1级，easy， medium，

array（18）

**217. has duplicate?** 可以用dictionary。或者直接用set：len(nums) != len(set(nums))

**219. has duplicate2?有重复的数，且数的index差<=k：**建dict，key为数，value为index的list。

**26. remove duplicate(sorted)**：j是计数器，i往下扫,遇到不重复的数(跟i-1 的数不一样),j就+=1。 所以长度就是j

**27 remove固定的数val**：可以用while val in list: list.remove(val)。也可以用先set把重复去掉，再转成list，再list.remove(val),最后都是返回list

**283 move zeros,把所有的0都移到后面去**。跟26题一样，i往下扫，遇到0，i就+=1，遇到非0，nums[j] = nums[i]，让j位置上的数变成现在i位置上的数，然后i,j都+=1

**48：90度旋转image**：先reverse,再双循环 matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]

**66. list变成数字，数字+1后变成list**：for循环把数字找到：sum += digits[-(i+1)]\*(10\*\*i) 末位乘以10的n次方，把数字变成list[int(i) for i in str(sum)]

**88. merge两个sorted list:** 就是两个list相加nums1[:m]+nums2[:n]， 再sorted

**118 Pascal's三角**：[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]两层循环，row.append(list[i-1][j-1]+list[i-1][j])

**119 Pascal's三角2**：就是找到这个三角里的第k行，同上

**169 majority element**，就是那个数出现的次数多余一半。用dictionary。还可以用majority vote多数投票法，从第一个数开始设为candidate，往后遍历，遇到一样的，count++，否则count--，count变成0时candidate重新设为当前那个数且count++，最后candidate那个数再检验一下

**229 majority element2**，找到出现次数多余1/3的数。多数投票法。这回最多有两个数会出现1/3次，所以两个candidate两个count，其他一样

**189 rotate array**，n个数的array，从后往前调数，调k次。array.insert(0,array.pop())

**228. 连续的range**：比如[0,1,2,4,5,7]🡺["0->2","4->5","7"].单独一个def设计板式“a->b”。挨着的数的差大于1时，就写入板式。if nums[i]+1 != nums[i+1]: res.append(printRange(nums[start], nums[i]))

**163 missing range:** 例如 [0, 1, 3, 50, 75] , 从0到99🡺 ["2", "4->49", "51->74", "76->99"]。也是单独一个def设计板式，把头，中间，尾的情况都考虑进去

**243 element的最短距离**，例如["practice", "makes", "perfect", "coding", "makes"]找到"makes"和"coding"间的最短距离：单独一个def生成index list，然后两两比较index list里的值的差

**244 element的最短距离**，考虑这个函数会被重复调用很多次，用class，在constructor里receives a list of words

**245 element的最短距离**，考虑重复找到"makes"和"makes"间的最短距离

string（14）

**5 longest palindromic substring**: 先k=j=i，往下遍历i,一旦while s[k + 1] == s[j - 1]: k, j = k + 1, j – 1。这样k和j分别向两边走，能找到回文的头和尾。

**9 Palindrome Number** 直接return str(x)==str(x)[::-1] 或者把原数的除以10的余数（找个位数的意思）填到新的数里面再乘以10，得到的看和原数一样不

**234 Palindrome Linked List**

**125 Valid Palindrome**是否为回文。两个指针分别从头和尾往中间走，用while，如果一样，两个一起往下走一个。直到相遇。如果不一样就false

**131 Palindrome Partitioning**

**266 Palindrome Permutation** 字母排列后有没有回文。用collections.Counter(s).values())算出每个字母出现的次数，这个次数对2取余，因为只能有一个单泵的，所有加起来不能大于2.

**6 zigzag**。划分块：cycle = (2\*nRows - 2)，在每个circle里，除first和last行，其他行都是两个数。 i是row,k是circle，j是第一竖趟的index，secondJ 是其他竖趟的index。 j = i + cycle\*k。secondJ = (j - i) + cycle - i。（j-i）就是circle的倍数。

**165 version对比**。先用‘.’把version写成list： versions1 = [int(v) for v in version1.split(".")]在对应着比较。

**67 binary**。给两个二进制数，算他们的和。

**58 length of last word**.用空格分开，然后算最后一个的长度

**38 count and say** 。‘1, 11, 21, 1211, 111221..’后面数是前面数的描述。两个def,一个从1开始产生后面的描述。一个用来给数生成描述

**28. Implement strStr()**。一个string是不是在另一个string里，找到起始位置。双层循环，里层遍历小string,如果发现第一个相同，i,j同时向后走

**14 Longest Common Prefix**。找到一个list strings的最长的前缀。用list里第一个str做标杆，取str里的第一个字符，看其他str里有没，然后再取第二个字符，看其他str里有没，直到没有，就return前面的字符

**8 String to Integer (atoi)** 要去除前后的空格(.strip),要留下前面的正负号,如果有不是数字的element(用.isdigit检查),return 0

binary search（5）

**69 sqrt(x)开根号**。公式为cur = pre/2 + x/(2\*pre)

**242 valid anagrams**.是不是变位词 s = sorted(s)，t = sorted(t)，return s == t

**274 H-index**：H篇文章被引用H次以上。例如 [3, 0, 6, 1, 5]，有3篇文章引用3遍以上。先sorted [0,1,3,5,6]，1往后的文章都引用1遍以上，citations[i] >= (n-i)

**275 H-index2**：已经sorted好的。另一种做法是binary search，每次掐到中间mid = (low + high) / 2 if N - mid > citations[mid]:

**278 first bad version**.找到最先坏的那个版本，同上。

math（14）

**7 reverse integer**。对10取余，然后这个余乘以10的n次方，然后把这个数整除10得到少一位的数，result += x%10\*(10\*\*(lenx-i-1)) 然后x = x//10

**168 Excel Sheet column title**.给数字退出字母。 先列出d[0]='Z'一直到d[26]='Z',然后再用下面的语句 a = temp % 26, temp /= 26（跟十进制一样）

**171 Excel column number**给字母推出数字先列出d['A']=1一直到d['Z']=26，然后用num = num+26\*\*i\*d[le[len(le)-1-i]]

**172 factorial zero**.求阶乘的结果里有几个0.就是要看里面有多少个5的倍数，同时要考虑的25，125，625这些也是5的倍数。所以用递归n/5 + self.trailingZeroes(n/5)

**202 happy number**。digit的平方和一直算，能算到1就是happy。单独一个def算平方和。主函数while这个def知道等于1，如果while100次还不等于1，就false

**223 求两个长方形的总面积**。面积和去掉重合部分

**231 2的幂**。2进制。略

**326 3的幂**。用log。x = math.log(n,3),然后abs(round(x) - x) < 0.0000000000001。或者：用最大的3的幂对这个数取余return 1162261467 % n == 0

**258 add digits**。一个数的digit相加直到加到各位数为止。或者就是求9的余数。（比如11%9=2，21%9=3）

**263 ugly num**。能被2，3，5整除。prime=[2,3,5]用while，num去除以prime里面的每一个元素，除尽为止，最后如果剩下的如果是1就true

**264 ugly num2**。找到第n个ugly number。这个队列ugly是：1，2，3，2\*2，5，3\*2（或2\*3），4\*2，3\*3，2\*5，6\*2（或4\*3）就是2，3，5的倍数按大小顺序往ugly里append，倍数是从ugly里出的。u2, u3, u5就是这些个数,i2,i3,i5是找到u的index.每个index用完,就继续自加(+=1)

**313 super ugly num**.这回不是2，3，5，是任意给定的质数。比如prime=[2，7，13，19]。求第n个number。ugly[i] = min(prime[j]\*ugly[index[j]]) j ~(0,len-1), Index = [0,0,0,0]

**292 Nim game**。两人给一叠石头，每次只能移走1，2，3块，最后移走最后一块谁赢。你先来，问你能赢不。return bool(n % 4)

**303 range sum query**。算list里从i到j的数的和。用（0~j）减去（0~i）。这题要求用constructor，在\_\_init\_\_里list是输入，累积和accu是输出for k in range(1,len(nums)): self.accu.append( self.accu[k-1] + nums[k]) 用了递归。

hash table（5）

**36 valid Sudoku**。9\*9的格子里random有数，横不能重复，纵不能重复，3\*3的块里不能重复。用到for i, row in enumerate(board)

和for j, c in enumerate(row)的双重循环。往一个list里填数对,seen += (c,j),(i,c),(i/3,j/3,c) (c,j)c表示那个数,j表示那个列.所以这个意思是第j列有这个数c,(i,c)第i行有这个数c,这个小方块里有这个数，最后如果这个list没有重复的,就true

**204 count prime**。n以内(<n)有几个质数。从2开始，2的倍数都不是质数，然后3的倍数，然后往下依次。所以设一个n个数的list=[false false,true,true,true…..] ，双层循环，I,j 每次if res[i] == True:list[i\*j] = False()就是倍数就为false。

**205 isomorphic strings**。同构string例如"egg", "add"。用dictionary，第一个词作为key，第二个词作为value。一对一的

**290 word pattern**。类似同构。pattern = "abba", str = "dog cat cat dog"用dictionary，value用append。最后每个value个数都是1才是true

**299 bulls and cows**。Secret number:"1807";Friend's guess:"7810": 1 bull and 3 cows. (The bull is 8, the cows are 0, 1 and 7.) 第一轮循环，用对应关系把bull找到，并且把secret里没被猜到的数作为dic里的key,value为+=1. 第二轮，遍历guess，里面的数，在dict的key里没有，就不是cow, 如果有，且对应的dict里的key的值>0,就说明是cows，cows的计数就+=1，然后value -=1

linked list(LL)（11+1）

**19 remove node** 。从LL尾移除第n个node。两个pointer,fast,slow，使fast到tail时，slow到想删除的那个数的前一个。然后让slow.next = slow.next.next。

**203 remove element** 。移除LL的所有值为val的node。用while，遇到 cur.next.val == val就 cur.next = cur.next.next，其他的就cur = cur.next

**83 remove duplicate**。把重复的从sorted里移除（本身不删）。用while，如果node.val == node.next.val就node.next = node.next.next，其他的就node = node.next

**82 remove duplicate**。把重复的从sorted里移除（本身删）。如果cur跟他下一个node的值一样，就往下指，指到不一样，然后cur再往下指一个，前面的跟这个连上。

**CTCI 2.1 remove duplicate**把重复的从unsorted里移除本身不删）。用dict，用while如果dict里的key有跟某个node.value一样，就跳过这个node(node\_pre.next = node\_current.next).否则就接着走并且把值填到key里

**237 delete node 。**删掉某个value的node。node.val = node.next.val；node.next = node.next.next

**21 merge two LL**。把两个sorted的合并。cur指0头，然后比较两个list的头，谁小，cur.next就等于谁，并且谁往下走一个。

**160 intersection two LL**。两个LL不是一个头，但是从某个点开始就重合了，就那个点。先分别求len.然后求两个len的差，长的那个先走这个差，然后两个在一起往后走，知道value一样，就是那个点。

**86 partition list**。给一个value，比value大或等于的都跑到比value小的后面。比如1->4->3->2->5->2, x = 3return 1->2->2->4->3->5.先从头开始找到比value大的那个点，再从这个点开始往后没找到一个比value小的就放这个值前面。

**206 reverse LL** 。是把头一个个往尾巴上写入：先让last=None。cur=head, cur指向last，再把cur赋给last（就是last往前提一个）同时head指向下一个。

**234 palindrome LL**。fast，slow，rev一个走到尾一个走到中间一个把头半块的箭头反向了。然后rev和slow分别往下遍历，就是两个人从中间分别往两边走，看每一步的值都一样不。

**328 odd even LL**。让所有奇数点排前面，后边跟着偶数点。odd指向head,然后odd挪到这个点，even指向head.next,然后even挪到这个点。然后head移到第三个点，重复。

tree（13+1）

**100 same tree**。递归return p.data == q.data and self.isSameTree(p.left,q.left) and self.isSameTree(p.right,q.right)

**subtree** 。递归。一个树是不是另一个的子树，就是这个数跟另一个数的某个分支是不是same。所以边界条件就是check是不是same，return 是不是subtree

**101 symmetric tree** 。对称树。用stack，DFS.往stack里存node。先存root的左右孩子为一个element，然后pop出去，检查被pop出来的符合（两个node.val一样，或者都是空，后者一空一不空）的条件不。然后存（左左，右右）和（左右，右左）两个element，然后分别pop出去，直到循环到都是空。

**102 binary tree level order traversal**。BFS从上到下。temp存每一行的所有node，next存temp里的所有node的左右子节点，result把每次的都存进去积累所有的node

**107 binary tree level order traversal2**。BFS从下到上。用stack。往current里塞node，然后pop出来，每pop一个，就把他的左右子node加入到next里，pop空后，next这个list的val插到result头，next里的node整体加入current.然后继续pop.

**104 max depth**。递归。max=1+max(self.maxDepth(root.left), self.maxDepth(root.right))。条件是如果没有root，return 0，如果有root但是root的左右子树没有，return 1.

**111 min depth**。递归。min(self.minDepth(root.right),self.minDepth(root.left))+1。条件是如果没root，return 0。如果左右子树有一个没有，return左右子树的depth的和再加1. （因为dep指的是走到那个节点再也没有路了，所以一边走不下去，就要走另一边，并不是一边没有就走完了，之所以是相加关系，是因为有一边是0）

**110 balanced binary tree**。给一个binary tree,看他是不是height-balanced，就是高度差小于2.用两个函数，一个算height（递归，1 + max(self.getHeight(root.left), self.getHeight(root.right))）的一个判断是不是的(递归，左右子树高度差小于2，且左右子树都是balanced：abs(self.getHeight(root.left) - self.getHeight(root.right)) < 2 and self.isBalanced(root.left) and self.isBalanced(root.right))。

**112 path sum**。给一个binary tree和一个sum.问是否有一个path值加起来等于sum。递归。检验左右子树if self.hasPathSum(root.left,sum-root.val)==True。条件是，if root==None,return False, if root左右为none，root.val=sum return True

**113 path sum2**。跟上一题一样，用递归。不过需要stack存路径，如果meet sum了，就把这个stack存result里

**257 binary tree paths**。找到二叉树的所有path。用dfs + stack。用一个stack = [(root, "")] 每一个element存node和path，所以node, ls = stack.pop()。如果还有左右孩子，stack.append((node.right, ls+str(node.val)+"->"))如果没有左右孩子，res.append(ls+str(node.val))

**226 invert binary tree**。左右子树交换。递归。left = self.invertTree(root.left)；right = self.invertTree(root.right)；root.right = left；root.left = right

**235 lowest common ancestor in BST**。给一个BST和两个nodes找他俩的LCA。就是找到一个node.val在这两个node中间。如果都比root小就往左子树移动，如果都比root大就往右子树移动，直到“不都小”或“不都大”时，那个root就是要找的点。

**270 closest BST value**。给一个BST一个target，找到一个node.val最接近target。用while，因为是BST，如果target比root小,就去左子树找接近的，不断更新gap= abs(root.val - target)和root

stack

**20 valid parentheses**。括号合法性。建空stack,建dict key为右括号,value为左括号,遍历所给括号组,如果在value里(左),就append到stack里,如果在key里(右),就看它对应的value和stack里pop出来的那个一样不.

**155 min stack**。设计一个stack，可以push,pop,top和得到里面最小的元素。stack的每一个element由两个数组成（刚压入的数，现在最小的数）

**225 implement stack using queues**。用collections.deque()里的popleft方法pop出list[0]。从队头出来（self.stack.popleft()）再回队尾（stack.append），一直到队头为原来的最后一个数

**232 implement queue using stacks**。把一个stack POP出去塞入一个新的stack，全塞完，pop一下，就是那个头。

DP

**70 climbing stair**。爬 n 阶的楼梯,每次只能1蹬或2蹬,问一共多少种走法。递归：最后剩1蹬的走法，最后剩2蹬的走法，return climbStairs(n-1)+climbStairs(n-2)，条件是if n == 1:return1；if n == 2:return 2。DP:

# n = 1: 走法有: 1 :f\_1 = 1  
# n = 2: 走法有: 1+1; 2: f\_2 = 2  
# n = 3: 走法有: 1+1+1; 1+2; 2+1 : f\_3 = f\_1 + f\_2 = 3(先走1step,剩下的是f\_2,先走2step,剩下的是f\_1)  
# n = 4: 走法有: 1+1+1+1; 1+1+2; 1+2+1; 2+2; 2+1+1; : f\_4 =f\_3 + f\_2 = 5 (先走1step,剩下的是f\_3 = 3,先走是2step,剩下的是f\_2 = 2)  
#f\_n = f(n-1)+f(n-2)

**198 house robber**。抢劫房子,list里的数为房子里的钱,连着的房子不能抢,问最多能抢多少钱。其实就是在某个数组里面找一个序列， 序列里面的每一个元素都不能相邻，然后求其最大和。

# f\_0 = nums[0]  
# f\_1 = max(nums[0], nums[1])  
# f\_2 = max(f\_0 +nums[2],f\_1)  
# f\_3 = max(f\_1 +nums[3],f\_2)

# if f\_pre\_pre+nums[i] >= f\_pre: f\_now = f\_pre\_pre+nums[i]; else f\_now = f\_pre

**213. House Robber II**。这回房子排成圈。就是第一个和最后一个不能同时。求两种情况的最大：最后一个不算和第一个不算

**337. House Robber III**。这回房子排成binary tree。就是他个他的父节点及左右子节点不能同时。

**303 range sum query**。Given an integer array nums, 得到 i and j(inclusive)直接的sum。先写一个函数算0-i的sum。然后用0-j的sum减0-i的sum。

**276 - Paint Fence。刷n个栅栏，k种颜色，每种颜色不能挨着。问多少种方法。**