**1.两数之和**

**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 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。但是，数组中同一个元素不能使用两遍。**

**你可以按任意顺序返回答案。**

**示例 1：**

**输入：nums = [2,7,11,15], target = 9**

**输出：[0,1]**

**解释：因为 nums[0] + nums[1] == 9 ，返回 [0, 1] 。**

**示例 2：**

**输入：nums = [3,2,4], target = 6**

**输出：[1,2]**

**示例 3：**

**输入：nums = [3,3], target = 6**

**输出：[0,1]**

**### 代码**

**```***python*

class Solution(object):

    def twoSum(self, nums, target):

        """

        :type nums: List[int]

        :type target: int

        :rtype: List[int]

        """

      # 用len()方法取得nums列表的长度

        n = len(nums)

        # x取值从0一直到n（不包括n）

        for x in range(n):

            # y取值从x+1一直到n（不包括n）

            # 用x+1是减少不必要的循环,y的取值肯定是比x大

            for y in range(x + 1, n):

                # 假如 target-nums[x]的某个值存在于nums中

                if nums[y] == target - nums[x]:

                    # 返回x和y

                    return x, y

                    break

                else:

                    continue

**```**

#### [2. 两数相加](https://leetcode-cn.com/problems/add-two-numbers/)

难度中等5458

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。

****示例 1：****



****输入：****l1 = [2,4,3], l2 = [5,6,4]****输出：****[7,0,8]****解释：****342 + 465 = 807.

****示例 2：****

****输入：****l1 = [0], l2 = [0]****输出：****[0]

****示例 3：****

****输入：****l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]****输出：****[8,9,9,9,0,0,0,1]

# Definition for singly-linked list.

# class ListNode(object):

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution(object):

    def addTwoNumbers(self, l1, l2):

        """

        :type l1: ListNode

        :type l2: ListNode

        :rtype: ListNode

        """

        dummy = ListNode(None)             # 建立虚拟头结点

        temp = dummy                       # 临时指针

        carry = 0                          # 表示进位的变量

        while l1 or l2:                    # 遍历

            num = 0                        # 对应位数字的和

            if l1:

                num += l1.val              # 加上l1的数

                l1 = l1.next               # l1指针后移

            if l2:

                num += l2.val              # 加上l2的数

                l2 = l2.next               # l2指针后移

            num = num+carry                # 再加上进位的数，最终对应位置数的和

            carry = num //10               # 新的进位数

            num = num % 10                 # 个位数

            temp.next = ListNode(num)      # 链接起来

            temp = temp.next               # 变量后移

        if carry > 0:                         # 若最后最高位还有进位

            temp.next = ListNode(carry)       # 则链接起来

        return dummy.next

#### [3. 无重复字符的最长子串](https://leetcode-cn.com/problems/longest-substring-without-repeating-characters/)

难度中等4772

给定一个字符串，请你找出其中不含有重复字符的 ****最长子串****的长度。

****示例 1:****

****输入:**** s = "abcabcbb"****输出:**** 3 ****解释:**** 因为无重复字符的最长子串是 "abc"，所以其长度为 3。

****示例 2:****

****输入:**** s = "bbbbb"****输出:**** 1****解释:**** 因为无重复字符的最长子串是 "b"，所以其长度为 1。

****示例 3:****

****输入:**** s = "pwwkew"****输出:**** 3****解释:**** 因为无重复字符的最长子串是 "wke"，所以其长度为 3。

  请注意，你的答案必须是 ****子串**** 的长度，"pwke" 是一个子序列，不是子串。

****示例 4:****

****输入:**** s = ""****输出:**** 0

class Solution(object):

    def lengthOfLongestSubstring(self, s):

        """

        :type s: str

        :rtype: int

        """

        mapping = set()        # 定义哈希集合，记录字符是否出现过

        n = len(s)             # 字符串长度

        l, ans = -1, 0         # l表示左指针，ans表示最终结果

        for i in range(n):                      # 遍历

            if i != 0:                          # 滑动窗口

                mapping.remove(s[i - 1])

            while l + 1 < n and s[l + 1] not in mapping:

                mapping.add(s[l + 1])           # 移动左指针

                l += 1

            ans = max(ans, l - i + 1)           # 比较

        return ans

#### [4. 最长回文子串](https://leetcode-cn.com/problems/longest-palindromic-substring/)

难度中等3053

给定一个字符串 s，找到 s 中最长的回文子串。你可以假设 s 的最大长度为 1000。

****示例 1：****

****输入:**** "babad"****输出:**** "bab"****注意:**** "aba" 也是一个有效答案。

****示例 2：****

****输入:**** "cbbd"****输出:**** "bb"

class Solution(object):

    def longestPalindrome(self, s):

        """

        :type s: str

        :rtype: str

        """

        n = len(s)              # 字符串长度

        dp = [[False]\*n for \_ in range(n)]            # 初始化状态矩阵

        ans = ""

        for l in range(n):

            for i in range(n):

                j = i+l

                if j >= len(s):             # 越界跳出

                    break

                if l == 0:                  # 单个字符，就是一个回文串，为True

                    dp[i][j] = True

                elif l == 1:                # 相邻字符，相等就是一个回文串，

                    dp[i][j] = (s[i] == s[j])

                else:                       # 长度更长的字符串

                    dp[i][j] = (dp[i+1][j-1] and s[i] == s[j])

                if dp[i][j]:

                    ans = s[i:j+1]

        return ans

#### [5. Z 字形变换](https://leetcode-cn.com/problems/zigzag-conversion/)

难度中等954

将一个给定字符串根据给定的行数，以从上往下、从左到右进行 Z 字形排列。

比如输入字符串为 "LEETCODEISHIRING" 行数为 3 时，排列如下：

L C I R

E T O E S I I G

E D H N

之后，你的输出需要从左往右逐行读取，产生出一个新的字符串，比如："LCIRETOESIIGEDHN"。

请你实现这个将字符串进行指定行数变换的函数：

string convert(string s, int numRows);

****示例 1:****

****输入:**** s = "LEETCODEISHIRING", numRows = 3****输出:**** "LCIRETOESIIGEDHN"

****示例 2:****

****输入:**** s = "LEETCODEISHIRING", numRows = 4****输出:**** "LDREOEIIECIHNTSG"****解释:****

L D R

E O E I I

E C I H N

T S G

class Solution(object):

    def convert(self, s, numRows):

        """

        :type s: str

        :type numRows: int

        :rtype: str

        """

        end=''

        n=len(s)

        if numRows<=1:

            return s

        T=numRows\*2-2

        res=""

        for i in range(numRows):

            cnt=0

            while(1):

                if cnt\*T+i<len(s):

                    res+=s[cnt\*T+i]

                else:

                    break

                if i==0 or i==T/2:

                    cnt+=1

                    continue

                if cnt\*T+T-i<len(s):

                    res+=s[cnt\*T+T-i]

                else:

                    break

                cnt+=1

        return res

#### [6. 字符串转换整数 (atoi)](https://leetcode-cn.com/problems/string-to-integer-atoi/)

难度中等941

请你来实现一个 atoi 函数，使其能将字符串转换成整数。

首先，该函数会根据需要丢弃无用的开头空格字符，直到寻找到第一个非空格的字符为止。接下来的转化规则如下：

* 如果第一个非空字符为正或者负号时，则将该符号与之后面尽可能多的连续数字字符组合起来，形成一个有符号整数。
* 假如第一个非空字符是数字，则直接将其与之后连续的数字字符组合起来，形成一个整数。
* 该字符串在有效的整数部分之后也可能会存在多余的字符，那么这些字符可以被忽略，它们对函数不应该造成影响。

注意：假如该字符串中的第一个非空格字符不是一个有效整数字符、字符串为空或字符串仅包含空白字符时，则你的函数不需要进行转换，即无法进行有效转换。

在任何情况下，若函数不能进行有效的转换时，请返回 0 。

****提示：****

* 本题中的空白字符只包括空格字符 ' ' 。
* 假设我们的环境只能存储 32 位大小的有符号整数，那么其数值范围为 [−231,  231 − 1]。如果数值超过这个范围，请返回  INT\_MAX (231 − 1) 或 INT\_MIN (−231) 。

****示例 1:****

****输入:**** "42"****输出:**** 42

****示例 2:****

****输入:**** " -42"****输出:**** -42****解释:**** 第一个非空白字符为 '-', 它是一个负号。

  我们尽可能将负号与后面所有连续出现的数字组合起来，最后得到 -42 。

****示例 3:****

****输入:**** "4193 with words"****输出:**** 4193****解释:**** 转换截止于数字 '3' ，因为它的下一个字符不为数字。

****示例 4:****

****输入:**** "words and 987"****输出:**** 0****解释:**** 第一个非空字符是 'w', 但它不是数字或正、负号。

因此无法执行有效的转换。

****示例 5:****

****输入:**** "-91283472332"****输出:**** -2147483648****解释:**** 数字 "-91283472332" 超过 32 位有符号整数范围。

  因此返回 INT\_MIN (−231) 。

class Solution:

    def myAtoi(self, str):

        """

        :type str: str

        :rtype: int

        """

        str=str.strip()

        flag = ''

        if len(str)==0:

            return 0

        if len(str)==1:

            if '9'>=str>='0':

                return int(str)

            else:

                return 0

        if str[0] in '-+':

            flag=str[0]

            str=str[1:]

        result = ''

        for i in str:

            if '9'>=i>='0':

                result+=i

            else:

                break

        if result=='':

            return 0

        else:

            result = flag+result

        if -2\*\*31<=int(result)<=2\*\*31-1:

            return int(result)

        elif flag == '-':

            return -2\*\*31

        else:

            return 2\*\*31-1

#### [7. 回文数](https://leetcode-cn.com/problems/palindrome-number/)

难度简单1352

判断一个整数是否是回文数。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。

****示例 1:****

****输入:**** 121****输出:**** true

****示例 2:****

****输入:**** -121****输出:**** false****解释:**** 从左向右读, 为 -121 。 从右向左读, 为 121- 。因此它不是一个回文数。

****示例 3:****

****输入:**** 10****输出:**** false****解释:**** 从右向左读, 为 01 。因此它不是一个回文数。

class Solution:

    def isPalindrome(self, x):

        """

        :type x: int

        :rtype: bool

        """

        if str(x)[0]=='-':

            return False

        y=str(x)[::-1]

        if x==(int(y)):

            return True

        else:

            return False

#### [8. 盛最多水的容器](https://leetcode-cn.com/problems/container-with-most-water/)

难度中等2090

给你 n 个非负整数 a1，a2，...，an，每个数代表坐标中的一个点 (i, ai) 。在坐标内画 n 条垂直线，垂直线 i 的两个端点分别为 (i, ai) 和 (i, 0) 。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。

****说明：****你不能倾斜容器。

****示例 1：****



****输入：****[1,8,6,2,5,4,8,3,7]****输出：****49 ****解释：****图中垂直线代表输入数组 [1,8,6,2,5,4,8,3,7]。在此情况下，容器能够容纳水（表示为蓝色部分）的最大值为 49。

****示例 2：****

****输入：****height = [1,1]****输出：****1

****示例 3：****

****输入：****height = [4,3,2,1,4]****输出：****16

****示例 4：****

****输入：****height = [1,2,1]****输出：****2

class Solution(object):

    def maxArea(self, height):

        """

        :type height: List[int]

        :rtype: int

        """

        l = 0                                    # 左指针

        r = len(height)-1                        # 右指针

        temp = (r-l) \* min(height[l],height[r])        # 开始时，可容纳的水

        while l < r:                          # 遍历

            if height[l] < height[r]:         # 若左边界 < 右边界，则左边界右移

                l = l+1

                temp = max(temp,(r-l)\*min(height[l],height[r]))   # 右移后的可容纳的水

            if height[l] >= height[r]:        # 若左边界 >= 右边界，则右边界左移

                r = r-1

                temp = max(temp,(r-l)\*min(height[l],height[r]))   # 左移后的可容纳的水

        return temp

#### [9. 整数转罗马数字](https://leetcode-cn.com/problems/integer-to-roman/)

难度中等466

罗马数字包含以下七种字符： I， V， X， L，C，D 和 M。

****字符**** ****数值****

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

例如， 罗马数字 2 写做 II ，即为两个并列的 1。12 写做 XII ，即为 X + II 。 27 写做  XXVII, 即为 XX + V + II 。

通常情况下，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 不写做 IIII，而是 IV。数字 1 在数字 5 的左边，所表示的数等于大数 5 减小数 1 得到的数值 4 。同样地，数字 9 表示为 IX。这个特殊的规则只适用于以下六种情况：

* I 可以放在 V (5) 和 X (10) 的左边，来表示 4 和 9。
* X 可以放在 L (50) 和 C (100) 的左边，来表示 40 和 90。
* C 可以放在 D (500) 和 M (1000) 的左边，来表示 400 和 900。

给定一个整数，将其转为罗马数字。输入确保在 1 到 3999 的范围内。

****示例 1:****

****输入:**** 3****输出:**** "III"

****示例 2:****

****输入:**** 4****输出:**** "IV"

****示例 3:****

****输入:**** 9****输出:**** "IX"

****示例 4:****

****输入:**** 58****输出:**** "LVIII"****解释:**** L = 50, V = 5, III = 3.

****示例 5:****

****输入:**** 1994****输出:**** "MCMXCIV"****解释:**** M = 1000, CM = 900, XC = 90, IV = 4.

class Solution(object):

    def intToRoman(self, num):

        """

        :type num: int

        :rtype: str

        """

        m = [

            ['', 'M', 'MM', 'MMM'],

            ['', 'C', 'CC', 'CCC', 'CD', 'D', 'DC', 'DCC', 'DCCC', 'CM'],

            ['', 'X', 'XX', 'XXX', 'XL', 'L', 'LX', 'LXX', 'LXXX', 'XC'],

            ['', 'I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII', 'IX']

        ]

        d = [1000, 100, 10, 1]

        r=''

        for k,v in enumerate(d):

            r+=m[k][int(num/v)]

            num=num%v

        return r

#### [10. 最长公共前缀](https://leetcode-cn.com/problems/longest-common-prefix/)

难度简单1401

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 ""。

****示例 1:****

****输入:**** ["flower","flow","flight"]****输出:**** "fl"

****示例 2:****

****输入:**** ["dog","racecar","car"]****输出:**** ""****解释:**** 输入不存在公共前缀。

****说明:****

所有输入只包含小写字母 a-z 。

class Solution(object):

    def longestCommonPrefix(self, strs):

        """

        :type strs: List[str]

        :rtype: str

        """

        res = ""

        if len(strs) == 0:

            return ""

        for each in zip(\*strs):#zip()函数将原列表解压

            if len(set(each)) == 1:#利用集合判断是不是每个元素都相同

                res += each[0]

            else:

                return res

        return res

#### [11. 三数之和](https://leetcode-cn.com/problems/3sum/)

难度中等2859

给你一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a，b，c ，使得 a + b + c = 0 ？请你找出所有满足条件且不重复的三元组。

****注意：****答案中不可以包含重复的三元组。

****示例：****

给定数组 nums = [-1, 0, 1, 2, -1, -4]，

满足要求的三元组集合为：

[

[-1, 0, 1],

[-1, -1, 2]

]

class Solution(object):

    def threeSum(self, nums):

        """

        :type nums: List[int]

        :rtype: List[List[int]]

        """

        if len(nums) < 3:                 # 若数组长度 < 3,直接返回空

            return []

        nums = sorted(nums)               # 排序

        re = []                           # 结果变量

        for i in range(len(nums)):                  # 遍历

            if nums[i] > 0:                         # 若 nums[i] > 0,则没有数字之和为0，直接返回re

                return re

            if i > 0 and nums[i] == nums[i-1]:      # 排除重复数组

                continue

            L = i + 1                               # 左指针

            R = len(nums) - 1                       # 右指针

            while L < R:                                      # 遍历

                temp = nums[i] + nums[L] + nums[R]            # 三数之和

                if temp == 0:                                 # 若temp = 0

                    re.append([nums[i],nums[L],nums[R]])      # 添加到结果列表中

                    while L < R and nums[L] == nums[L+1]:     # 避免左指针下一个数字重复，跳过

                        L = L + 1

                    while L < R and nums[R] == nums[R-1]:     # 避免右指针下一个数字重复，跳过

                        R = R - 1

                    L = L + 1                       # 左指针右移，遍历

                    R = R - 1                       # 右指针左移，遍历

                elif temp > 0:                                # 若temp > 0

                    R = R - 1                                 # 则应该让三数之和小一些，故右指针左移

                else:

                    L = L + 1                                 # 反之，应该让三数之和大一些，故左指针右移

        return re

#### [12. 最接近的三数之和](https://leetcode-cn.com/problems/3sum-closest/)

难度中等650

给定一个包括 n 个整数的数组 nums 和 一个目标值 target。找出 nums 中的三个整数，使得它们的和与 target 最接近。返回这三个数的和。假定每组输入只存在唯一答案。

****示例：****

****输入：****nums = [-1,2,1,-4], target = 1****输出：****2****解释：****与 target 最接近的和是 2 (-1 + 2 + 1 = 2) 。

****提示：****

* 3 <= nums.length <= 10^3
* -10^3 <= nums[i] <= 10^3
* -10^4 <= target <= 10^4

class Solution(object):

    def threeSumClosest(self, nums, target):

        """

        :type nums: List[int]

        :type target: int

        :rtype: int

        """

        nums.sort()

        l=len(nums)

        if l==3:

            return nums[0]+nums[1]+nums[2]

        final=nums[0]+nums[1]+nums[2]

        for i in range(l-2):

            left=i+1

            right=l-1

            while 1:

                s=nums[i]+nums[left]+nums[right]

                if abs(s-target)<abs(final-target):

                    final=s

                if s>target:

                    right-=1

                elif s<target:

                    left+=1

                else:

                    return s

                if right<=left:

                    break

        return final

#### [13. 删除链表的倒数第N个节点](https://leetcode-cn.com/problems/remove-nth-node-from-end-of-list/)

难度中等1164

给定一个链表，删除链表的倒数第 n 个节点，并且返回链表的头结点。

****示例：****

给定一个链表: ****1->2->3->4->5****, 和 **n **= 2****.

当删除了倒数第二个节点后，链表变为 ****1->2->3->5****.

****说明：****

给定的 n 保证是有效的。

# Definition for singly-linked list.

# class ListNode(object):

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution(object):

    def removeNthFromEnd(self, head, n):

        """

        :type head: ListNode

        :type n: int

        :rtype: ListNode

        """

        if n == 0:                       # 若 n 等于0， 直接返回原链表

            return head

        dummy = ListNode(None)           # 建立虚拟头结点

        dummy.next = head

        slow = fast = dummy              # 快慢双指针

        for \_ in range(n):               # 循环，让快指针先走 n 步

            fast = fast.next

        while fast and fast.next:        # 快慢指针一起走

            slow = slow.next

            fast = fast.next

        slow.next = slow.next.next       # 跳过倒数第n个节点

        return dummy.next

#### [14. 有效的括号](https://leetcode-cn.com/problems/valid-parentheses/)

难度简单2071

给定一个只包括 '('，')'，'{'，'}'，'['，']' 的字符串，判断字符串是否有效。

有效字符串需满足：

1. 左括号必须用相同类型的右括号闭合。
2. 左括号必须以正确的顺序闭合。

注意空字符串可被认为是有效字符串。

****示例 1:****

****输入:**** "()"****输出:**** true

****示例 2:****

****输入:**** "()[]{}"****输出:**** true

****示例 3:****

****输入:**** "(]"****输出:**** false

****示例 4:****

****输入:**** "([)]"****输出:**** false

****示例 5:****

****输入:**** "{[]}"****输出:**** true

class Solution(object):

    def isValid(self, s):

        """

        :type s: str

        :rtype: bool

        """

        if s=='':

            return True

        l=len(s)

        left='({['

        right=')}]'

        flag=[]

        for i in range(l):

            if s[i] in left:

                if s[i]=='(':

                    flag.append(')')

                elif s[i]=='[':

                    flag.append(']')

                elif s[i]=='{':

                    flag.append('}')

            elif s[i] in right:

                if flag==[]:

                    return False

                if not s[i]==flag[-1]:

                    return False

                else:

                    flag.pop(-1)

        if flag==[]:

            return True

        else:

            return False

#### [15. 合并两个有序链表](https://leetcode-cn.com/problems/merge-two-sorted-lists/)

难度简单1458

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。

****示例：****

****输入：****1->2->4, 1->3->4****输出：****1->1->2->3->4->4

# Definition for singly-linked list.

# class ListNode(object):

#     def \_\_init\_\_(self, x):

#         self.val = x

#         self.next = None

class Solution(object):

    def mergeTwoLists(self, l1, l2):

        """

        :type l1: ListNode

        :type l2: ListNode

        :rtype: ListNode

        """

        if l1==None:

            return l2

        if l2==None:

            return l1

        if l1.val>l2.val:

            l3=l2

            l2=l2.next

        else:

            l3=l1

            l1=l1.next

        l4=l3

        while 1:

            if l1!=None and l2!=None:

                if l1.val>l2.val:

                    l3.next=l2

                    l3=l3.next

                    l2=l2.next

                else:

                    l3.next=l1

                    l3=l3.next

                    l1=l1.next

            elif l1==None and l2!=None:

                l3.next=l2

                break

            elif l1!=None and l2==None:

                l3.next=l1

                break

        return l4

#### [16. 删除排序数组中的重复项](https://leetcode-cn.com/problems/remove-duplicates-from-sorted-array/)

难度简单1769

给定一个排序数组，你需要在**[原地](http://baike.baidu.com/item/%E5%8E%9F%E5%9C%B0%E7%AE%97%E6%B3%95)** 删除重复出现的元素，使得每个元素只出现一次，返回移除后数组的新长度。

不要使用额外的数组空间，你必须在 **[原地](https://baike.baidu.com/item/%E5%8E%9F%E5%9C%B0%E7%AE%97%E6%B3%95)**修改输入数组****并在使用 O(1) 额外空间的条件下完成。

****示例 1:****

给定数组 nums = ****[1,1,2]****,

函数应该返回新的长度 ****2****, 并且原数组 nums 的前两个元素被修改为 **1**, **2**。

你不需要考虑数组中超出新长度后面的元素。

****示例 2:****

给定 nums = ****[0,0,1,1,1,2,2,3,3,4]****,

函数应该返回新的长度 ****5****, 并且原数组 nums 的前五个元素被修改为 **0**, **1**, **2**, **3**, **4**。

你不需要考虑数组中超出新长度后面的元素。

****说明:****

为什么返回数值是整数，但输出的答案是数组呢?

请注意，输入数组是以****「引用」****方式传递的，这意味着在函数里修改输入数组对于调用者是可见的。

你可以想象内部操作如下:

// ****nums**** 是以“引用”方式传递的。也就是说，不对实参做任何拷贝

int len = removeDuplicates(nums);

// 在函数里修改输入数组对于调用者是可见的。

// 根据你的函数返回的长度, 它会打印出数组中****该长度范围内****的所有元素。

for (int i = 0; i < len; i++) {

    print(nums[i]);

}

class Solution(object):

    def removeDuplicates(self, nums):

        """

        :type nums: List[int]

        :rtype: int

        """

        l=len(nums)

        if l==0 or l==1:

            return l

        flag=0

        for i in range(1,l):

            i-=flag

            if nums[i]==nums[i-1]:

                nums.pop(i)

                flag+=1

        return len(nums)

#### [17. 字符串相乘](https://leetcode-cn.com/problems/multiply-strings/)

难度中等540

给定两个以字符串形式表示的非负整数 num1 和 num2，返回 num1 和 num2 的乘积，它们的乘积也表示为字符串形式。

****示例 1:****

****输入:**** num1 = "2", num2 = "3"****输出:**** "6"

****示例 2:****

****输入:**** num1 = "123", num2 = "456"****输出:**** "56088"

****说明：****

1. num1 和 num2 的长度小于110。
2. num1 和 num2 只包含数字 0-9。
3. num1 和 num2 均不以零开头，除非是数字 0 本身。
4. ****不能使用任何标准库的大数类型（比如 BigInteger）****或****直接将输入转换为整数来处理****。

class Solution(object):

    def multiply(self, num1, num2):

        """

        :type num1: str

        :type num2: str

        :rtype: str

        """

        if num1[0]=='0' or num2[0]=='0':

            return '0'

        num1=num1[::-1]

        num2=num2[::-1]

        l1=len(num1)

        l2=len(num2)

        len\_of\_final=l1+l2

        final=['0' for i in range(len\_of\_final)]

        for i in range(l1):

            for j in range(l2):

                s=int(num1[i])\*int(num2[j])

                final[i+j]=str(s%10+int(final[i+j]))

                final[i+j+1]=str(s/10+int(final[i+j+1]))

        for i in range(len\_of\_final-1):

            s=int(final[i])

            final[i]=str(s%10)

            final[i+1]=str(s/10+int(final[i+1]))

        end=''

        for each in final:

            end+=each

        end=end[::-1]

        while 1:

            if end[0]=='0':

                end=end[1:]

            else:

                break

        return end

#### [18. 全排列](https://leetcode-cn.com/problems/permutations/)

难度中等1061

给定一个****没有重复**** 数字的序列，返回其所有可能的全排列。

****示例:****

****输入:**** [1,2,3]****输出:****

[

[1,2,3],

[1,3,2],

[2,1,3],

[2,3,1],

[3,1,2],

[3,2,1]

]

通过次数236,155提交次数305,143

class Solution(object):

    def permute(self, nums):

        """

        :type nums: List[int]

        :rtype: List[List[int]]

        """

        global ans

        global s

        ans=[]

        l=len(nums)

        for i in range(l):

            s=[]

            s.append(nums[i])

            rec=nums[i]

            nums.remove(rec)

            flag=s[:-1]

            self.find(nums)

            s=flag

            nums.insert(i,rec)

        return ans

    def find(self,nums):

        global ans

        global s

        l=len(nums)

        if l==0:

            tmp=list(s)

            ans.append(tmp)

        else:

            for i in range(l):

                s.append(nums[i])

                rec=nums[i]

                nums.remove(rec)

                flag=s[:-1]

                self.find(nums)

                s=flag

                nums.insert(i,rec)

#### [19. 螺旋矩阵](https://leetcode-cn.com/problems/spiral-matrix/)

难度中等577

给定一个包含 m x n 个元素的矩阵（m 行, n 列），请按照顺时针螺旋顺序，返回矩阵中的所有元素。

****示例 1:****

****输入:****

[

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

]****输出:**** [1,2,3,6,9,8,7,4,5]

****示例 2:****

****输入:****

[

[1, 2, 3, 4],

[5, 6, 7, 8],

[9,10,11,12]

]****输出:**** [1,2,3,4,8,12,11,10,9,5,6,7]

class Solution(object):

    def spiralOrder(self, matrix):

        """

        :type matrix: List[List[int]]

        :rtype: List[int]

        """

        self.asr=[]

        if matrix==[]:

            return []

        self.find\_outer(matrix)

        return self.asr

    def find\_outer(self,matrix):

        if len(matrix)==0:

            pass

        elif len(matrix)==1:

            self.asr.extend(matrix.pop(0))

        elif len(matrix[1])==1:

            for each in matrix:

                self.asr.extend(each)

        else:

            flag=False

            for each in matrix:

                if each!=[]:

                    flag=True

                    break

            if flag:

                self.asr.extend(matrix.pop(0))

                for each in matrix:

                    self.asr.append(each.pop(-1))

                flag=matrix.pop(-1)

                flag.reverse()

                self.asr.extend(flag)

                l=len(matrix)

                for i in range(l-1,-1,-1):

                    self.asr.append(matrix[i].pop(0))

                self.find\_outer(matrix)

#### [20. 旋转链表](https://leetcode-cn.com/problems/rotate-list/)

难度中等386

给定一个链表，旋转链表，将链表每个节点向右移动 k 个位置，其中 k 是非负数。

****示例 1:****

****输入:**** 1->2->3->4->5->NULL, k = 2****输出:**** 4->5->1->2->3->NULL****解释:****

向右旋转 1 步: 5->1->2->3->4->NULL

向右旋转 2 步: 4->5->1->2->3->NULL

****示例 2:****

****输入:**** 0->1->2->NULL, k = 4****输出:**** 2->0->1->NULL****解释:****

向右旋转 1 步: 2->0->1->NULL

向右旋转 2 步: 1->2->0->NULL

向右旋转 3 步: 0->1->2->NULL

向右旋转 4 步: 2->0->1->NULL

# Definition for singly-linked list.

# class ListNode(object):

#     def \_\_init\_\_(self, x):

#         self.val = x

#         self.next = None

import copy

class Solution(object):

    def rotateRight(self, head, k):

        """

        :type head: ListNode

        :type k: int

        :rtype: ListNode

        """

        if head==None or k==0 or head.next==None:

            return head

        length=0

        o\_head=copy.deepcopy(head)

        while head!=None:

            length+=1

            head=head.next

        head=copy.deepcopy(o\_head)

        k=k%length

        if k==0:

            return head

        for i in range(length-k):

            flag=head.next

            head=head.next

        o\_flag=flag

        while flag.next!=None:

            flag=flag.next

        flag.next=o\_head

        for i in range(length-k):

            flag=flag.next

        flag.next=None

        return o\_flag

#### [21. 合并两个有序数组](https://leetcode-cn.com/problems/merge-sorted-array/)

难度简单727

给你两个有序整数数组 nums1 和 nums2，请你将 nums2 合并到 nums1 中，使 nums1 成为一个有序数组。

****说明：****

* 初始化 nums1 和 nums2 的元素数量分别为 m 和 n 。
* 你可以假设 nums1 有足够的空间（空间大小大于或等于 m + n）来保存 nums2 中的元素。

****示例：****

****输入：****

nums1 = [1,2,3,0,0,0], m = 3

nums2 = [2,5,6], n = 3

****输出：****[1,2,2,3,5,6]

class Solution:

    def merge(self, nums1, m, nums2, n):

        nums1\_copy = nums1[:m]            # 将数组nums1复制到数组nums1\_copy

        nums1[:] = []                             # 清空数组nums1，用来存放结果

        p1 = 0                # 定义数组nums1\_copy指针

        p2 = 0               # 定义数组nums2指针

        while p1 < m and p2 < n:             # 任意数组循环遍历完，就跳出

            if nums1\_copy[p1] < nums2[p2]:          # 如果nums1\_copy的元素小于nums2的元素

                nums1.append(nums1\_copy[p1])      # 将小的元素添加到结果数组中

                p1 += 1                    # 指针右移，以便再次比较

            else:

                nums1.append(nums2[p2])                  # 将小的元素添加到结果数组中

                p2 += 1                    # 指针右移，以便再次比较

        if p1 < m:            # 如果数组nums1没有遍历完，说明剩下的都是大数，我们直接添加到结果数组中

            nums1[p1+p2:] = nums1\_copy[p1:]

        if p2 < n:            # 如果数组nums2没有遍历完，说明剩下的都是大数，我们直接添加到结果数组中

            nums1[p1 + p2:] = nums2[p2:]

        return nums1          # 返回结果

#### [22. 杨辉三角](https://leetcode-cn.com/problems/pascals-triangle/)

难度简单434

给定一个非负整数 numRows，生成杨辉三角的前 numRows 行。



在杨辉三角中，每个数是它左上方和右上方的数的和。

****示例:****

****输入:**** 5****输出:****

[

[1],

[1,1],

[1,2,1],

[1,3,3,1],

[1,4,6,4,1]

]

class Solution:

    def generate(self, num\_rows):             # num\_rows为行数

        triangle = []     # 定义结果列表

        for row\_num in range(num\_rows):                      # 遍历所有行：

            row = [None for \_ in range(row\_num+1)]      # 先写一个元素都为None的杨辉三角形

            row[0], row[-1] = 1, 1                         # 将每一行的两端变为1

            for j in range(1, len(row)-1):      # 填写中间元素

                row[j] = triangle[row\_num-1][j-1] + triangle[row\_num-1][j]

            triangle.append(row)            # 不断的将每一行放进结果列表中

        return triangle

#### [23. 买卖股票的最佳时机](https://leetcode-cn.com/problems/best-time-to-buy-and-sell-stock/)

难度简单1374

给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。

如果你最多只允许完成一笔交易（即买入和卖出一支股票一次），设计一个算法来计算你所能获取的最大利润。

注意：你不能在买入股票前卖出股票。

****示例 1:****

****输入:**** [7,1,5,3,6,4]****输出:**** 5****解释:**** 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。

注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格；同时，你不能在买入前卖出股票。

****示例 2:****

****输入:**** [7,6,4,3,1]****输出:**** 0****解释:**** 在这种情况下, 没有交易完成, 所以最大利润为 0。

class Solution(object):

    def maxProfit(self, prices):

        """

        :type prices: List[int]

        :rtype: int

        """

        if prices == None or len(prices) == 0:     # 若数组为空，则直接返回0

            return 0

        maxprices = 0           # 初始化最大利润为0

        tmpnum = prices[0]      # 定义随机变量初始值为数组的首元素，即买入股票的价格

        for i in prices:        # 循环

            if tmpnum > i:      # 找最小元素，即买入股票的价格

                tmpnum = i

            if i - tmpnum >= 0 and i - tmpnum > maxprices:    # 若利润（i-tmpnum）大于0，并且当前利润大于过去利润，则更新最大利润的值

                maxprices = i - tmpnum

        return maxprices

#### [24. 买卖股票的最佳时机 II](https://leetcode-cn.com/problems/best-time-to-buy-and-sell-stock-ii/)

难度简单1049

给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。

设计一个算法来计算你所能获取的最大利润。你可以尽可能地完成更多的交易（多次买卖一支股票）。

****注意：****你不能同时参与多笔交易（你必须在再次购买前出售掉之前的股票）。

****示例 1:****

****输入:**** [7,1,5,3,6,4]****输出:**** 7****解释:**** 在第 2 天（股票价格 = 1）的时候买入，在第 3 天（股票价格 = 5）的时候卖出, 这笔交易所能获得利润 = 5-1 = 4 。

  随后，在第 4 天（股票价格 = 3）的时候买入，在第 5 天（股票价格 = 6）的时候卖出, 这笔交易所能获得利润 = 6-3 = 3 。

****示例 2:****

****输入:**** [1,2,3,4,5]****输出:**** 4****解释:**** 在第 1 天（股票价格 = 1）的时候买入，在第 5 天 （股票价格 = 5）的时候卖出, 这笔交易所能获得利润 = 5-1 = 4 。

  注意你不能在第 1 天和第 2 天接连购买股票，之后再将它们卖出。

  因为这样属于同时参与了多笔交易，你必须在再次购买前出售掉之前的股票。

****示例 3:****

****输入:**** [7,6,4,3,1]****输出:**** 0****解释:**** 在这种情况下, 没有交易完成, 所以最大利润为 0。

****提示：****

* 1 <= prices.length <= 3 \* 10 ^ 4
* 0 <= prices[i] <= 10 ^ 4

class Solution(object):

    def maxProfit(self, prices):

        """

        :type prices: List[int]

        :rtype: int

        """

        if prices == None or len(prices) == 0:     # 若数组为空，则返回0

            return 0

        maxnum = 0

        for i in range(1,len(prices)):         # 循环

            if prices[i] > prices[i-1]:        # 若当前价格大于前面的价格，则有利润

                maxnum += prices[i] - prices[i-1]    # 不断累加

        return maxnum

#### [25. 最长连续序列](https://leetcode-cn.com/problems/longest-consecutive-sequence/)

难度困难639

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。

****进阶：****你可以设计并实现时间复杂度为 O(n) 的解决方案吗？

****示例 1：****

****输入：****nums = [100,4,200,1,3,2]****输出：****4****解释：****最长数字连续序列是 [1, 2, 3, 4]。它的长度为 4。

****示例 2：****

****输入：****nums = [0,3,7,2,5,8,4,6,0,1]****输出：****9

****提示：****

* 0 <= nums.length <= 104
* -109 <= nums[i] <= 109

class Solution:

    def longestConsecutive(self, nums):

        """

        :type nums: List[int]

        :rtype: int

        """

        nums = set(nums)

        res = 0

        for i in nums:

            if i-1 not in nums:

                y = i +1

                while y in nums:

                    y += 1

                res = max(res, y - i)

        return res

#### [26. 多数元素](https://leetcode-cn.com/problems/majority-element/)

难度简单828

给定一个大小为 n 的数组，找到其中的多数元素。多数元素是指在数组中出现次数 ****大于**** ⌊ n/2 ⌋ 的元素。

你可以假设数组是非空的，并且给定的数组总是存在多数元素。

****示例 1：****

****输入：****[3,2,3]****输出：****3

****示例 2：****

****输入：****[2,2,1,1,1,2,2]****输出：****2

 class Solution(object):

    def majorityElement(self, nums):

        """

        :type nums: List[int]

        :rtype: int

        """

        mid = len(nums) // 2     # 数组长度的一半

        dic = {}          # 定义字典

        for i in range(len(nums)):      # 循环遍历

            if nums[i] not in dic:      # 若元素不在字典中，添加到字典中

                dic[nums[i]] = 1

            else:                       # 若元素在字典中，其Value加一

                dic[nums[i]] += 1

            if dic[nums[i]] > mid:      # 如果某Value值大于mid，则为多数元素，

                return nums[i]

#### [27. 数字 1 的个数](https://leetcode-cn.com/problems/number-of-digit-one/)

难度困难188

给定一个整数 n，计算所有小于等于 n 的非负整数中数字 1 出现的个数。

****示例 1：****

****输入：****n = 13****输出：****6

****示例 2：****

****输入：****n = 0****输出：****0

 class Solution:

    def countDigitOne(self, n: 'int') -> 'int':

        if n<=0:

            return 0

        p,place,total=n,1,0

        while p>0:

            last=p%10

            p//=10

            total+=p\*place

            if last==1:

                total+=n%place+1

            elif last>1:

                total+=place

            place\*=10

        return total

#### [28. 丢失的数字](https://leetcode-cn.com/problems/missing-number/)

难度简单358

给定一个包含 [0, n] 中 n 个数的数组 nums ，找出 [0, n] 这个范围内没有出现在数组中的那个数。

****进阶：****

* 你能否实现线性时间复杂度、仅使用额外常数空间的算法解决此问题?

****示例 1：****

****输入：****nums = [3,0,1]****输出：****2**解释：**n = 3，因为有 3 个数字，所以所有的数字都在范围 [0,3] 内。2 是丢失的数字，因为它没有出现在 nums 中。

****示例 2：****

****输入：****nums = [0,1]****输出：****2**解释：**n = 2，因为有 2 个数字，所以所有的数字都在范围 [0,2] 内。2 是丢失的数字，因为它没有出现在 nums 中。

****示例 3：****

****输入：****nums = [9,6,4,2,3,5,7,0,1]****输出：****8**解释：**n = 9，因为有 9 个数字，所以所有的数字都在范围 [0,9] 内。8 是丢失的数字，因为它没有出现在 nums 中。

****示例 4：****

****输入：****nums = [0]****输出：****1**解释：**n = 1，因为有 1 个数字，所以所有的数字都在范围 [0,1] 内。1 是丢失的数字，因为它没有出现在 nums 中。

class Solution(object):

    def missingNumber(self, nums):

        """

        :type nums: List[int]

        :rtype: int

        """

        missing = len(nums)          # missing是数组的长度

        for i, num in enumerate(nums):     # 循环遍历，得到数组的index和value

            missing ^= i ^ num        # 数组元素和其对应下标进行异或，再和数组长度异或

        return missing                # 返回缺失值

#### [29. 移动零](https://leetcode-cn.com/problems/move-zeroes/)

难度简单901

给定一个数组 nums，编写一个函数将所有 0 移动到数组的末尾，同时保持非零元素的相对顺序。

****示例:****

****输入:**** [0,1,0,3,12]****输出:**** [1,3,12,0,0]

****说明****:

1. 必须在原数组上操作，不能拷贝额外的数组。
2. 尽量减少操作次数。

class Solution(object):

    def moveZeroes(self, nums):

        """

        :type nums: List[int]

        :rtype: None Do not return anything, modify nums in-place instead.

        """

        p = 0

        for i in range(len(nums)):

            if nums[i] != 0:

                nums[i],nums[p] = nums[p],nums[i]

                p += 1

#### [30. 第三大的数](https://leetcode-cn.com/problems/third-maximum-number/)

难度简单197

给定一个非空数组，返回此数组中第三大的数。如果不存在，则返回数组中最大的数。要求算法时间复杂度必须是O(n)。

****示例 1:****

****输入:**** [3, 2, 1]

****输出:**** 1

****解释:**** 第三大的数是 1.

****示例 2:****

****输入:**** [1, 2]

****输出:**** 2

****解释:**** 第三大的数不存在, 所以返回最大的数 2 .

****示例 3:****

****输入:**** [2, 2, 3, 1]

****输出:**** 1

****解释:**** 注意，要求返回第三大的数，是指第三大且唯一出现的数。

存在两个值为2的数，它们都排第二。

class Solution:

    def thirdMax(self, nums: List[int]) -> int:

        a=sorted(list(set(nums)))

        a=a[-1::-1]

        if len(a)>=3:

            return a[2]

        else:

            return max(a)