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)。