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。

1. 将有序数组转换为二叉搜索树

题目信息：给你一个整数数组nums，其中元素已经按升序排列，请你将其转换为一棵

平衡二叉搜索树。

考察知识点：树、二叉搜索树、数组、分治、二叉树

解题思路：注意平衡二叉树要求左右分支高度差不超过1。从升序数组构建二叉搜索树，即中序遍历二叉搜索树得到升序序列的逆过程，由于平衡二叉树的要求，根节点选在序列的中间，在递归构造其左子树和右子树。

1. 搜索插入位置

题目信息：给定一个排序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，返回它将会被按顺序插入的位置。请必须使用时间复杂度为O(log n) 的算法。

考察知识点：数组、二分查找

解题思路：从要求的时间复杂度可以知道题目要求我们采用二分查找的方法。定义start和end，计算出mid，用mid和目标值比较，目标值小于mid的时候将end移动到mid-1的位置，大于mid的时候将start移动到mid+1的位置，等于的时候找到。在二分查找的过程中start必须小于end，若在这个条件之前仍未找到符合的数字，则返回需要插入的位置start。

1. 二进制求和

题目信息：给你两个二进制字符串a和b，以二进制字符串的形式返回它们的和。

考察知识点：位运算、数学、字符串、模拟

解题思路：二进制加法，每一位相加的结果包括和与进位，参与加法包括两个字符串的当前位和进位。方向是从右向左，对于长度不一样的，现将相同长度部分做相加，在将剩余一个字符串的部分与进位相加，保存的时候也是从右往左保存。利用栈来存储结果，取出后转换为字符串。这里需要注意的是字符与数字之间的相互转换。

1. 颠倒二进制位

题目信息：颠倒给定的 32 位无符号整数的二进制位。

考察知识点：位运算、分治

解题思路：朴素算法是循环32次，遍历每一位，result初始值为0，每次将结果乘以2并加上余数，整数右移一位。分治的方法是先将数据分成两半，然后交换这两半，在分别对这两半中的每一半重复这样的操作。

1. 位1的个数

题目信息：编写一个函数，获取一个正整数的二进制形式并返回其二进制表达式中设置位的个数（也被称为汉明重量）。

考察知识点：位运算、分治

解题思路：朴素的算法是遍历判断每个位是否为1，累加得道汉明权重。另外一种方法是用n&(n-1)来消除n的最后一位1，迭代直至n为0，记录消除的次数即1的个数。

1. 只出现一次的数字

题目信息：给你一个非空整数数组nums，除了某个元素只出现一次以外，其余每个元素均出现两次。找出那个只出现了一次的元素。你必须设计并实现线性时间复杂度的算法来解决此问题，且该算法只使用常量额外空间。

考察知识点：位运算、数组；

解题思路：遍历所有数字做异或运算，因为其余每个元素均出现两次，所以异或的结果是唯一一个出现一次的数字。

1. 回文数

题目信息：给你一个整数x，如果x是一个回文整数，返回true；否则，返回false。

回文数是指正序（从左向右）和倒序（从右向左）读都是一样的整数。

考察知识点：数学

解题思路：首先可以确定小于0的数必然不是回文数，将数的各个位倒过来计算得到的数是否与原来的数相等，但是这种方法会出现溢出错误。于是，我们考虑只翻转一半数据后停止，判断这两半是否相等，但是此时需要注意边界与数字奇偶数位的情况。边界条件是10的倍数在这种方法下会判断错误并且他们本身不是回文数，另外奇数位翻转完的数是偶数被的10倍，中间一位的数不影响回文数的判断。

1. 加一

题目信息：给定一个由整数组成的非空数组所表示的非负整数，在该数的基础上加一。

最高位数字存放在数组的首位，数组中每个元素只存储单个数字。你可以假设除了整数0之外，这个整数不会以零开头。

考察知识点：数组、数学

解题思路：朴素解法，遍历数组，计算整数加1。另外，需要利用起来加1这个信息，对于简化解题方法很重要，对于一个数字加1有两种情况9+1有进位，和其他数字加1没有进位。所以在遍历过程中若加1后没有进位（当前位没有变为0即可返回），另外在全为9的条件下，需要在最后的数组前面插入一个1。

69. x的平方根

题目信息：给你一个非负整数x，计算并返回x的算术平方根。由于返回类型是整数，结果只保留整数部分，小数部分将被舍去。注意：不允许使用任何内置指数函数和算符，例如 pow(x, 0.5) 或者 x \*\* 0.5 。

考察知识点：数学、二分查找

解题思路：采用二分查找，边界条件0的算术平方根为0。起始为1，end为x，为了防止溢出，将mid平方与x的差值改为商作差，二分查找运算，最后返回end值。可以通过将end初始值设置为x/2来避免多余的计算。

1. 爬楼梯

题目信息：假设你正在爬楼梯。需要n阶你才能到达楼顶。每次你可以爬 1 或 2 个台阶。你有多少种不同的方法可以爬到楼顶呢？

考察知识点：记忆化搜索、动态规划、数学

解题思路：从1开始记录路径数，1有1种，2有2种，3有（1 + 2）种，4有（2 + 3）中，即后一个是前两个的和，这是由于每次可以爬1或2个台阶。所以1和2是动态规划中的初值，dp数组为dp[n] = dp[n-1] + dp[n-2]。

11.盛水最多的容器

题目信息：给定一个长度为n的整数数组height。有n条垂线，第i条线的两个端点是(i, 0)和(i, height[i])。找出其中的两条线，使得它们与 x 轴共同构成的容器可以容纳最多的水。返回容器可以储存的最大水量。说明：你不能倾斜容器。

考察知识点： 贪心、数组、双指针

解题思路：

1. 暴力法是遍历所有组合找出面积最大的组合，时间复杂度在O(n^2)。
2. 考虑满足条件的结果是间隔乘以高度最小值的乘积最大。以当前柱子为桶的高度进行查找，分别遍历当前柱子为桶的左右边界进行查找。
3. 也可以采用双指针，利用贪心的原则让它们向中间靠拢，找到最大的组合。起始时刻两个指针分别在数组首尾，此时宽度一定，最小的高度决定了当前组合的乘积结果；考虑高度低的向中间靠拢，此时宽度变小，可能会导致乘积结果变小、不变、变大，而高度高的向中间靠拢，可能会导致乘积结果变小、不变，因此为了找到更大的结果，让高度低的向中间靠拢，且找到比当前高度大的（这样才可能使得结果更大）。
4. 旋转链表

题目信息：给你一个链表的头节点 head，旋转链表，将链表每个节点向右移动k个位置。

考察知识点：链表、双指针

解题思路：将链表的首尾连接形成一个环形并记录链表的长度len，根据k值来找到旋转后链表的head（step = len - k % len - 1），将head与前一个结点断开，返回head结束。

1. 无重复字符的最长子串

题目信息：给定一个字符串s，请你找出其中不含有重复字符的最长子串的长度。

考察知识点：哈希表、字符串、滑动窗口

解题思路：

遍历一遍字符串，分别以每个字符为开始，创建一个哈希表，连续添加后面的字符直至出现重复，记录其长度与当前最大长度比较。

采用双指针或者角滑动窗口，保证窗口内的字符串是不含有重复字符的。初始窗态窗口左右边界相等，将右边界右移，若未出现重复字符更新maxlen，若出现重复字符左边界右移直至无重复字符且左边界不可超过右边界。当右边界超出范围结束。窗口内的字符串使用哈希表来存储，key为字符，value为位置，新字符加入哈希表，重复出现的字符需要更新其出现的位置。需要注意的是对于重复出现的字符要同时判断left是否已经超过第一次出现该字符的位置，以免错误更新left值，这一条件是判断窗口内该字符是否重复，哈希表存储了历史遍历的字符，大于窗口的范围。

1. 最长递增子序列

题目信息：给你一个整数数组nums，找到其中最长严格递增子序列的长度。子序列是由数组派生而来的序列，删除（或不删除）数组中的元素而不改变其余元素的顺序。例如，[3,6,2,7] 是数组 [0,3,1,6,2,2,7] 的子序列。

考察知识点：数组、二分查找、动态规划

解题思路：

采用动态规划的方法，dp数组的含义，每个元素表示截止第i个最长子序列；初值，每个初始值为1；递推关系，dp[i]=max(dp[j]+1,dp[i])，遍历当前元素的子数组，运用这个递归关系来更新dp[i]，找到最长子序列。

采用二分查找的方法，维护一个递增的序列，序列长度为最长子序列长度，如果结尾的值比新的值小序列长度加1，否则在序列中找到倒数第一个小于新的值的位置并将该新的值替换其后一个位置。这种方法也用到了贪心算法，如果上升子序列的结尾数值越小，那么遍历数组后面接上的数可能会更多，所以记录长度固定的情况下，结尾最小的那个元素。

322.零钱兑换

题目描述：给你一个整数数组coins，表示不同面额的硬币；以及一个整数amount，表示总金额。计算并返回可以凑成总金额所需的最少的硬币个数。如果没有任何一种硬币组合能组成总金额，返回-1。你可以认为每种硬币的数量是无限的。

考察知识点：广度优先搜索、数组、动态规划

解题思路：将问题转换为子问题的迭代，定义dp数组，dp[j]表示j金额需要的最少硬币数量，遍历当前的coins数组，对于coins[i]计算min(dp[j-coins[i]]+1,dp[j])，它表示使用硬币coins[i]当前j金额需要的最少硬币数。初始状态dp[0]=0，由于对于amount金额j的最大值是amount，所以dp数组中其他元素数值最大值设置为amount+1。（背包问题，包的容量一定，给定几个值，用最少的值装满）

1. 删除链表中的倒数第N个结点

题目描述：给你一个链表，删除链表的倒数第n个结点，并且返回链表的头结点。

考察知识点：链表、双指针

解题思路：朴素的解法，遍历链表得到链表的结点个数，删除倒数第N个结点对应正数第几个，找到要删除的结点位置做删除的操作。这需要遍历两遍链表，题目要求只遍历一遍来实现这个目标。有这样两种方法，一种是利用栈先入后出的特性，遍历一次链表将结点存到栈中，在取出N个后找到要被删除的结点的前驱结点；一种是利用双指针，这两个指针的间隔是N，当第二个指针指向最后一个结点是，第一个指针指向要删除的前驱结点。删除结点需要找到前驱结点改变它的next。有一个细节是添加一个dummy node减少删除头结点的边界判断。

1. 打家劫舍

题目信息：你是一个专业的小偷，计划偷窃沿街的房屋。每间房内都藏有一定的现金，影响你偷窃的唯一制约因素就是相邻的房屋装有相互连通的防盗系统，如果两间相邻的房屋在同一晚上被小偷闯入，系统会自动报警。

给定一个代表每个房屋存放金额的非负整数数组，计算你不触动警报装置的情况下 ，一夜之内能够偷窃到的最高金额。

考察知识点：数组、动态规划

解题思路：第一种思路，当打劫当前屋子时，前一间屋子跳过，当前最大值为前i-2中的最大值加上当前屋子的价值，最后最大值定在打劫倒数第一间屋子和倒数第二件屋子之间。第二种思路，在当前屋子可盗窃的最大值，在前一间屋子和前前一间加上当前之间取最大，返回值即为dp结尾的数。

1. 最小路径和

题目信息：给定一个包含非负整数的m x n网格grid，请找出一条从左上角到右下角的路径，使得路径上的数字总和为最小。说明：每次只能向下或者向右移动一步。

考察知识点：数组、动态规划、矩阵

解题思路：由于只能向下和向右运动，所以对于(i,j)元素，其路径最小值为当前值加上min((i-1,j),(i,j-1))，需要判断元素是否没有上或者左，没有上即i=0用(i,j-1)，没有左即j=0用(i-1,j)。遍历整个二维数据计算出二维dp。

1. 最长连续子序列

题目信息：给定一个未排序的整数数组nums，找出数字连续的最长序列（不要求序列元素在原数组中连续）的长度。请你设计并实现时间复杂度为O(n)的算法解决此问题。

考察知识点：并查集、数组、哈希表

解题思路：采用哈希表的方法解决，利用set先去重，然后遍历一遍set对每个元素判断num-1是否在set中，不在就是队首以此为起点开始计算长度，若存在则跳过表明存在前驱结点。

1. 三角形最小路径和

题目信息：给定一个三角形 triangle，找出自顶向下的最小路径和。每一步只能移动到下一行中相邻的结点上。相邻的结点在这里指的是下标与上一层结点下标相同或者等于上一层结点下标+ 1的两个结点。也就是说，如果正位于当前行的下标 i ，那么下一步可以移动到下一行的下标 i 或 i + 1 。

考察知识点：数组、动态规划

解题思路：定义二维dp数组，元素表示到达当前结点路径最小值，边界条件最左面和最右面的前驱结点只有一种可能，其余的选择两个中较小的一个。最终的结果在最后一行中选择最小值。

1. Pow(x,n)

题目信息：实现 pow(x, n)，即计算x的整数n次幂函数。

考察知识点：递归、数学

解题思路：快速幂算法，利用移位和位运算，采用分治的思想，考虑n，若二进制最后一位为1则为奇数，将x乘到res上，x乘等到x^2，n右移一位。若n为复数则x=1/x。（从二进制的角度去理解更清晰，将n表示为二进制，则x的n次方可拆分，每次计算一位后右移计算下一位）

1. 单词拆分

题目信息：给你一个字符串s和一个字符串列表wordDict作为字典。如果可以利用字典中出现的一个或多个单词拼接出s则返回 true。注意：不要求字典中出现的单词全部都使用，并且字典中的单词可以重复使用。

考察知识点：字典树、记忆化搜索、数组、哈希表、字符串、动态规划

解题思路：采用动态规划的方法，定义一维的dp数组，每个元素表示以当前元素为结尾的字符串是否可以用列表中的字符串拼接出来。初始状态dp[0]表示空字符串，它的值为true即有效。状态转换过程，遍历每个字符串元素，在每一次遍历字符串列表，是否存在以当前字符为结尾的子串与字符串列表中的字符串相等，若相等则dp[i]的值由dp[i-sz]决定，sz为匹配上的子串长度。这样将问题转化为了子问题，可以递归解决。

1. 分隔链表

题目信息：给你一个链表的头节点 head 和一个特定值x，请你对链表进行分隔，使得所有小于x的节点都出现在大于或等于x的节点之前。你应当保留两个分区中每个节点的初始相对位置。

考察知识点：链表、双指针

解题思路：先找到第一个大于等于x的位置，随后遍历剩余的结点，将每一个小于x的值插入到该位置之前。还有一种简单的方法，将小于x的放入small链表，大于等于的放入big链表，最后将它们连接上即可。

72.编辑距离

题目信息：给你两个单词 word1和word2，请返回将word1转换成word2所使用的最少操作数。你可以对一个单词进行如下三种操作：插入一个字符，删除一个字符，替换一个字符

考察知识点：字符串、动态规划

解题思路：没有思路的一题，看题解学习一下。动态规划的方法，定义dp数组二维，第一个维度是word1的size+1，第二个维度是word2的size+1，dp[i][j]代表word1中到i的字符串转换成word2中到j的字符串搜需要的操作数。之所以在size上加1是考虑了空字符，即word1和word2前均加入空字符。第一行每一个元素表示空字符转换为word2的每个字串需要的操作数，第一列表示word1中的每个字串转换为空字符需要的操作数。动态规划的状态转换关系，对于遍历到的当前i和j，若word1[i]==word2[j]则dp[i][j] = dp[i-1][j-1]+1,若word1[i]!=word2[j]，考虑可做的操作为替换、删除、插入，分别对应dp[i-1][j-1]+1,dp[i-1][j]+1,dp[i][j-1]+1，将它们之间的最小值赋值给dp[i][j]即可。最后返回dp[end][end]。（注意代码中应该是word1[i-1]==word2[j-1]，因为i和j的范围是1到len）

接下来在继续47道例题

1. 两数之和

题目信息：给定一个整数数组nums和一个整数目标值target，请你在该数组中找出和为目标值target的那两个整数，并返回它们的数组下标。你可以假设每种输入只会对应一个答案，并且你不能使用两次相同的元素。你可以按任意顺序返回答案。

考察知识点：数组、哈希表

解题思路：遍历一次数组，利用哈希表存储已经遍历的元素key为该元素的值，value为索引，找到的时候直接返回当前元素和哈希表中元素的索引即可。

1. 两数相加

题目信息：给你两个非空的链表，表示两个非负的整数。它们每位数字都是按照逆序的方式存储的，并且每个节点只能存储一位数字。请你将两个数相加，并以相同形式返回一个表示和的链表。你可以假设除了数字0之外，这两个数都不会以0开头。

考察知识点：递归、链表、数学

解题思路：遍历两个数，进位c初始为0，sum初始为0，当两个数存在非空结点的时候，更新计算sum和c，然后将sum连接到result后，最后需要判断c是否为1加入到result。

4. 寻找两个正序数组的中位数

题目信息：给定两个大小分别为m和n的正序（从小到大）数组nums1和nums2。请你找出并返回这两个正序数组的中位数。算法的时间复杂度应该为O(log (m+n))。

考察知识点：数组、二分查找、分治

解题思路：从算法的时间复杂度要求可以看出暴力法是不满足的，需要采用二分查找的方法。暴力法是将两者合并，如果是奇数个选择中间的一个数，如果是偶数个选择中间的两个数的平均值。思考两个数组，如果一个数组的左边界大于另一个的右边界，则中位数是数值小的那个数组（m+n）/2个数，不够的在数值大的数组中取。如果一个数组的左边界小于另一个数组的右边界，则中位数是他们混合后的中间数值，中位数位置之前，两个数组中的数据满足这样的规律，num1中的right1大于num2中的left2，num2中的right2大于num1中left1.根据这样的规律，看到一个题解的思路是固定值找出前一半数，不固定的是在两个数组中切割下来的部分，在短的数组中二分查找即短数组切位置二分，长数组用固定值减去它，根据前面的关系来判断lo和hi的移动，right1<left2则lo在mid上加一，left1>right2则hi在mid上减1；为了统一奇数个和偶数个的区别，虚拟将所有数的前后插入字符，数据索引号增加一倍。

5.最长回文字串

题目信息：给你一个字符串s，找到s中最长的回文子串。

考察知识点：双指针、字符串、动态规划

解题思路：回文串指的是正向和反向读都是一样的字符串，所以判断字符串是否为回文串需要用双指针。在动态规划方法中定义dp数组为二维的其中dp[i][j]表示i到j这一闭区间内的字符串是否为回文串，这里注意j一定是大于或等于i的，当j-i <3的情况下dp[i][j]是否为回文串取决于s[i]和s[j]是否相等。其他时候dp[i][j]还取决于dp[i+1][j-1]。画出二维的格子图，可以知道我们需要按列去写dp数组。

1. 三数之和

题目信息：给你一个整数数组nums，判断是否存在三元组 [nums[i], nums[j], nums[k]]满足 i != j、i != k 且 j != k ，同时还满足 nums[i] + nums[j] + nums[k] == 0 。请你返回所有和为 0 且不重复的三元组。注意：答案中不可以包含重复的三元组。

考察知识点：数组、双指针、排序

解题思路：对于找出三数之和等于固定值，可以通过两层遍历原数组，并采用哈希表来存储已经遍历的值来判断是否满足要求。题目要求不重复，对原数组排序，在每次遍历元素的时候，出现与前一个重复的元素则跳过。在每次找到满足要求的数值后，跳过后面与当前数值相同的数。

1. 最大子数组和

题目信息：给你一个整数数组nums，请你找出一个具有最大和的连续子数组（子数组最少包含一个元素），返回其最大和。子数组是数组中的一个连续部分。

考察知识点：数组、分治、动态规划

解题思路：暴力法是遍历所有子数组，计算和，找出最大值。用动态规划，一维dp数组，dp[i]表示当前元素为结尾的最大和，dp[i]和dp[i-1]的关系，当num[i]大于0且dp[i-1]>0则加上，否则不加，当num[i]小于0且dp[i-1]>0则加上，否则不加。

7.整数反转

题目信息：给你一个32位的有符号整数x，返回将x中的数字部分反转后的结果。如果反转后整数超过32位的有符号整数的范围[−2^31, 2^31−1] ，就返回0。假设环境不允许存储64位整数（有符号或无符号）。

考察知识点：数学

解题思路：反转整数是求余数得到n，x\*10+n，再除以10缩小的这一过程。题目存不下超过32位有符号数表示范围的数值，所以要提前判断是否超出范围。对于x\*10+n>INT\_MAX的情况，x>INT\_MAX/10并且还要附加一个n的情况下，一定会溢出，此外x=INT\_MAX/10且n>7的情况下也会溢出，对于x\*10+n < INT\_MIN的情况，x<INT\_MIN/10且需要添加n的情况下必会溢出，此外x=INT\_MIN/10且n<-8的情况下会溢出。

1. 接雨水

题目信息：给定n个非负整数表示每个宽度为1的柱子的高度图，计算按此排列的柱子，下雨之后能接多少雨水。

考察知识点：栈、数组、双指针、动态规划、单调栈

解题思路：暴力法是遍历每一个柱子，计算它可以接到雨水的数值。通过计算其左侧最高和右侧最高，再从这两个最高值中找到最小值，该最小值若大于该柱子高度，可接到雨水，数值是他们的差值。最后每个柱子可以接到雨水的数值之和即为结果。为了减少时间复杂度，用两个数组分别存储左侧最大和右侧最大的高度值，空间换时间。为了节省空间，左侧的最大值可以用一个变量存储，因为历史数据不需要存储。进一步，同时用两个柱子接雨水，这样左侧和右侧均可以用一个变量来存储。注意：当前index的左侧计算方法，max（max\_left，index-1）即该元素左侧一个和左侧的左侧最大值比较，递推得到。右侧求解方法同理。可以接雨水的柱子不包括首和尾。还有一种栈、单调栈的方法，栈中存储的index对应的元素值不递增。当前元素和栈顶元素比更大，则取出栈顶元素，新栈顶元素和当前元素计算原栈顶的雨水，直到当前的元素不大于栈顶或栈空当前元素入栈，在此之前循环计算雨水的操作。若当前元素小于等于栈顶直接入栈，或者栈是空的直接入栈。这个过程类似与匹配括号，遇到空隙就接雨水，它与前面四种方法接雨水计算的方式不一样，采用区间乘以高度差的形式。

1. 正则表达式匹配

题目信息：给你一个字符串s和一个字符规律p，请你来实现一个支持'.'和'\*'的正则表达式匹配。'.' 匹配任意单个字符；'\*' 匹配零个或多个前面的那一个元素，所谓匹配，是要涵盖 整个字符串s的，而不是部分字符串。

考察知识点：递归、字符串、动态规划

解题思路：定义dp数组，size为lens+1和lenz+1。从后往前进行匹配，因为从前往后匹配需要考虑的点比较复杂，反过来更容易分析。s的索引从0~i-1,p从0~j-1，s和p匹配上，则dp[i][j]为true。考虑当前i-1和j-1是否匹配，若字母相等或者j-1为‘.’则它们匹配上了，dp[i][j] = dp[i-1][j-1]。若没匹配上且j-1为‘\*’,考虑‘\*’前面的字符，重复0次，1次或者大于等于2次；重复0次意味着前一个字符和当前也不匹配，则dp[i][j] = dp[i][j-2],重复1次时dp[i][j] = dp[i-1][j-2]，重复大于等于2次，拿出一个和当前的匹配dp[i][j]=dp[i-1][j]。否则就为false。考虑边界条件，p为空且s不为空则必不可匹配，p不为空且s为空可能匹配上在p的结尾是’\*’时，此外匹配补上。

1. 括号生成

题目描述：数字n代表生成括号的对数，请你设计一个函数，用于能够生成所有可能的并且 有效的括号组合。

考察知识点：字符串、动态规划、回溯

解题思路：动态规划，如何由n-1到n，新增加的括号固定在最左侧，那么n-1个括号可能在左侧括号的里面或者在它的右侧。括号里面的括号个数为p，括号右侧的括号个数q，p从0到n-1遍历，q从n-1到0遍历，它们的和为n-1。当括号里为p时，则p个括号所有可能的组合，括号外为q时，则q个括号所有可能的组合，遍历这些所有组合得到不重复且完整的括号组合。

1. 链表反转

题目描述：给你单链表的头节点head，请你反转链表，并返回反转后的链表。

考察知识点：递归、链表

解题思路：新建一个哑结点，遍历原始阶段，将每一个遍历到的结点插入到哑结点后面。

1. LRU缓存机制

题目描述：请你设计并实现一个满足LRU (最近最少使用)缓存约束的数据结构。

实现 LRUCache 类：LRUCache(int capacity) 以正整数作为容量 capacity 初始化LRU缓存

int get(int key)如果关键字key存在于缓存中，则返回关键字的值，否则返回-1。

void put(int key, int value)如果关键字key已经存在，则变更其数据值value；如果不存在，则向缓存中插入该组key-value。如果插入操作导致关键字数量超过capacity，则应该逐出最久未使用的关键字。函数get和put必须以O(1)的平均时间复杂度运行。

考察知识点：设计、哈希表、链表、双向链表

解题思路：这道题关键是理解两点，1.LRU的功能可以使用双向链表实现，访问到的节点移动到头部，超出容量的从尾部删除。2.要实现O(1)得使用HashMap，里面存储key与链表节点即可，这样可以快速定位节点，然后删除它，将它移动到链表头部。

首先，定义双向链表节点结构体，每个节点包含key、value、前驱节点以及后驱节点，定义其构造函数，包含有参数的和无参数的构造函数。

然后对于LRUCache类，定义私有成员变量，包括哈希表，每个节点存key和链表节点地址，还包含链表伪头结点、伪尾结点、链表当前的sise、Cache的最大缓存容量。

定义public类型的成员函数，包含构造函数，get，put，addToHead，removeNode，moveToHead，removeTail。其中，构造函数是根据缓存容量构造一个size为0的空链表，get是查找key节点，不存在返回-1，存在则返回其value，并将它移动到头部；put是将节点添加到链表，若不存在，创建该节点添加至哈希表并添加到链表的头部，此后如果链表的size超过了缓存容量，将尾部的节点删除同时哈希表中对应的项删除，若存在那么修改该节点的value并将它移动到头部。而get函数需要使用moveToHead，put函数需要使用addToHead、removeTail、removeToHead，而removeTohead包含removeNode和addToHead两个操作。

1. 搜索旋转排序数组

题目信息：不存在重复元素的整数数组在k处旋转，给出一个target，若不存在返回-1，否则返回它在数组中的下标（旋转后的），时间复杂度必须为O(logn)。

考察知识点：数组、二分查找

解题思路：与未旋转的数组相比，这种情况下整个数组是非单调递增的，存在骤减的两个点。需要这样思考，当前半段是递增的且target在前半段则向前规约，当后半段是递增的且target在后半段则向后规约，否则向相反的方向规约。

1. 全排列

题目信息：给定一个不含重复元素的数组，返回其所有可能的全排列，顺序可任意。

考察知识点：数组、回溯

解题思路：一种方法是采用深度优先遍历，通过元素交换来固定当前深度的数值，递归终止条件是当前位为结尾位，可将结果加入到res集合中；递推是对当前深度进行固定数值，然后对下一深度同样操作，直至达到终止条件。还有另外一种方法是采用深度优先遍历，遍历所有分支，用数组存储元素是否使用，该条件作为剪枝的条件。与第一种方法类似。总结一下，两种方法均为递归，采用深度优先遍历的思想，遍历所有路径。递归的入口是当前处于的深度，递归的过程是对下一层进行遍历，递归的出口是深度到达边界，该路径被记录下来。需要在回溯的时候将状态重置或者恢复到下一层遍历之前。

1. 柱状图中最大的矩形

题目信息：给定一组非负整数，用来表示每个柱子的高度，柱子的宽度是1，求在该柱状图中，能够勾勒出的矩形的最大面积。

考察知识点：栈、数组、单调栈

解题思路：矩形的面积等于长乘以高，长是单位1乘以柱子个数，高是柱子中最矮的高度。首先用暴力法，对于当前的元素以此为起点向左找到第一个小于它的，向右找到第一个小于它的，然后计算以当前元素为高度的该区间内矩形面积，遍历所有元素计算的最大值为结果。但是暴力法超出了时间限制。或者另外一种暴力法是枚举宽度，在所有宽度内找到最小的高度计算面积，遍历所有可能求出最大值。采用单调栈的方法，维护一个单调递增的栈，栈中存储heights中元素的位置，为了减少边界的特殊判断，将heights首和尾插入一个0。遍历heights，若栈为空或者当前的元素大于等于栈顶则入栈，若栈不为空且当前元素小于栈顶则计算栈顶元素的矩形面积，当前元素位置和其前一个元素位置区间内乘以该栈顶元素的高度。这种方法采用栈存储了其左侧第一个小于它的数值，空间换时间，降低了时间复杂度。