次存储到新空间中。但是题目要求原位存储在nums1中，这种方法不可行。双指针从两个数组的结尾开始遍历，存储的时候是从最后一个元素开始往前存储。考虑边界条件，如果其中有一方是空的判断，数组是否遍历完，存储元素索引是否到达0。

代码模块：边界判断是否存在空的数组，主while遍历，在遍历的模块中优先判断是否遍历完。

1. 移除元素

题目信息：给定一个数组nums和一个值val，原地移除所有数值等于val的元素，返回nums中与val不同的元素的数量。

考察知识点：数组、双指针

解题思路：需要审题返回值只会考虑前面与val不同的元素的数量。因此最简单的思路是使用快慢指针，快指针遍历一次数组，慢指针指向需要返回的前面与val不同的元素部分，如果快指针指向的值不是val那么就复制到慢指针指向的位置处。这种方法时间复杂度是n。采用双指针，会更快，头指针遍历将所有val置换到尾指针，当头指针超过尾指针遍历结束。

1. 删除有序数组中的重复项

题目信息：已知非严格递增排列的数组nums，原地删除重复出现的元素，结果每个元素只出现一次，输出删除后数组的新长度。注意元素的相对顺序应该保持一致 ，nums被改变后前k个元素为不重复出现的，其余元素与其数组大小不重要。

考察知识点：数组、双指针

解题思路：双指针，快慢指针，慢指针指向需要返回的部分，快指针指向遍历的数组。边界条件是只有一个元素时不做处理。

**总结：考察双指针有一个特征是原位操作，不考虑数组中除返回部分的其他部分**

1. 多数元素

题目信息：已知nums数组大小为n，返回其中出现次数大于n/2的元素；进阶：尝试设计时间复杂度为O(n)、空间复杂度为O(1)的算法解决此问题。

考察知识点：数组、哈希表、分治、计数、排序

解题思路：使用哈希表，空间换时间，将数据存入哈希表中，key值为数字，value为出现的次数，查找到出现次数满足条件的就返回，但注意存在一个边界条件，就是只有一个元素或者所有元素都只出现一次的情况，返回首元素。还有一种思路是排序后返回n/2位置的元素，这两种方法空间复杂度不能达到O(1)，后者时间复杂度最快nlog2n。

1. 买卖股票的最佳时机

题目信息：给定一个数组prices，它的第i个元素prices[i]表示一支给定股票第i天的价格。选择某一天买入，在未来某一个不同的日子卖出股票。设计算法计算能获取的最大利润。若不能获取任何利润，返回0。

考察知识点：数组、动态规划。

解题思路：忘记动态规划的模板了，自己思考先解一下。暴力法时间复杂度太高了，两层for循环。考虑用双指针或者可以称之为滑窗法吗，前面的指针保证指向最小值，后面的尽可能指向最大值，用它们之间的差值控制两个指针的移动，如果差值小于零则前面的指针后移，说明起始数字变小了，更可能找到更大的差值。否则后面的指针向后移动找到当前起点对应的最大差值。边界条件是一个元素的时候返回0。自我感觉比官方题解清晰易懂一些。

1. 罗马数字转整数

题目信息：罗马数字包含7种字符，对应7个数值。通常罗马数字中小的数字在大的数字的右边，但也存在特例，例如4不是4个1而是5减1表示的，只适用于1,10,100分别放在5和10,50和100,500和1000的左边。给定一个罗马数字，将其转换成整数。

考察知识点：哈希表，字符串，数学

解题思路：本来想的是遍历字符串从后往前，遇到1,10,100多个判断看后面的是不是大于它的，是就减一；看了官方解法，进一步将字符和数字用哈希表存起来，会更简便一些。利用好题目中的小数在大数前面的这一条件。注意Python的语法。

1. 最后一个单词的长度

题目信息：给定一个字符串s，由若干单词组成，单词前后用一些空格字符隔开。返回字符串中最后一个单词的长度。单词是指仅由字母组成，不包含任何空格字符的最大子字符串。

考察知识点：字符串

解题思路：遍历字符串，一旦是空格就跳过，如果不是空格就从0开始计数单词的长度，直至遍历完毕，注意对于不是空格的字符需要判断前一个字符是否为空格，重新开始计数。

1. 最长公共前缀

题目信息：编写一个函数来查找字符串数组中的最长公共前缀。如果不存在公共前缀，返回空字符串””。

考察知识点：字典树、字符串。

解题思路：朴素的思路是先找出最短的字符串的长度，然后从第0位开始遍历，直到出现不一样的字符串结束，相同的长度就是需要返回的前缀。评论区看到一个巧妙的方法，就是将字符串们按照字典序进行排序，找出排序后第一个字符串和最后一个字符串的公共前缀。官方题解的一个理解比较关键，就是所有字符串的公共前缀是任意两个字符串的公共前缀，也就是说遍历所有字符串，两个两个去比较，不断的更新公共前缀，即可得到所有字符串的公共前缀。

1. 找出字符串中第一个匹配项的下标

题目信息：已知两个字符串haystack和needle，在haystack字符串中找出needle字符串的第一个匹配项的下标（下标从0开始）。如果needle不是haystack的一部分，则返回-1。

考察知识点：双指针、字符串和字符串匹配。

解题思路：首先可以想到的是暴力法遍历haystack，遍历以每个haystack中字符为开头的字符串是否与needle一致，若一致则直接停止查找返回下标。若遍历完成后未找到，则返回-1。这种方式时间复杂度最差是O（m\*n）。这种方法会导致一些重复的计算量，可以想到通过空间换时间的方式，来降低时间复杂度，一个经典的用于解决字符串匹配的算法角KMP算法。这个方法就是通过存储匹配到模式串中的某一位不等时回退的位置，从而减少比较的次数。

KMP：核心是计算next数组，表示当前字符不匹配时，模式串的指针退回的位置，而文本串的指针永不回退，时间复杂度为O(N)，空间复杂度为O(M)。

1. 验证回文串

题目信息：如果在将所有大写字符转换为小写字符、并移除所有非字母数字字符之后，短语正着读和反着读都一样。则可以认为该短语是一个回文串。注意字母和数字都属于字母数字字符。给你一个字符串，如果它是回文串，则返回true；否则，返回false。

考察知识点：双指针、字符串。

解题思路：双指针分别指向头和尾，大写字母和小写字母之间相差32，指针移动的条件是若匹配上，或者遇到非字母数字字符。若未匹配上则返回false。边界条件是只有一个的时候直接返回true。通过这道题发现isalnum函数判断是否为字母数字字符，lower可以将大写字母转换为小写字母。

1. 判断子序列

题目信息：给定字符串s和t，判断s是否为t的子序列。字符串的一个子序列是原始字符串删除一些（也可以不删除）字符而不改变剩余字符相对位置形成的新字符串。

考察知识点：双指针、字符串、动态规划。

解题思路：首先想到的暴力法思路，两个指针分别指向s和t，遍历t每比对一个过程后改变依次改变t的位置，分别与s比对是否包含。至于好的双指针方法和动态规划方法还没有想出来。因为无需考虑是否连续的字符串，所以双指针法即可，且无需回退。

1. 赎金信

题目信息：给你两个字符串：ransomNote和magazine，判断ransomNote能不能由magazine里面的字符构成。如果可以，返回true，否则返回false。magazine中的每个字符只能在ransomNote中使用一次。

考察知识点：哈希表、字符串、计数

解题思路：先遍历一次magazine，用哈希表记录下每个字符出现的次数，再遍历ransomNote，去哈希表里找当前字符是否存在，存在计数器减一，若哈希表中不存在需要的字符，则返回false。官方题解的思路是，首先判断magazine长度是否大于ransomNote，否则返回false，然后由于只包含26个字母，用数组来存储，遍历magazine，存每个字母出现的次数，遍历ransomNote，出现的字母次数减一，结果小于0则返回false。

1. 同构字符串

题目信息：给定两个字符串s和t，判断他们是否是同构的。如果s中的字符可以按某种映射关系得到t，那么这两个字符串是同构的。不改变顺序，每个出现的字符都应当映射到另一个字符上，不同字符不能映射到同一个字符上，字符可以映射到自己本身。

考察知识点：哈希表、字符串。

解题思路：首先判断两个字符串的长度是否一致，不一致不可同构。然后遍历两个字符串，s串的字符为key值，t串的字符为val值构建一个哈希表。反过来key和value构建另一个哈希表。在构建过程中如若出现key值存在但是，与当前value不匹配，即不可映射。两个哈希表的空间复杂度是双倍的，时间复杂度均为遍历1次即可。

1. 单词规律

题目信息：给定一个模式和一个字符串，判断它们是否遵循相同的模式。要求精确匹配，在字符串模式中的每个字母和每个非空单词之间存在对应关系。

考察知识点：哈希表、字符串

解题思路：利用哈希表，key为模式串中的字母value为字符串中的单词，遍历模式串，向哈希表中添加键值对，如果哈希表中key已经存在，判断此时的字符串和value是否相等。

在中文版中提交后未通过，但是wandbox上复现后输出是对的！中文版LeetCode...

这道题需要注意的是模式和字符串长度可能不一致，需要判断；此外除了要求模式和字符串匹配，反过来字符串和模式也需要键值对去匹配，所以需要构建两个哈希表，双向配对。

1. 有效的字母异位词

题目信息：给定两个字符串s和t，编写一个函数来判断t是否是s的字母异位词。若s和t中每个字符出现的次数相同，则称s和t互为字母异位词。

考察知识点：哈希表、字符串、排序

解题思路：由于题目限制了只包含小写字母，所以创建两个数组，size都是26，每个元素初始值都是0，用于记录每个字符出现的次数，统计完两个字符串出现的次数后，在比对两个数组是否一致。另外一种方式就是对原始的字符串进行排序，排序后是否一致，一致则互为字母异位词。

1. 快乐数

题目信息：快乐数定义，对于一个正整数，每一次将该数替换为它每个位置上的数字的平方和。然后重复这个过程直到这个数变为1，也可能是无限循环但始终变不到1。如果这个过程结果为1，那么这个数就是快乐数。

考察知识点：哈希表、数学、双指针。

解题思路：对于快乐数的计算过程，每个整数依次求取每次经过各个位平方和的结果，判断是否为1。如果不是快乐数，这个计算过程会出现重复的数字，那么需要终止这个运算，并返回false。所以用哈希表来记录是比较合适的。

1. 存在重复元素2

题目信息：给定一个数组nums和一个整数k，判断数组中是否存在两个不同的索引nums[i] = nums[j]，且满足（i-j）<=k。存在返回true，不存在则返回false。

考察知识点：数组、哈希表、滑动窗口。

解题思路：采用哈希表，构建key为元素值，val为index的哈希表，遍历一次数组，每次出现哈希表中已有的key可以计算index的差值是否满足要求。注意在索引差值超过k时更新当前key的index。

1. 汇总区间

题目信息：给定一个无重复元素的有序整数数组。返回恰好覆盖数组中年所有数字的最小有序区间范围列表。输出格式：“a->b”或“a”。

考察知识点：数组。

解题思路：首先数组的长度可能为0，此时返回空的结果。定义一个需要返回的数组，遍历源数组，记录下连续出现数字区间的start和end，将结果依次添加到结果数组中。若start和end不相等用第一种输出格式，否则用第二种输出格式。这里需要注意几个边界条件，一个是原数组只有一个元素与结尾有一个单独元素的情况，需要在for循环后面添加一个判断以免遗漏这部分数据，另外end的初值需要设置为start，另外判断是否连续采用前一个值加1等于后一个值以免出现溢出情况。

1. 有效括号

题目信息：给定一个只包括圆括号、花括号、方括号的字符串S，判断字符串是否有效。

有效字符串需要满足：左括号必须用相同类型的右括号闭合；左括号必须以正确的顺序闭合；每个右括号都有一个对应的相同类型的左括号。

考察知识点：栈、字符串

解题思路：遍历一次字符串，若是左括号入栈，若是右括号，判断栈顶元素是否匹配（栈为空也返回false），匹配将栈顶删除，否则返回false，遍历结束后栈不为空返回false，否则返回true。

1. 环形链表

题目信息：给你一个链表的头结点，判断链表是否有环。

考察知识点：哈希表、链表、双指针

解题思路：定义快慢指针，遍历链表，若连个指针相遇，则存在环。注意在取指针的next时判断是否为空的条件。

1. 合并两个有序链表

题目信息：将两个升序链表合并为一个新的升序链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

考察知识点：递归、链表

解题思路：首先若其中一个链表是空的则返回另一个链表。新建一个表头，遍历两个链表，依次向新建的表头后添加元素。

1. 二叉树的最大深度

题目信息：给定一个二叉树 root ，返回其最大深度。二叉树的最大深度是指从根节点到最远叶子节点的最长路径上的节点数。

考察知识点：树、深度优先遍历、广度优先遍历、二叉树

解题思路：深度优先遍历，递归，返回当前左右子节点最大深度加1。广度优先遍历，遍历每一层，用队列存储当前层的结点，记录遍历的层数。

1. 相同的树

题目信息：给你两棵二叉树的根节点 p 和 q ，编写一个函数来检验这两棵树是否相同。

如果两个树在结构上相同，并且节点具有相同的值，则认为它们是相同的。

考察知识点：树、深度优先遍历、广度优先遍历、二叉树。

解题思路：深度优先遍历，当前结点以及子节点是否相等。首先是否结构不同广度优先遍历，同时遍历两个二叉树，比较每一个结点的val是否相同，若二叉树长度不同或者val不同，则返回false。

1. 对称二叉树

题目信息：给你一个二叉树的根节点 root ， 检查它是否轴对称。

考察知识点：树、深度优先遍历、广度优先遍历、二叉树

解题思路：广度优先遍历，将每一层的数据，正着和反着放入队列，比较是否相同。深度优先不知道如何实现。看了题解，深度优先遍历是，左右对称意味着对于两个结点，1的left和2的right，1的right和2的left是相等的，且这两个结点相等。

1. 翻转二叉树

题目信息：给你一棵二叉树的根节点 root ，翻转这棵二叉树，并返回其根节点。

考察知识点：树、深度优先遍历、广度优先遍历、二叉树

解题思路：深度优先遍历，可以新建一个函数来递归，对于当前结点若为空则返回，否则交换左右结点，对交换后的左右结点递归执行当前函数。也可以不新建一个函数来递归，对于当前结点若为空则返回空，否则将该结点的左右子节点交换（等号右侧的子节点递归调用函数），返回当前结点。

1. 路径总和

题目信息：给你二叉树的根节点 root 和一个表示目标和的整数 targetSum。判断该树中是否存在根节点到叶子节点的路径，这条路径上所有节点值相加等于目标和targetSum。如果存在，返回 true；否则，返回 false。叶子节点是指没有子节点的节点。

考察知识点：树、深度优先遍历、广度优先遍历、二叉树

解题思路：对于当前结点，如果是空节点则不存在符合的路径，如果是叶子结点判断数值是否与targetSum相等，不是叶子结点则递归判断左子树和右子树是否有满足的路径。也可以用广度优先遍历来解决，使用队列保存遍历到的每个结点和当前路径和，若当前结点为叶子结点且路径和满足条件，则返回true。

1. 完全二叉树的结点个数

题目信息：给你一棵完全二叉树的根节点 root，求出该树的节点个数。

完全二叉树的定义如下：在完全二叉树中，除了最底层节点可能没填满外，其余每层节点数都达到最大值，并且最下面一层的节点都集中在该层最左边的若干位置。若最底层为第 h 层，则该层包含 1~ 2^h 个节点。

考察知识点：位运算、树、二分查找、二叉树

解题思路：基础的遍历整个二叉树，采用递归的方式，对于当前结点若为空则范围0，否则为左子树加右子树长度加1。第二种方式是利用完全二叉树的性质来减少遍历的时间复杂度，对于一个满二叉树，其结点个数为2的层数次幂-1，计算二叉树左子树和右子树的层数，若左子树层数大于右子树，则右子树为满二叉树，反之左子树为满二叉树，对于满二叉树只需根据层数计算结点个数，对于另外一半递归计算。由于左子树的深度为当前结点深度减一，故左子树的深度无需遍历计算，减少计算量。二分查找方法是第三种解法，结点的总个数为最后一层结点数加满二叉树个数，后者可以用总层数减一的层数直接计算，而最后一层结点个数需要通过二分查找的方法确定。最后一层结点最大个数已知，最小是1，通过二分查找找到最后一个结点的位置，关键在于如何确定当前索引的结点是否在二叉树中。此处的技巧是通过index与中间分界线比较，确定该结点处于左子树还是右子树。

1. 二叉树的层平均值

题目信息：给定一个非空二叉树的根节点root , 以数组的形式返回每一层节点的平均值。与实际答案相差10^-5以内的答案可以被接受。

考察知识点：树、深度优先遍历、广度优先遍历、二叉树

解题思路：广度优先遍历，利用队列存储来计算当前层的节点的平均值。首先创建一个队列，将根节点存入队列，此时第一层的节点个数为1。通过while循环来迭代，将队列中的节点取出加到和中去，全部取完后除以节点个数，则为结果。

1. 二叉搜索树的最小绝对差

题目信息：给你一个二叉搜索树的根节点root，返回树中任意两不同节点值之间的最小差值。差值是一个正数，其数值等于两值之差的绝对值。

考察知识点：树、深度优先遍历、广度优先遍历、二叉搜索树、二叉树

解题思路：二叉搜索树是一种特殊的树形结构，它的每个结点都有一个可比较的键和一个关联的值。在二叉查找树中，对于任意结点，其左子树上所有结点的键都不大于该节点的键，而右子树上所有节点的键都不小于该节点的键。通过中序遍历可以得到非递减的数值序列，那么在遍历过程中除第一个结点之外，每个结点与前一个结点的差值与最小值相比，不断更新，直至中序遍历二叉树结束。

复习一下：中序遍历递归实现和迭代实现。递归方法是空节点返回，递归左子结点调用，根节点读key，递归右子节点调用。迭代方法是创建一个栈，将根节点以及根节点的所有左节点顺序加入栈中，取出最后一个左节点（他没有左节点，也可看做根节点）将key值加入结果，访问其右节点重复以上步骤，这样实现了中序遍历。