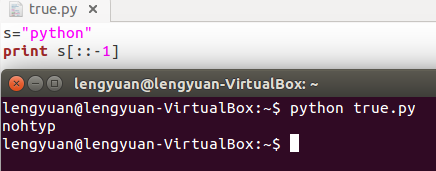
1. Two Sum

新建hash 数组存储数值和数值对应的下标；two pointer应该也可以

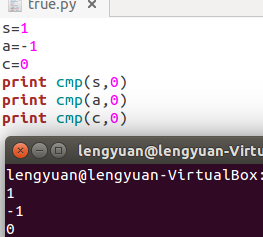
1. Reverse integer

Python 不支持字符串的替换，eg：s[1:4] = ‘A’

反转字符串：[::-1]; (列表list与字符串均可翻转)

eg:

cmp函数的使用：



Q：32-bit signed integer不应该是32位的带符号整数嘛？为什么用2\*\*31来限制而不是len（x）< 32-----懂了，划掉

注意！！！：input 和 output 都要被限制！

1. Roman to Integer

主要是罗马数字表的对应关系问题，应该有多种解法，以后有空再试

1. Longest common prefix

。。。。。。Input 为“ca”，“a”时没解决。。。之后再看

1. Remove duplicate from sorted array

用了del[], 时间99ms,能更快嘛

1. Remove element

用了remove（） //while true

1. Container with most water

If height[L] < height[R], move L, else move R. Say height[0] < height[5], area of (0, 4), (0, 3), (0, 2), (0, 1) will be smaller than (0, 5), so no need to try them.

主要就是算法思路的问题

15. 3sum

Two pointer，主要问题在处理重复（duplicate）上，可以转换为2sum；nums.sort()排序的思想；

16. 3sum closest

和15类似，set two pointer at i+1 and len(nums)-1, 注意循环要放在while内进行；无穷大的表示：float(‘inf’)，之后的414也有用到

1. Third maximum number

Easy

283. Moving zero

用了remove() and append(), 不难

1. Missing number

哇，真滴厉害了这算法，n（n+1）/2 -sum（nums）

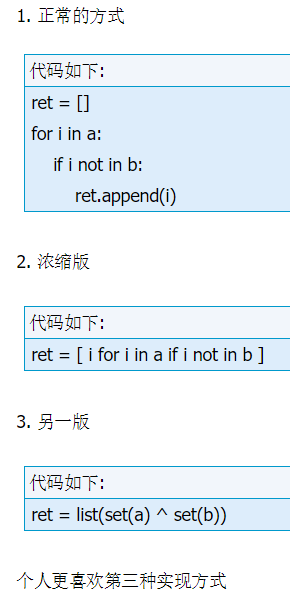
1. Contains Duplicate

运用了set()函数，set会将一个list转换为set；set() 函数之后也能运用到去重里。 Eg：print list(set(nums))

1. Contains Duplicate II

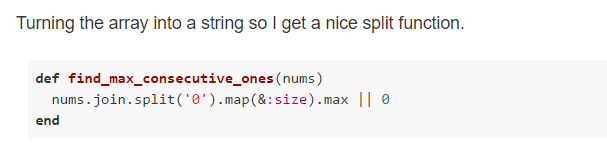
运用了dict， 同时用enumerate函数可以比for循环更快；两个连续的if判断的时候使用and连接更好

1. Find All Numbers Disappeared in an Array

求两个list差集的3种方法

1. Max Consecutive Ones

统计方法简单；将数组转换成字符串的方法回头再看看



1. Maximum Subarray

主要是算法思想，sum += nums[i], 每当sum<0时重置sum（一旦sum小于0说明这一部分subarray肯定不是maximum subarrary），sum = max（sum，0）

1. Plus One

Pow(a,b)函数表示a的b次方；一开始没看懂要求， 其实就是给一个用array表示的非负整数（4321用[4,3,2,1]表示），通过4\*1000+3\*100+2+10+1 =4321后加1再以数组的形式输出

1. Best Time to Buy and Sell Stock

没什么注意的

1. Best Time to Buy and Sell Stock II

求差值即可

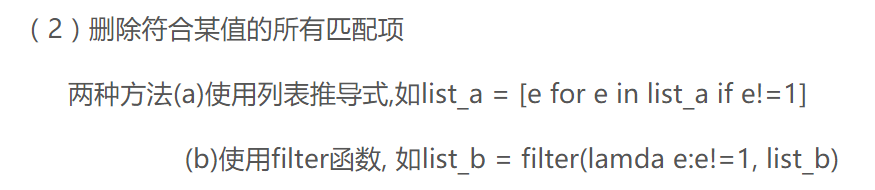
1. Pascal's Triangle

Pascal = [[1]\*(i+1) for i in range()], pascal[i][j] = pascal[i-1][j]+pascal[i-1][j-1] ; 另一种思路，帕斯卡三角每一行都可以看做C(n,m)

1. Pascal's Triangle II

阶乘没实现，代码有问题，再说

1. Majority Element



实测好像filter函数运行速度还更慢

1. Add Binary

Int(‘10’,8) = 8

1. Detect Capital

自己做得时候用的if循环；word.isupper()检测是否由大写字母构成, word.islower()检测小写,word.istitle()检测是否首字母大写

1. Number of Segments in a String

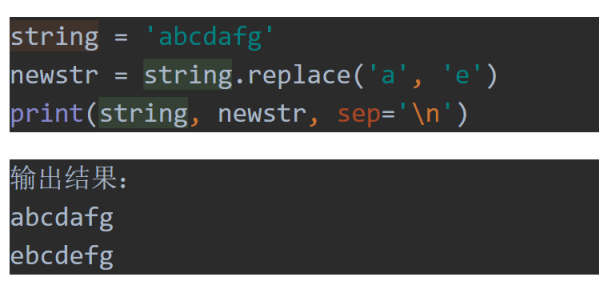
常规思路做得，速度beats 92%，but 10 lines；还可以用len(s.split())直接一步出结果，速度慢，code简洁 ##要查下split（）函数的使用方法

1. Ransom Note

Set()和count(), easy

1. Reverse Vowels of a String

python中修改字符串的三种方法：

1. 使用字符串的replace函数
2. 
3. 
4. Repeated Substring Pattern

主要是算法思路的问题；any（）函数

1. Valid Parentheses

Replace（）函数是一种解法，速度慢；更优解：创建stack = [], dict = {'(':')','[':']','{':'}'}, 还使用了pop()

1. Implement strStr()

理解了题意就没难度

125. Valid Palindrome

isalnum()函数用于判断字符串是否由字母及数字构成，是-true，否-false

1. Length of Last Word

Strip()函数用于移除字符串头尾指定的字符，默认为空格；Python split()通过指定分隔符对字符串进行切片，如果参数num 有指定值，则仅分隔 num 个子字符串

1. Count and say

还没搞懂，要再看看

1. Power of Two

Easy

1. Ugly Number

if num <= 0:

return False

ugly = [2,3,5]

for p in ugly:

while num % p == 0:

num /= p

return (num == 1) 在while的循环顺序上卡了很久，后来发现算法没理解透

1. Minimum Moves to Equal Array Elements

转换思路很重要！！！single-decrement such that all equals current minimum

1. Palindrome Number

数学方法，翻转（比较）一半即可；比较一个数左右两边的通用解法

1. Arranging Coins

可以转换为求解ax^2 + bx + c = 0 的 root，速度最快

367. Valid Perfect Square

可以用binary search的思想来解；<< 和 >> 都是位运算，左移一位末位补0.右移一位相当于除以2

1. Sqrt(x)

和367题非常相似

1. Power of Three

The positive divisors of 319 are exactly the powers of 3 from 30 to 319. That's all powers of 3 in the possible range here (signed 32-bit integer). So just check whether the number is positive and whether it divides 319.

204. Count Primes

埃拉托色尼筛选法(the Sieve of Eratosthenes)。。。惊了！

<https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes>

1. Happy Number

注意控制不是happy number情况下的输出false的条件；第一遍用的array来存储出现过的数，之后考虑用下hash结构，spce和time均为O（1）

1. Factorial Trailing Zeroes

其实就是在求5的总个数

1. Add Digits

Digital Root

this method depends on the truth:

N=(a[0] \* 1 + a[1] \* 10 + ...a[n] \* 10 ^n),and a[0]...a[n] are all between [0,9]

we set M = a[0] + a[1] + ..a[n]

and another truth is that:

1 % 9 = 1

10 % 9 = 1

100 % 9 = 1

so N % 9 = a[0] + a[1] + ..a[n]

means N % 9 = M

so N = M (% 9)

**as 9 % 9 = 0,so we can make (n - 1) % 9 + 1 to help us solve the problem when n is [9.as](http://9.as/) N is 9, ( 9 - 1) % 9 + 1 = 9**

1. Add Strings（感觉有点难，虽然是easy）

用了ord()和divmod(); 也可以考虑只用ord()和chr()

1. Excel Sheet Column Number

ABC = 1\*26^2 + 2\*26^1 + 3\*26^0

1. Excel Sheet Column Title

要注意正好被26整除的数；可以考虑用num-1%26 生成range 0-25避开26

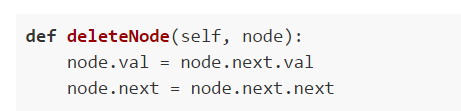
1. Linked List Cycle

It's not safe to use a step size bigger than 2 because then the fast pointer could "jump over" the slow one.

On the other hand, using the step size of 2, the pointers are guaranteed to collide. Suppose the loop is of length L and nodes are indexed starting at 0. Then when the slow pointer is at location i with i >= L / 2, the fast pointer is at (2 \* i) % L.

In particular, when the slow pointer is at location L - 1, the fast pointer is at (2 \* (L - 1)) % L = L - 2. It follows that at the next iteration of the while loop, both pointers will arrive back at the head simultaneously --- no matter the length L.

1. Delete Node in a Linked List



83. Remove Duplicates from Sorted List

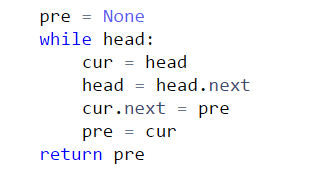
用于排除input为空的情况，linked list 中若只用node.next有可能出现node不存在（空集）的情况会报错

1. Remove Linked List Elements

First we remove all (if any) target nodes from the beginning (we do it because the removing logic is slightly different from when the node is not in the head). After that we just loop over all nodes, if the next one is one that should be removed, just get it out of the list by moving the next pointer to the next-next node. Otherwise just move along the list.(head元素要单独考虑)

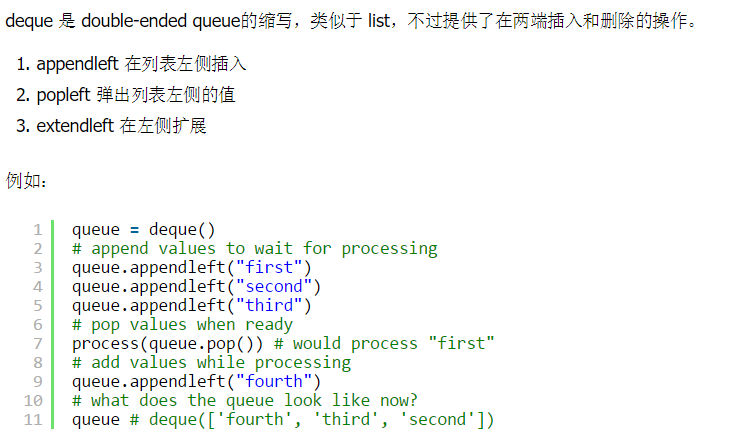
1. Reverse Linked List

循环和迭代2种做法，常用循环；循环法类似于3个指针，cur，head和pre（slow, head, fast），



225.Implement Stack using Queues

deque结构可以看作是内置的list结构的加强版,且比队列提供了更强大的方法,下面就通过几个小例子来详解Python的collections模块中的deque双端队列结构:



1. Min Stack

Pair store，eg：push时候append(x,curmin),避免最后一位是最小值的时候pop()操作可能会导致的问题

1. Implement Queue using Stacks

用两个堆栈（FILO）实现队列（FIFO），一个进一个出，两次先进后出就是先进先出

150. Evaluate Reverse Polish Notation

****Postfix to Evaluation****

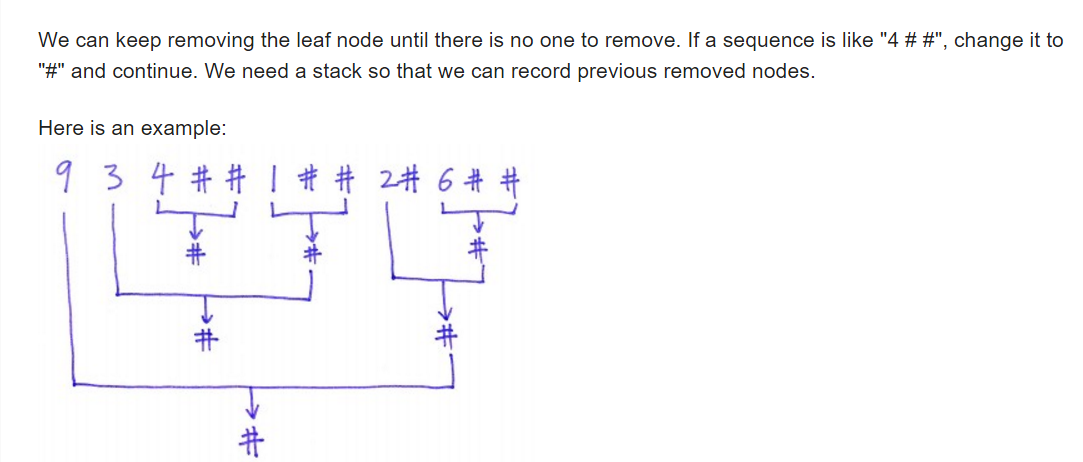
* (2+5)4 = 2,5,+,4,
* Operand: Push on the stack
* Operator: pop 2 from stack and apply operation.
* Corner case = division. -2/3? Python integer division with negative numbers gives weird results. Safest to use will be int(x/float(y))

1. Decode String

Concatenate digits and alphabets. When [ appears, push the current alphas and digits (as an integer n) into the stack, and start a new concatenation for alphas and digits. When a ] appears, pop the stack and extend the popped alphas n times to the current alphas.

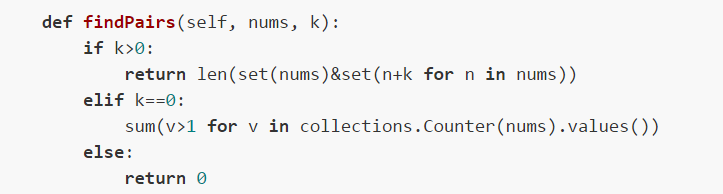
Complexity is linear in both time and space.

1. Verify Preorder Serialization of a Binary Tree



1. K-diff Pairs in an Array

这个算法厉害了！Amazing！

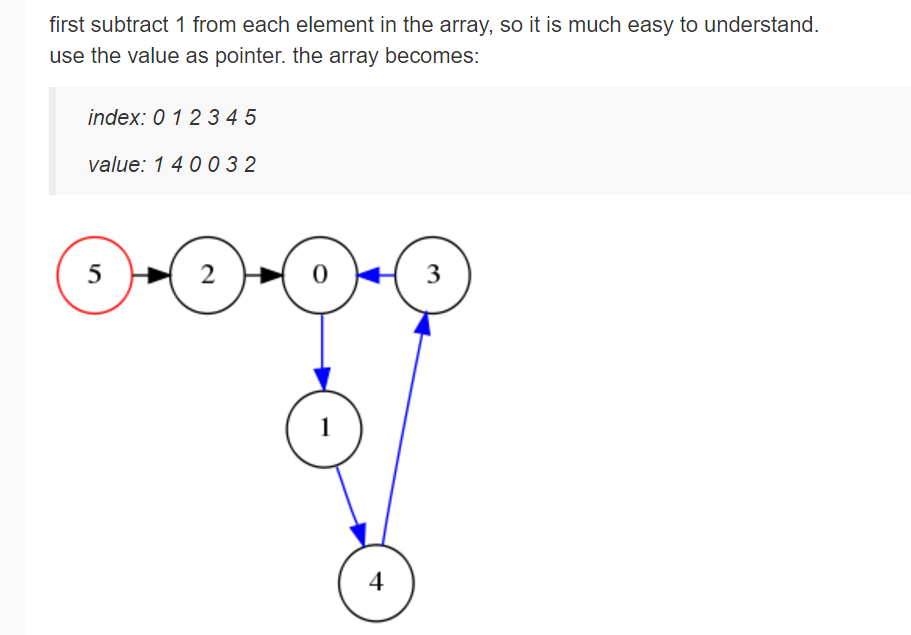


1. Rotate Array
2. Find the Duplicate Number

两种方法：1、two pointer 更类似于detect the cycle in the linked list的算法

****Find the starting point of the cycle in a linked list with a cycle****

1. Model the problem that way. Make sure you remember the solution for that problem.
2. nums[a] = b means a.next = b
3. Example: [2,5,1,1,4,3]. Model a graph using the indices of this array. If there is a duplicate b, then we will have a1.next = b and a2.next = b and a3.next = b .....Sketch a graph for the above.
4. Apply the linked list cycle detection algorithm here.



2、binary search

****Use a binary search variant****

1. The array is not sorted - but the indices of the array are sorted - #Insight
2. Find the mid index. Call it M.
3. Traverse the main array and count all numbers <= M. Say the count is K.
4. Say N = 200. Num elements = 201. M = 100. Low=1. High = 200.
5. Say K = 101. Then we know original list of 201 elements has 101 elements <= 100. There must be a duplicate from 1 to 100. ****So we should check in left half of index space - remember we are searching index space not element space****. high = mid.
6. Say K = 100. Then we know original list of 201 elements has 100 elements <= 100. It has 101 elements > 100 or from 101 to 200. So we should search right half of the index space: ****remember we are searching index space not element space**** low =mid+1
7. Triangle

二维数组，动态规划；注意头尾2个node要单独分情况讨论

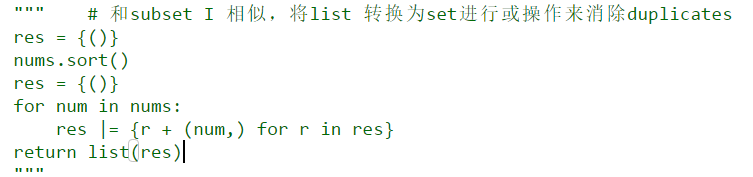
106. Construct Binary Tree from Inorder and Postorder Traversal

105. Construct Binary Tree from Preorder and Inorder Traversal

The idea is to find the root first, and then recursively build each left and right subtree

1. Subsets II

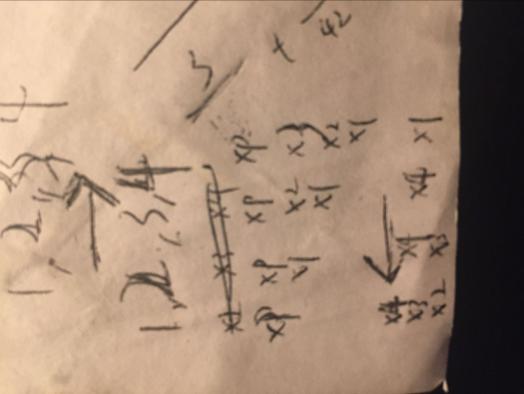
。。。。。。。。。。。。注意括号。。。用[] 别用（）。。。。。。



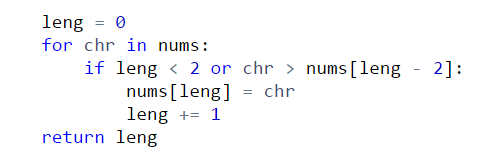
Method 2：写在leetcode上了，特别注意j的取值！

78. Subsets 同上

238. Product of Array Except Self

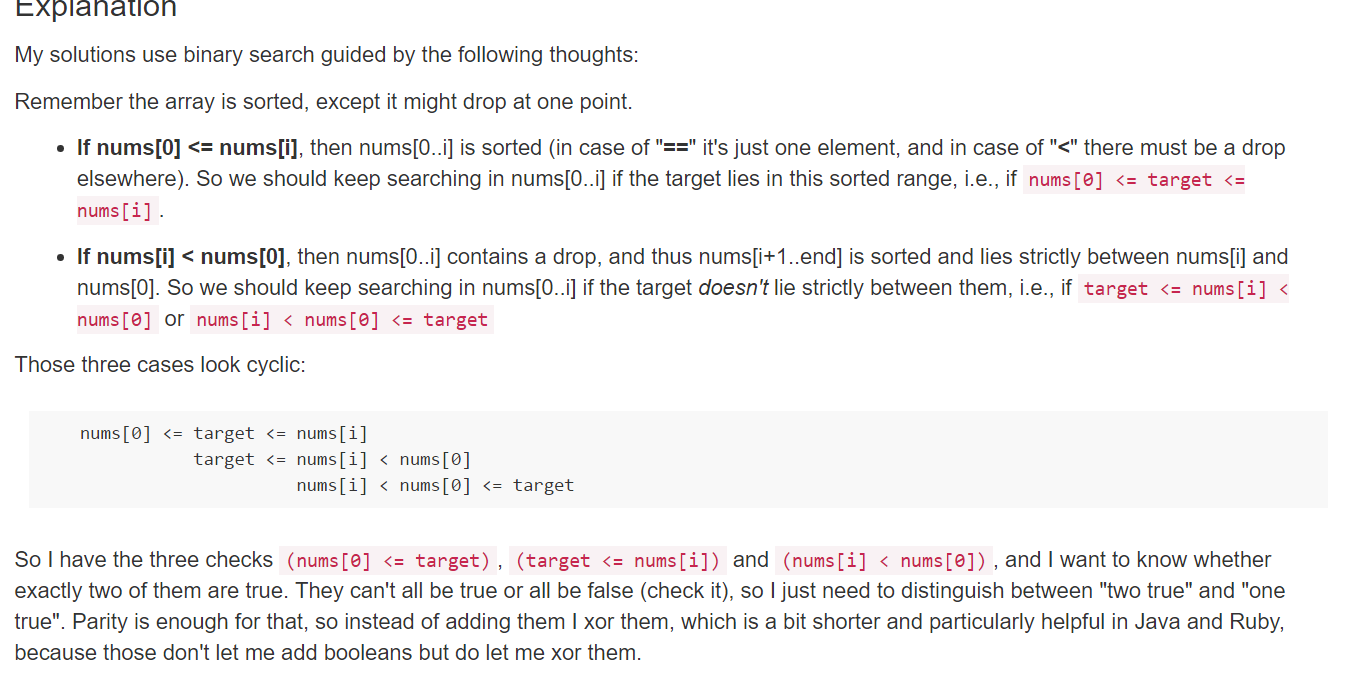
先从左向右，再从右向左，空间复杂度O(1)

1. Remove Duplicates from Sorted Array II

space complexity-O(1),chr对应的nums[i]必然快于nums[i] = chr 这步操作之后的nums，所以循环不会受到影响

顺便找人看下自己的程序有啥问题，感觉应该是对的啊

33. Search in Rotated Sorted Array



74. Search a 2D Matrix

就是正常的二分查找运用到二维数组；因为是sorted数组，所以也可以干当成一个大的list来做二分查找，应该还更方便

1. Set Matrix Zeroes

主要是要注意当每次找到0后，同行与同列的非0数要其他字符（eg：‘x’）来代替，以免对matrix中其他的0造成影响，最后再把所有‘x’换成0即可

64. Minimum Path Sum

Dynamic programming（DP solution），注意最左边和最上边要单独分情况讨论

62. Unique Paths

数学排列组合题

1. Unique Paths II

动态规划，和64非常相似

1. Spiral Matrix

其实不难，就是有点绕，自己看leetcode上的程序吧，注意edge的端点取值，一开始取错了各种出错

1. Jump Game

逆向思考，check last point 是否能回到起点，不难(第一次发自己的solution，激动！！！)

1. Merge Intervals

用2个数组分别存储start 和 end，之后再进行sort，然后再用while循环来判断是否有overlap；

Solution里面看到的对intervals针对start time进行sort操作的算法（语句）：

**if** intervals == []:

**return** []

intervals.sort(key = **lambda** x: x.start)

result = [intervals[0]]

**for** i **in** range(1, len(intervals)):

i1, i2 = result[-1], intervals[i]

**if** i2.start > i1.end:

result.append(i2)

**elif** i2.end >= i1.end:

i1.end = i2.end

**return** result

1. Rotate Image

Zip函数的运用！要再看看zip这个函数；

**[StefanPochmann](https://discuss.leetcode.com/user/stefanpochmann) 的7种解法**<https://discuss.leetcode.com/topic/15295/seven-short-solutions-1-to-7-lines>

1. Find All Duplicates in an Array

Sort一下之后就很简单了；第一眼看的时候满脑子都是Binary search的做法，但是懒得写code=。 =就用direct的方法了

1. Teemo Attacking

有点意思，但是很简单

1. Search for a Range

Basic Binary Search；由于二分查找很好，所以很多语言中的标准库都有二分查找的模块，这题也可以选择直接调用 bisect模块

1. L = [1,3,3,6,8,12,15]
2. x = 3
3. x\_insert\_point = bisect.bisect\_left(L,x)　　#在L中查找x，x存在时返回x左侧的位置，x不存在返回应该插入的位置..这是3存在于列表中，返回左侧位置１
4. **print** x\_insert\_point
5. x\_insert\_point = bisect.bisect\_right(L,x)  #在L中查找x，x存在时返回x右侧的位置，x不存在返回应该插入的位置..这是3存在于列表中，返回右侧位置３
6. **print** x\_insert\_point
7. x\_insort\_left = bisect.insort\_left(L,x)  #将x插入到列表L中，x存在时插入在左侧
8. **print** L
9. x\_insort\_rigth = bisect.insort\_right(L,x) #将x插入到列表L中，x存在时插入在右侧
10. **print** L

结果：

1  
3  
[1, 3, 3, 3, 6, 8, 12, 15]  
[1, 3, 3, 3, 3, 6, 8, 12, 15]

1. First Bad Version

基本的二分查找，就是在做得时候忽略了考虑lo <= hi的情况

1. Guess Number Higher or Lower

同样是简单的Binary Search

1. Sort Colors

自己做得时候用了3个array来分别存储3种color，用到了pop(), append()，但是感觉space complexity有点高；更trick一点直接用count计算个数再用temp array过度

1. Summary Ranges

挺简单的就注意下一些特殊的取值情况就好

1. Insert Delete GetRandom O(1)

难点主要在于remove函数的设计；（1）直接用remove - 600~800ms（2）用set - 400ms （3）exm？？？别人的hashmap <200ms，我的hashmap > 400ms？？？

1. Find Peak Element

Return low 比return mid不容易出错，下次要注意

1. Majority Element II

主要是python里自动取int值使得范围取值有点蛋疼，注意范围的取值问题即可；java中要事先定义数据类型应该就不存在这类问题

1. Reverse String II

翻转字符串的变形而已，不难

1. Multiply Strings

不能直接将input转为整数就按位转就是，不难

1. Minimum Time Difference

同样是sort solution，看到一个

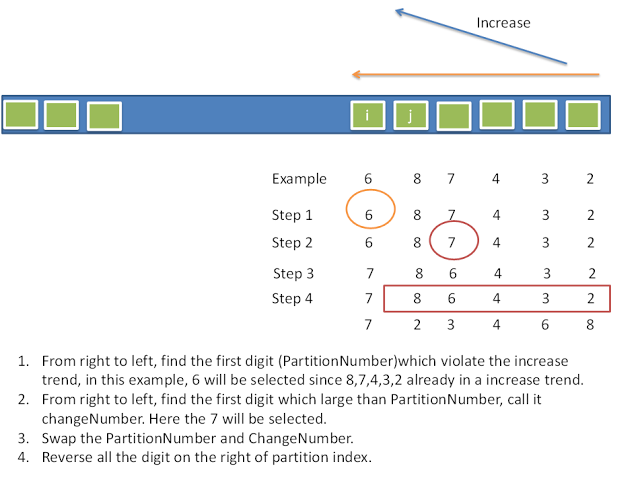
22. Generate Parentheses

Backtracking，答案大致能看懂，但是不太理解怎么实现的，要找人问问！！！

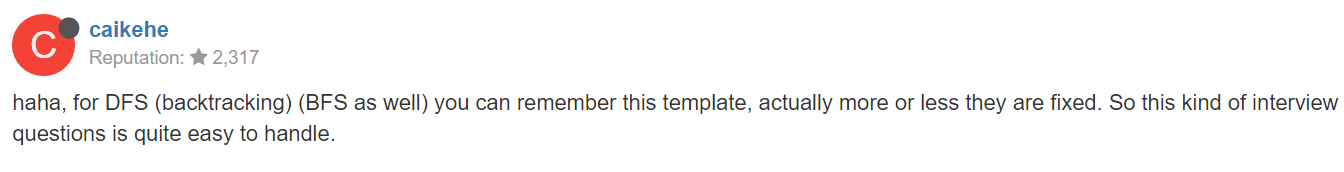
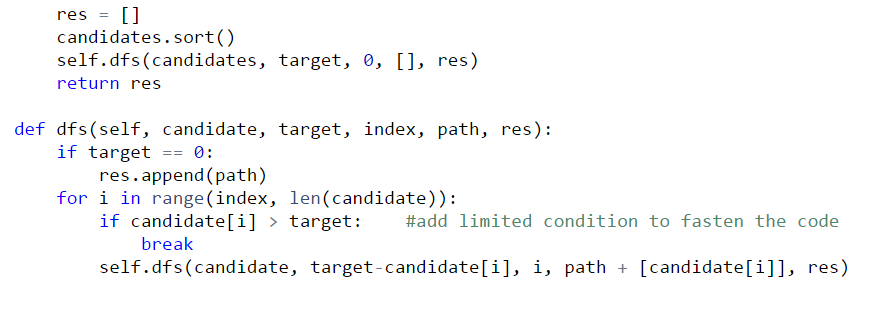
1. Two Sum II - Input array is sorted

要做到O（n）就用enumerate函数遍历和字典dictionary

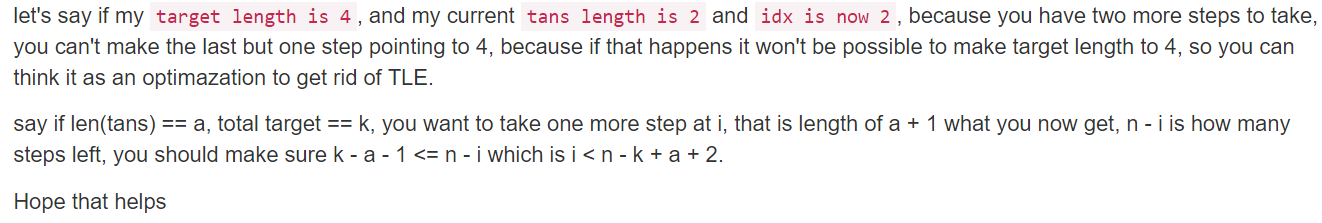
1. Next Permutation

主要是理解题意，惊了！

39. Combination Sum



1. Combinations
2. 直接用combinations函数（2）DFS（backtracking） method



1. Permutations

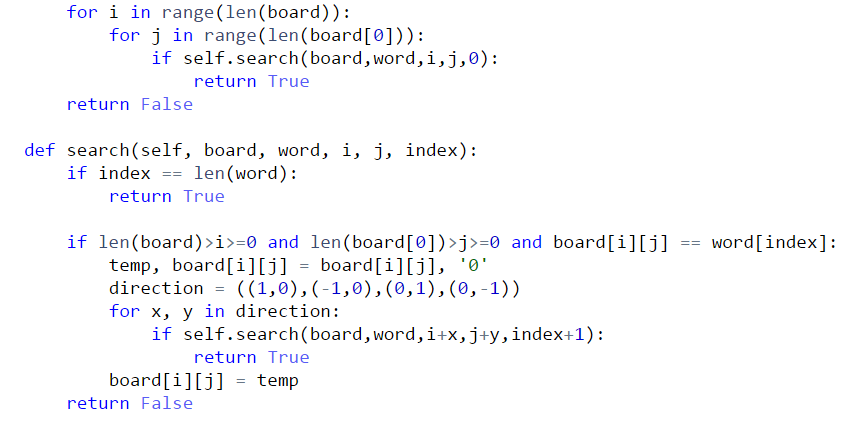
用的依然是39题那个结构！[A general approach to backtracking questions (Subsets, Permutations, Combination Sum, Palindrome Partioning)](https://discuss.leetcode.com/topic/46162/a-general-approach-to-backtracking-questions-in-java-subsets-permutations-combination-sum-palindrome-partioning" \t "https://leetcode.com/problems/permutations/" \l "/_blank)

1. Combination Sum III

DFS, 77题combination的简单变形

1. Word Search

依旧是DFS的思路，相当于每一对（i，j）都是一个DFS的起始点，注意之后的temp和direction的使用（我就是卡在这2点上），temp用于标记visited point，以免出现loop调用的情况(i,j)->(i+1,j)->(i,j)



1. Single Number

Solution 1：sum函数，数学问题 return sum(set(nums))\*2-sum(nums)

Solution 2：

1. Island Perimeter

****Count all grids and neighbors: O(MN)****

* For every grid which is 1, the contribution to the perimeter is 4 - x where x is the number of neighbors which are 1 as well.

1. Most Frequent Subtree Sum

DFS和字典的使用；

frequent = max(dic.values()) ##输出字典中频率最高的值

for chr in dic:

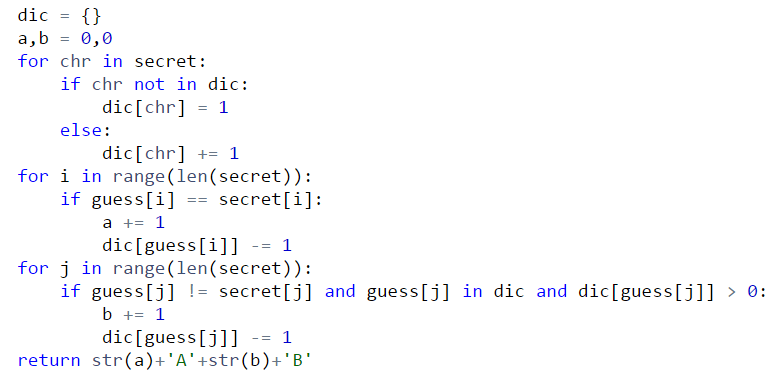
if dic[chr] == frequent:

output.append(chr)

return output

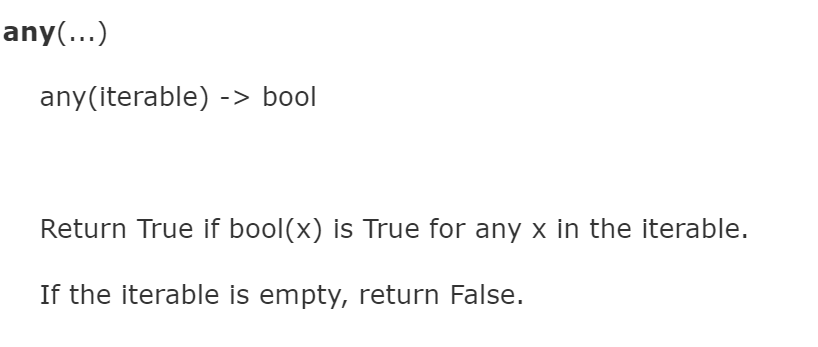
1. Bulls and Cows

基本的状况不难，主要是要注意有重复的情况，eg：（secret：1234，guess：1110），先将1234存入字典中，{1:1,2:1,3:1,4:`1}，然后guess中有相同的就减一，不同的就看是否在字典中并注意key值对应的values

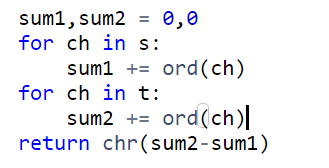


1. Keyboard Row

Solution1：用issubset函数判断一个set是否是另一个set的子集

Solution2：用all（）函数满足每一个字符都在同一行内即可存入res中

1. Find the Difference

solution1：，difference；主要是利用了字符与ASCII码的相互转换

Solution2：与解法一类似，但是将26个字母存入dictionary中且values不同；or用字典来存储s中字母的个数，然后t中一一减去，最后剩余的字母即为所加字母

Solution3：XOR

1. Intersection of Two Arrays

简单的将2个数组的set相与即可， return list(set(nums1)&set(nums2))

1. Intersection of Two Arrays II

242. Valid Anagram easy！只需要将s中字母的个数存入字典再减去t中字母的数量，均为0则return True

409. Longest Palindrome

290. Word Pattern

438. Find All Anagrams in a String

Counter类的目的是用来跟踪值出现的次数。它是一个无序的容器类型，以字典的键值对形式存储，其中元素作为key，其计数作为value。计数值可以是任意的Interger（包括0和负数）。这题可以直接用counter函数

Python Sliding Window Solution with counter: Maintain a window of len(p) in s, and slide to right until finish. Time complexity is O(len(s)).

1. Number of Boomerangs

没啥好说的，brute force method， O（n^2）

1. Isomorphic Strings

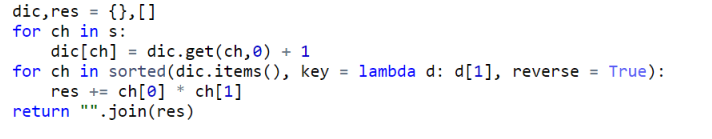
还是hash table的变形题，将s中元素作为key，t中元素作为value，最后检验元素是否都互异（我引入了一个exist数组存储已有的values）、

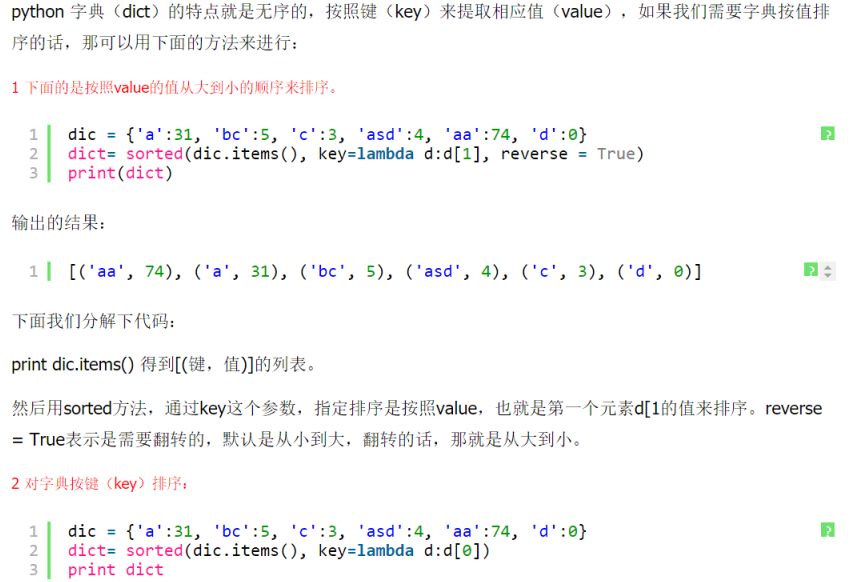
1. Sort Characters By Frequency

### 字典的排序

sorted(dic,value,reverse)

* dic为比较函数，value 为排序的对象（这里指键或键值），
* reverse：注明升序还是降序，True--降序，False--升序（默认）





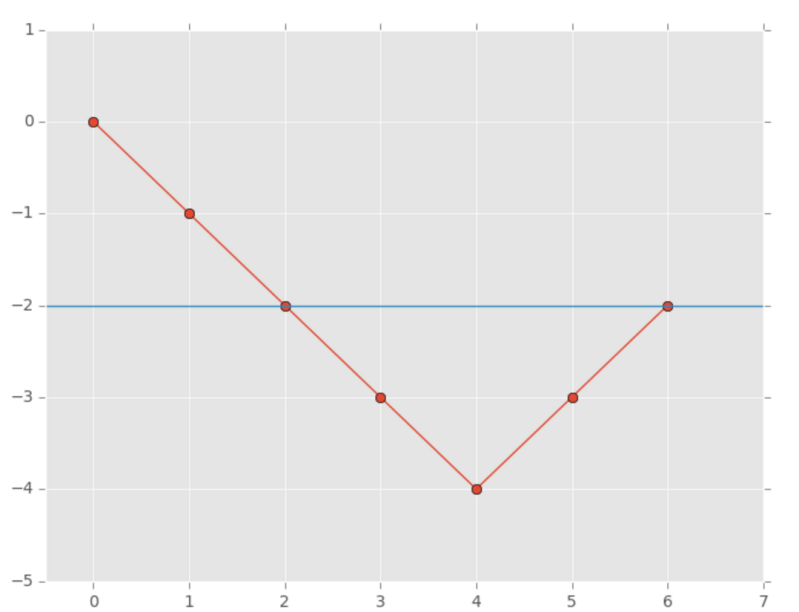
347. Top K Frequent Elements

和上题十分相似，需要对字典排序

1. Binary Tree Inorder Traversal

两种方法：（1）recursive - DFS （2）iterative - stack，迭代有点没看明白

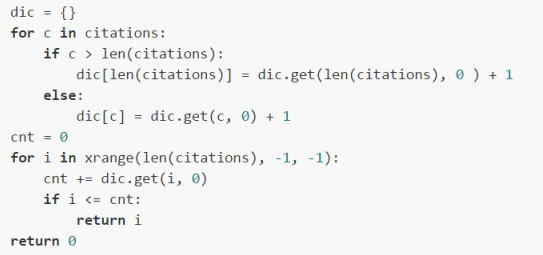
1. Contiguous Array



To find the maximum length, we need a dict to store the value of count (as the key) and its associated index (as the value). We only need to save a count value and its index at the first time, when the same count values appear again, we use the new index subtracting the old index to calculate the length of a subarray.

1. Valid Sudoku

分三种情况讨论，分辨检验row、column、grid是否满足有没有重复的数

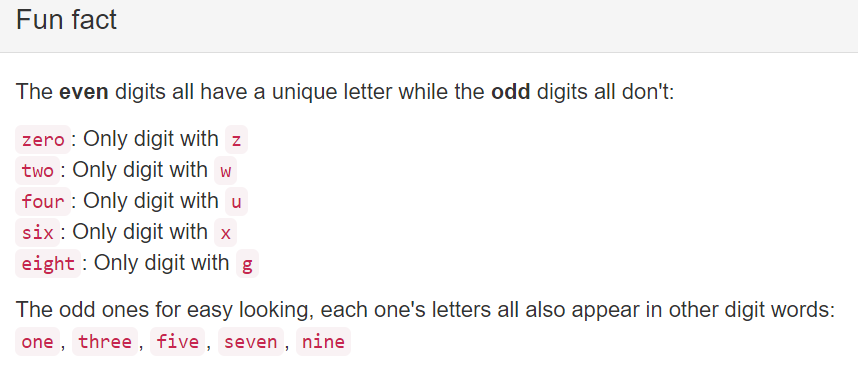
1. H-Index N个数中找出最大的h使得这N个数中有h个数大于等于h ----- sol#1：进行一次排序后就非常简单 SOL #2: 用字典
2. Group Anagrams

key = "".join(sorted(ch))，再将key作为字典的key，ch作为字典的value，速度会比直接str（sorted（ch））快很多

1. Repeated DNA Sequences

基本的hash table（字典）或者直接用set()， 不难

1. Longest Substring Without Repeating Characters
2. Reconstruct Original Digits from English

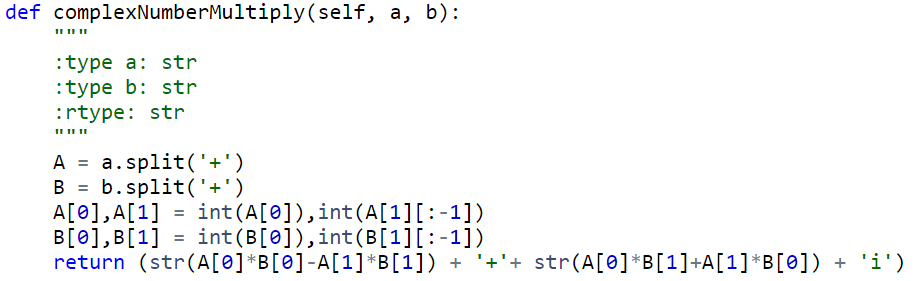


1. Perfect Number

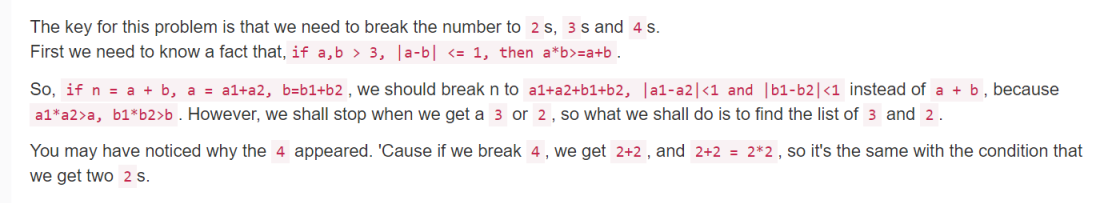
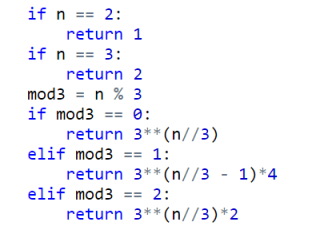
注意不要忽略num小于0以及num等于1的情况

1. Complex Number Multiplication

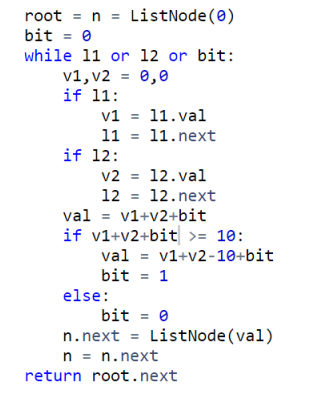
题目里已经帮我们限制了input会写成a+bi的模式，已经算是简化了题目，以后遇到复数也都可以自己写成这种模式使得运算简便



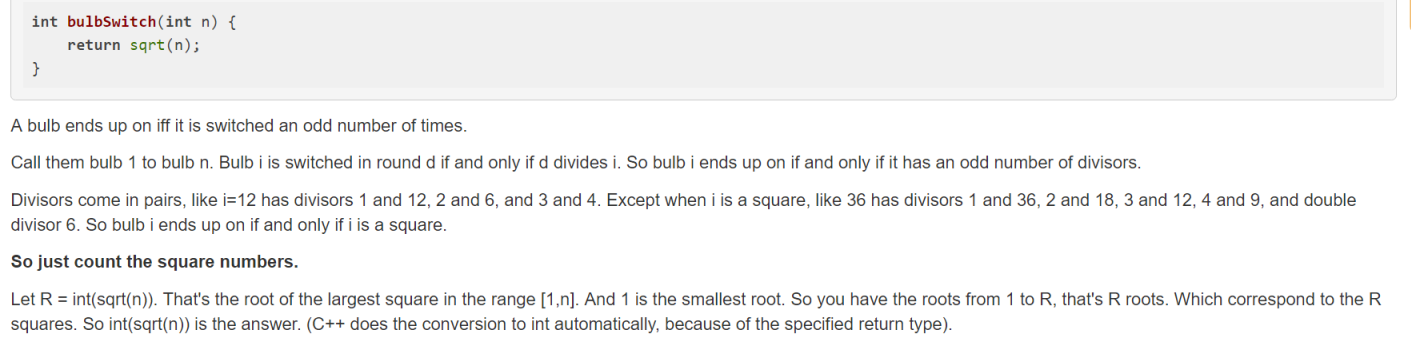
1. Integer Break

2. Add Two Numbers

主要还是链表的内容不够熟悉

1. Bulb Switcher

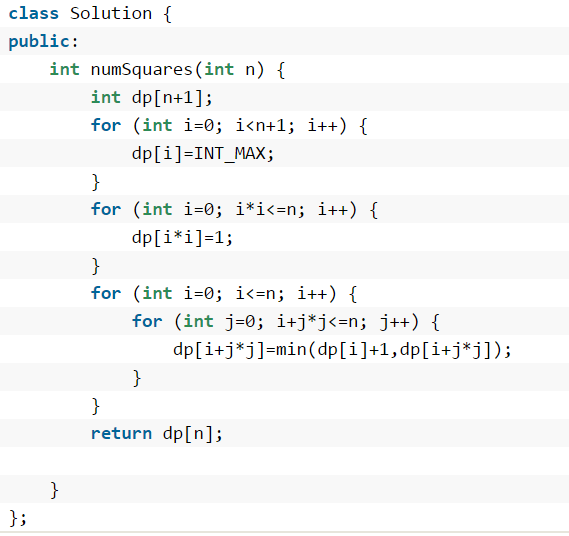
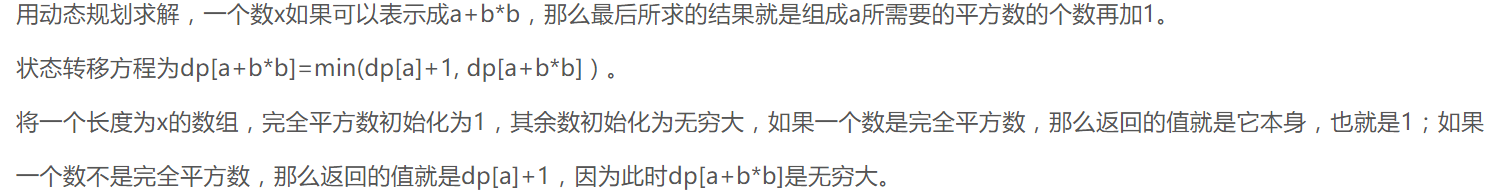


1. Minimum Moves to Equal Array Elements II

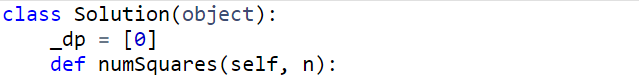
要注意最小的移动次数是移动至全为中位数的情况而不是平均数！！！

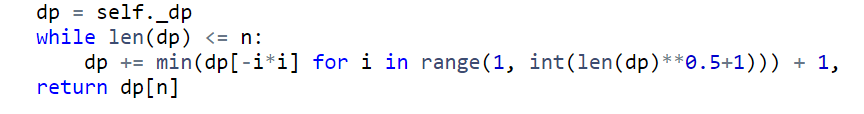
1. Perfect Squares

Dynamic programming 动态规划的题目都可以尝试先写出状态转移方程

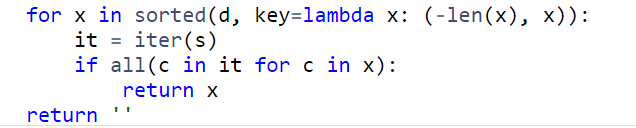


Python 可以用dp[-i^2] 来表示dp[i+j\*j]，and pyrhon方法使用了static DP solution，只需要call一次DP便会生成一个table，之后不需要再次调用这个函数



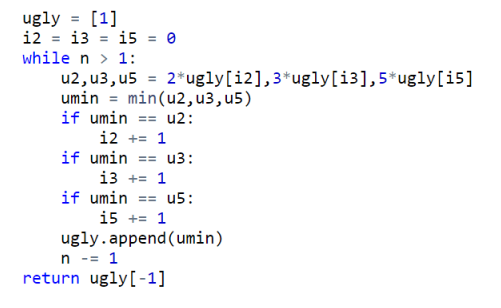


1. Longest Word in Dictionary through Deleting



这里iter()函数的使用极大的提高了程序的运行速度，迭代器可以保证it的顺序！！！

264. Ugly Number II

Sol #1:注意这边要用3个if，自己做得时候因为用了if。。elif。。Else导致了错误

1. Rectangle Area

a1 = abs((C - A) \* (D - B))

a2 = abs((G - E) \* (H - F))

overlap = max((min(C,G)-max(A,E)),0)\*max((min(D,H)-max(B,F)),0)

return a1 + a2 - overlap

纯数学问题就很烦=。 =

1. Divide Two Integers

这题主要是用straightforward的方法会超时；每次操作将除数\*2会极大的加快运算速度

for example, if we want to calc (17/2)

ret = 0;

17-2 ,ret+=1; left=15

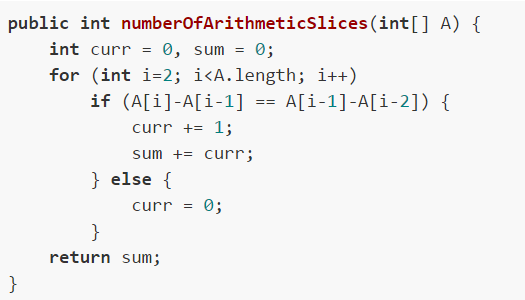
15-4 ,ret+=2; left=11

11-8 ,ret+=4; left=3

3-2 ,ret+=1; left=1

ret=8;

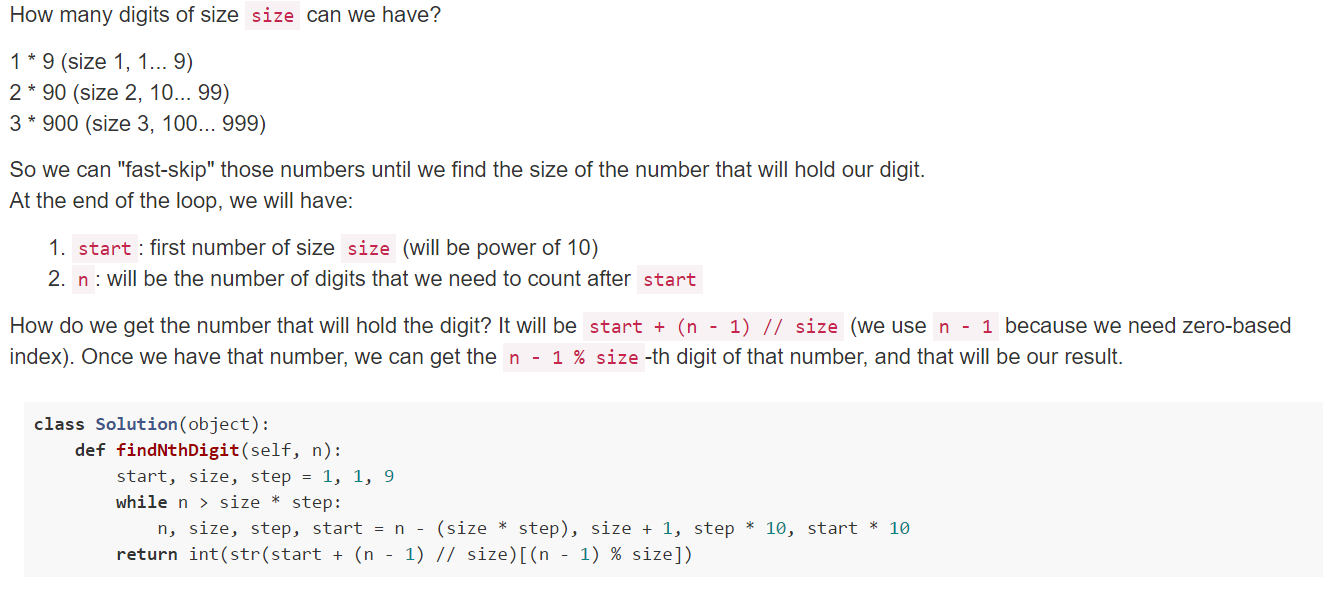
1. Arithmetic Slices

Straightforward方法还是很慢，虽然能Accepted；DP solution：java的code，但是python的思路也是类似的

1. Count Numbers with Unique Digits

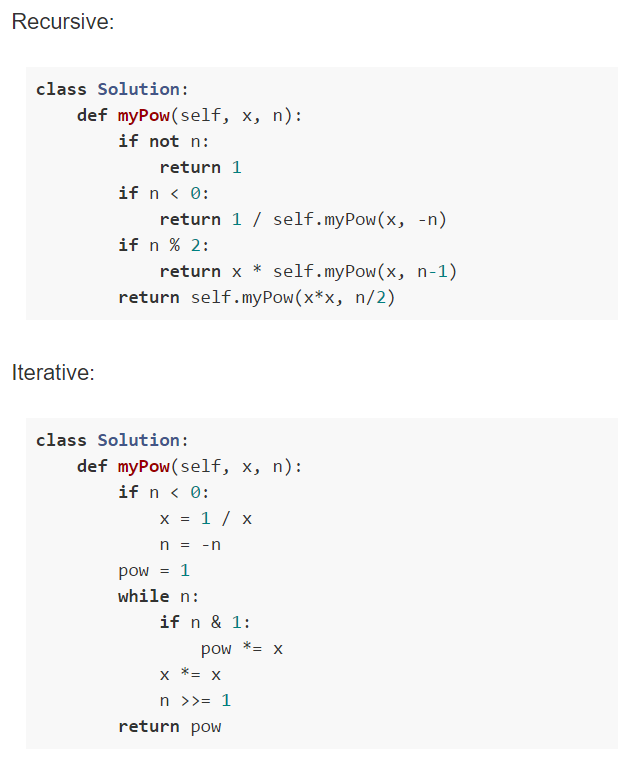
将n=1和n=2这两种情况单独拿出来计算就好，简单的递推

400. Nth Digit



1. Pow(x, n)

上手的时候完全没想到这居然是道位运算的题，利用位运算来使得time complexity 为O（log（n））；虽然这题用pow（x，n）和x\*\*n的tricky方式都能accepted



1. Integer Replacement

2 solutions: iterative(math) and recursive; 递归方法的速度会慢很多，

class Solution(object):

def integerReplacement(self, n):

"""

:type n: int

:rtype: int

"""

""" #math approach

im = 0

while n != 1:

if n % 2 == 0:

n = n / 2

else:

if n % 4 == 1 or n == 3: #数学推导。。。Wtf

n -= 1

else:

n += 1

im += 1

return im

""" #recursive method is much slower than iterative method

return self.replace(n,0)

def replace(self,n,im):

if n == 1:

return im

elif n % 2 == 0:

n /= 2

im += 1

return self.replace(n,im)

else:

im += 1

return min(self.replace(n+1,im),self.replace(n-1,im))

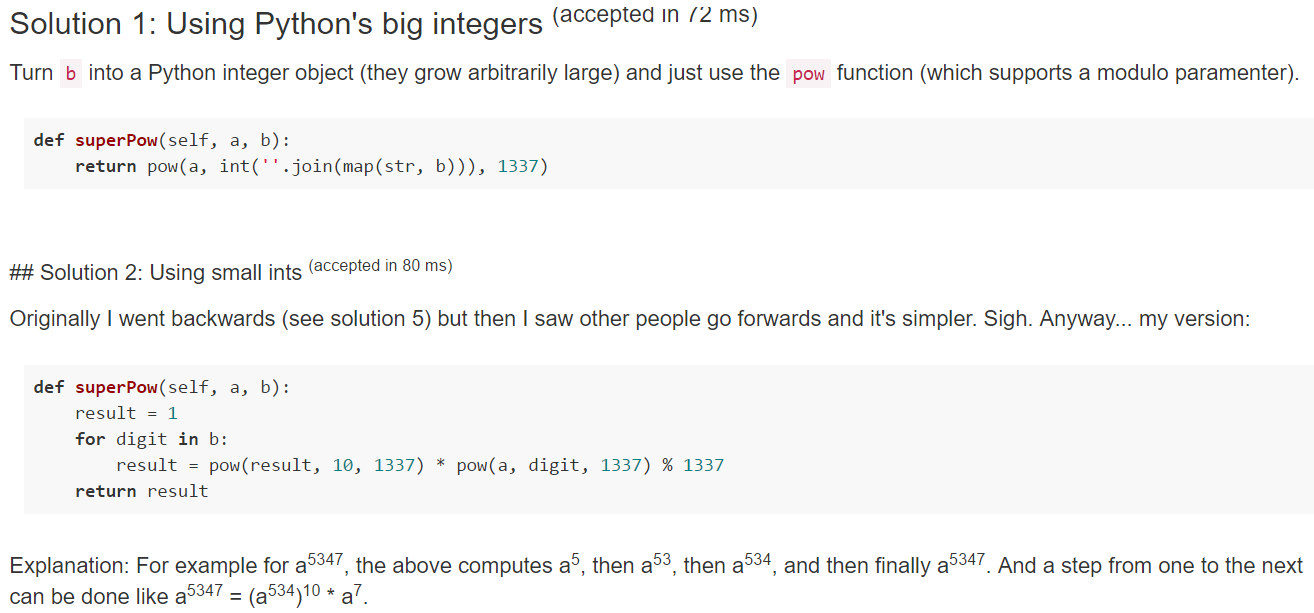
1. Rotate Function

Straightforward方法不可取，复杂度O（n^2）;找规律，仔细观察可以发现F(k)=F(k-1)+sum-len(A)\*A[len(A)-k]

1. Super Pow

map()函数接收两个参数，一个是函数，一个是序列，map将传入的函数依次作用到序列的每个元素，并把结果作为新的list返回。Eg：#使用lambda函数>>> print map(lambda x: x % 2, range(7))

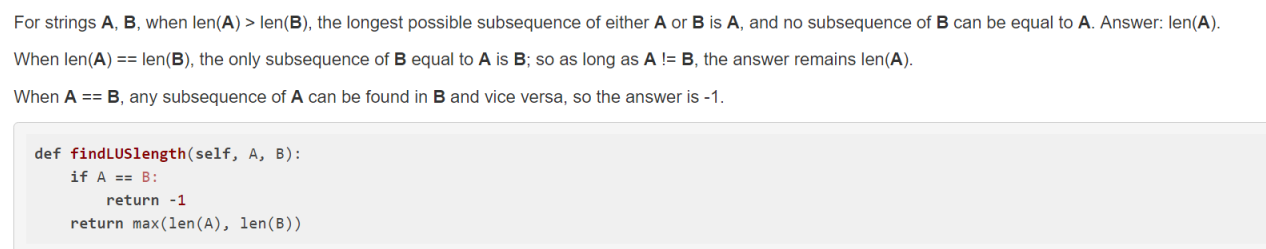
[0, 1, 0, 1, 0, 1, 0]



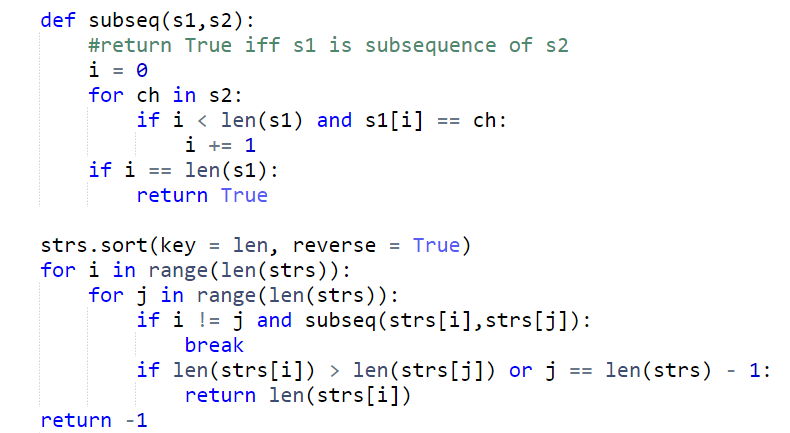
1. Student Attendance Record I

注意审题就好，不难

1. Longest Uncommon Subsequence I



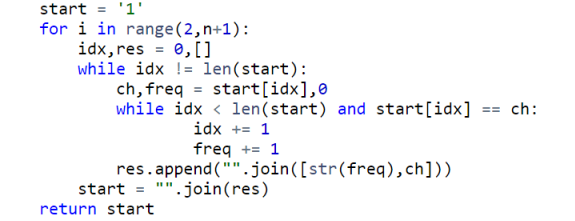
522. Longest Uncommon Subsequence II

主要是需要定义一个判断是否subsequence的子函数

1. Count and Say

1, 11, 21, 1211, 111221, ...

算法想的出来是怎样的。。。但是写出来总是出错。。。Ri



1. Reverse Words in a String

split()函数括号内为空时表示默认所有空字符，包括空格，换行符(\n)、制表符(\t)等

1. Reverse Words in a String III ---- easy, 简单的” ”.join(x[::-1] for x in s.split())
2. ZigZag Conversion
3. Compare Version Numbers
4. Longest Substring Without Repeating Characters

used,start,res = {},0,0

for i in range(len(s)):

if s[i] in used and start <= used[s[i]]: #之前一直出问题主要是这边没限制start<=used[s[i]]

start = used[s[i]]+1

else:

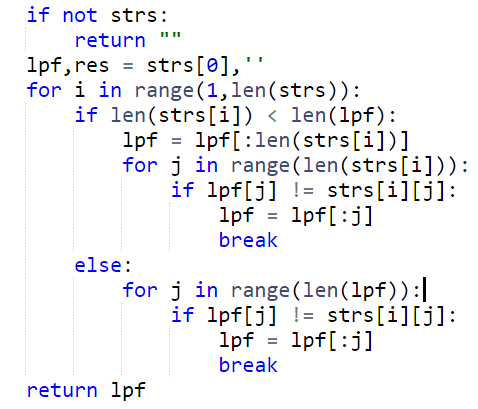
res = max(res,i-start+1)

used[s[i]] = i

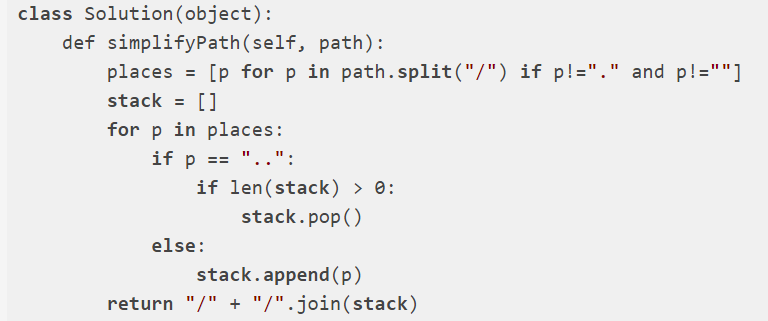
Return res

1. Longest Common Prefix

不难，注意截取了新的lpf要break一下，否则会超出范围



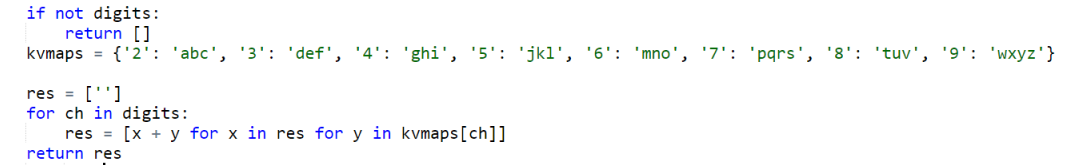
1. Simplify Path

用到了stack，注意最后return的地方非常tricky，一开始没想到这么处理；路径问题，/c/..表示返回c的母目录，/.忽略，/...保留即可

1. Validate IP Address

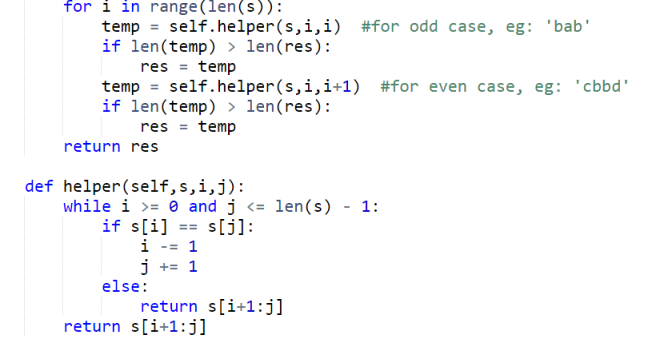
主要是注意ipv4和ipv6各自的一些特殊情况要单独考虑

1. Letter Combinations of a Phone Number



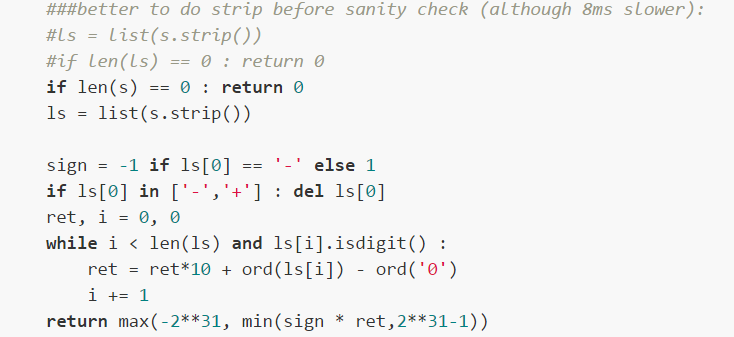
主要是这个dfs一开始把我搞晕了，没想到这么写；一开始想着用递归的方式引入一个新的helper函数，后来发现太麻烦没这种方法简单

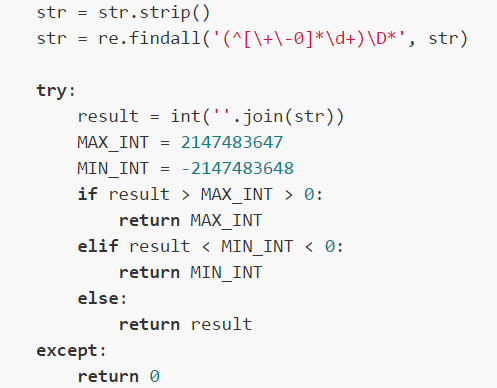
1. Longest Palindromic Substring



将字符串的每一个字符作为middle point来考虑，分别向两边拓展，要注意substring为odd和even这两种不同的情况

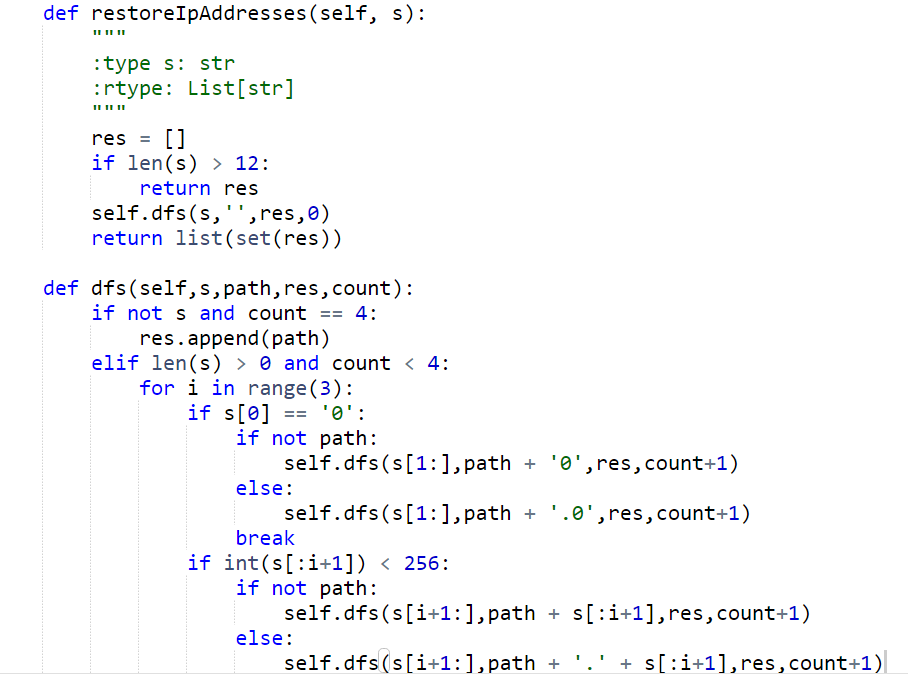
1. String to Integer (atoi)

自己写的straightforward的代码，感觉贼鸡儿蠢。。。Solutions里扒了2个不错的，sol2用了正则表达式

strip函数好久没用也忘了，( ╯□╰ )

1. Restore IP Addresses

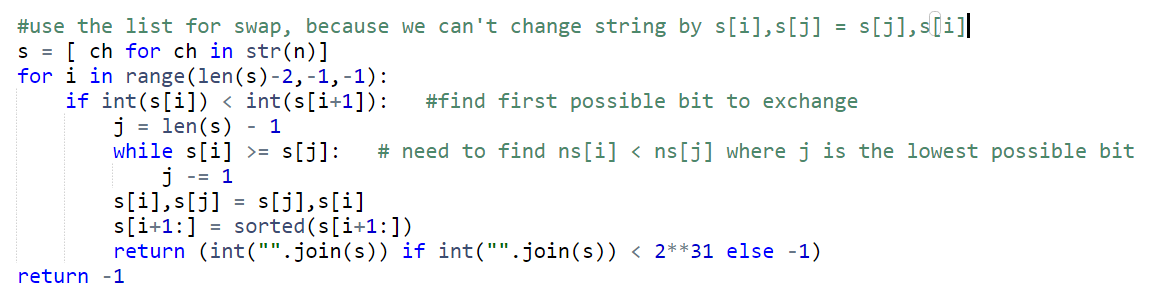
一道标准的dfs题，但是很久没做dfs记不清具体的套路了，只记得框架，别忘了要加path！！！本题中还需要注意leading zero的情况



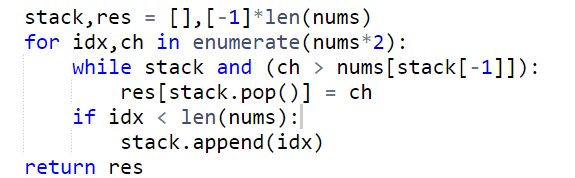
1. Optimal Division

看着很复杂，其实特别简单，只需要让第一个数作为被除数，后面的数作为分母先除就好了

1. Next Greater Element III



1. Next Greater Element II

stack的妙用，Push the index on the stack. If the current number b is bigger than the last number a in the stack(found by index), then we find the next great element for a. Process it twice as it is a circular array to make sure that we can reread the next greater element after every element.

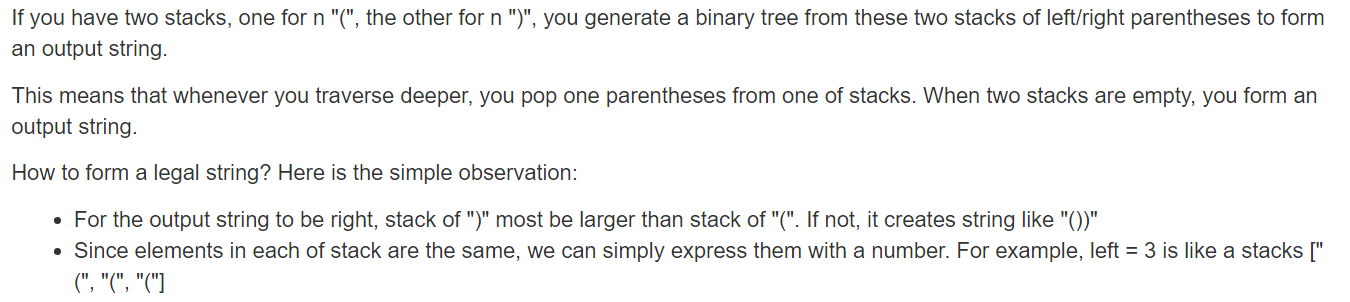
1. Next Greater Element I

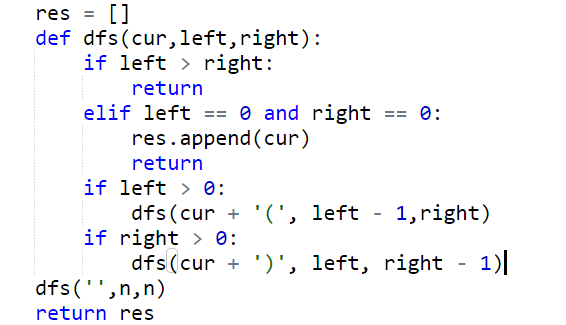
****Algorithm****

* <https://discuss.leetcode.com/topic/77916/java-10-lines-linear-time-complexity-o-n-with-explanation>
* Suppose we have a decreasing sequence followed by a greater number. For example [5, 4, 3, 2, 1, 6] then the greater number 6 is the next greater element for all previous numbers in the sequence.
* We use a stack to keep a decreasing sub-sequence, whenever we see a number x greater than stack.peek() we pop all elements less than x and for all the popped ones, their next greater element is x.
* For example [9, 8, 7, 3, 2, 1, 6]. The stack will first contain [9, 8, 7, 3, 2, 1] and then we see 6 which is greater than 1 so we pop 1 2 3 whose next greater element should be 6.

22. Generate Parentheses

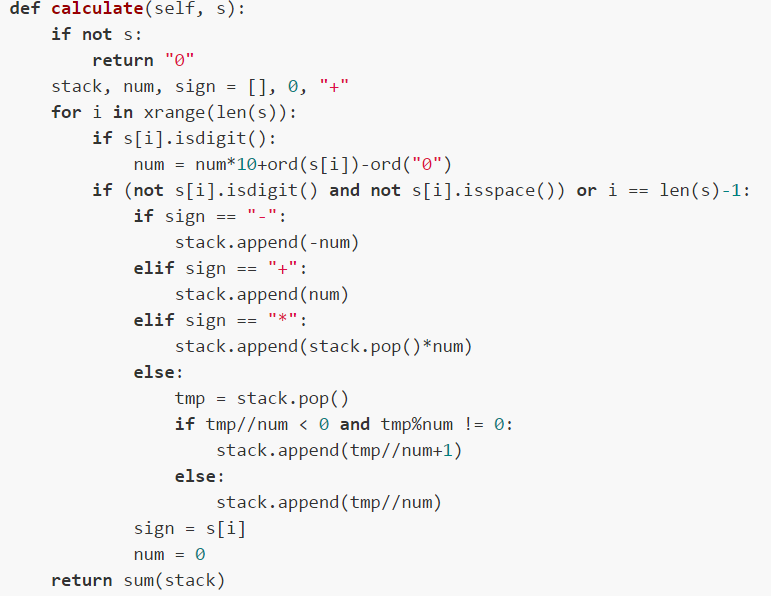
和一般的dfs不同在于这题用了check left、right方法



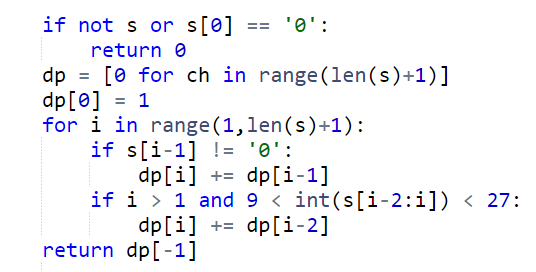


1. Basic Calculator II

要用到stack的方法，难倒是不难，就是太容易忽略特殊情况了，最气的的是sublime上的结果都是对的，放到leetcode上就不对又要改。。。这里放上solution里的一个简短答案，之后二刷的时候试一试

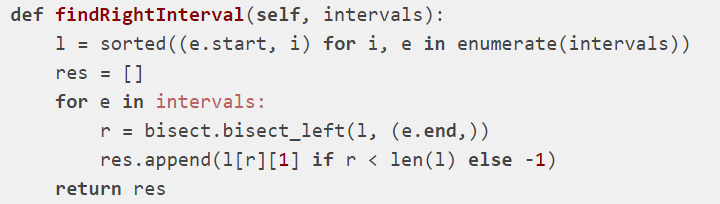


91. Decode Ways

DP solution，注意dp[]的长度是len(s)+1

436. Find Right Interval

二分查找题，这里为了方便用了bisect这个标准库内置模块，另外还需要使用字典排序的方法，get了一种新的字典排序的方法，as follow



1. Permutation in String

Two pointers（sliding window），不算难，就是要注意sliding window左右两端的范围问题；improvement:自己做得时候用了Counter函数，降低了速度，其实不需要引用这个函数

target = [0] \* 26

**for** x **in** A:

target[x] += 1 这部分code就可以作为counter函数使用了

88. Merge Sorted Array 只能用in-place method

while m > 0 and n > 0:

if nums1[m-1] >= nums2[n-1]:

nums1[m+n-1] = nums1[m-1]

m -= 1

else:

nums1[m+n-1] = nums2[n-1]

n -= 1

while n > 0:

nums1[m+n-1] = nums2[n-1]

n -= 1

1. Minimum Size Subarray Sum

思路是使用sliding window的做法

sumx,start,min\_l=0,0,len(nums)+1

**for** i **in** range(len(nums)):

sumx+=nums[i]

**while** sumx>=s:

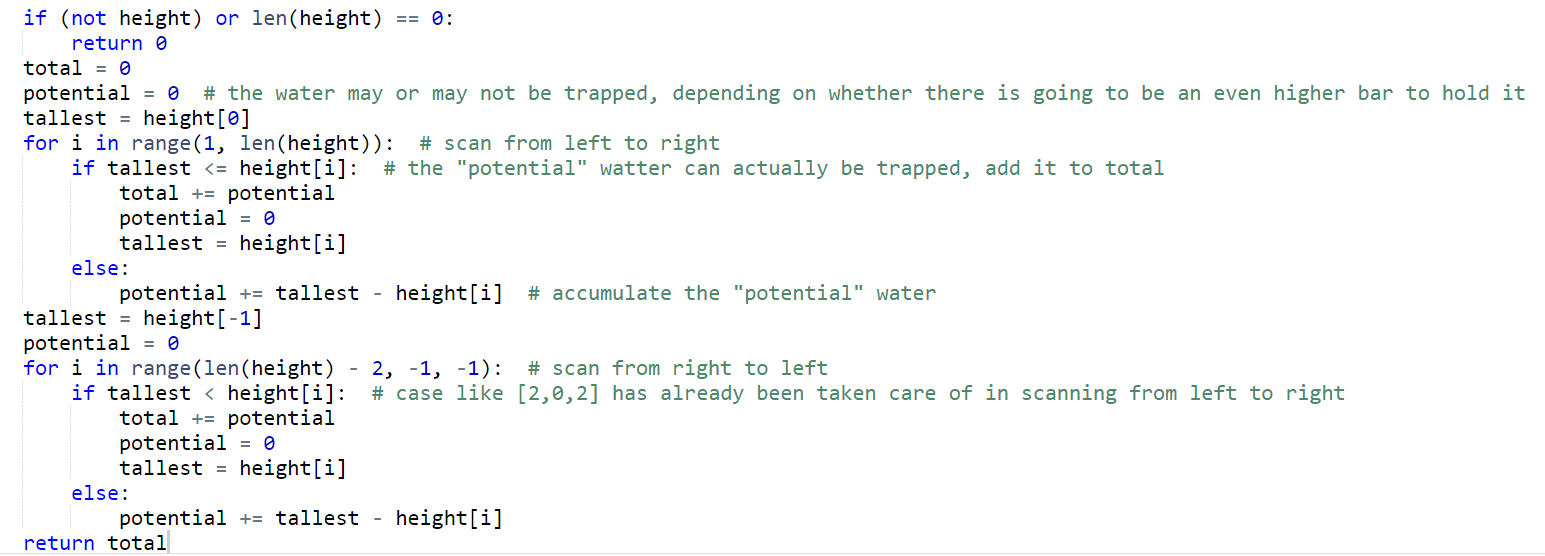
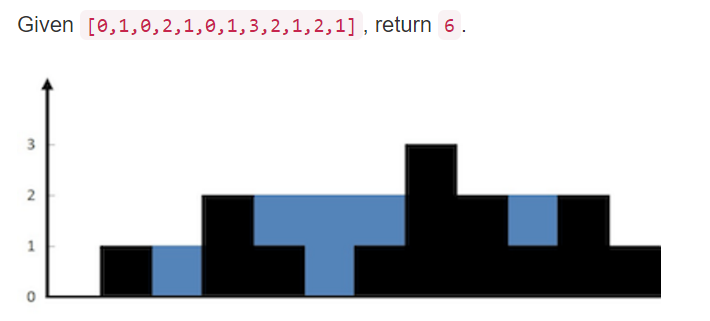
min\_l=min(min\_l,i-start+1)

sumx-=nums[start]

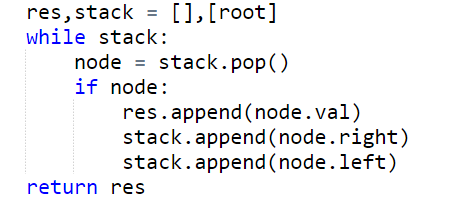
start+=1

**return** 0 **if** min\_l==len(nums)+1 **else** min\_l

1. Trapping Rain Water

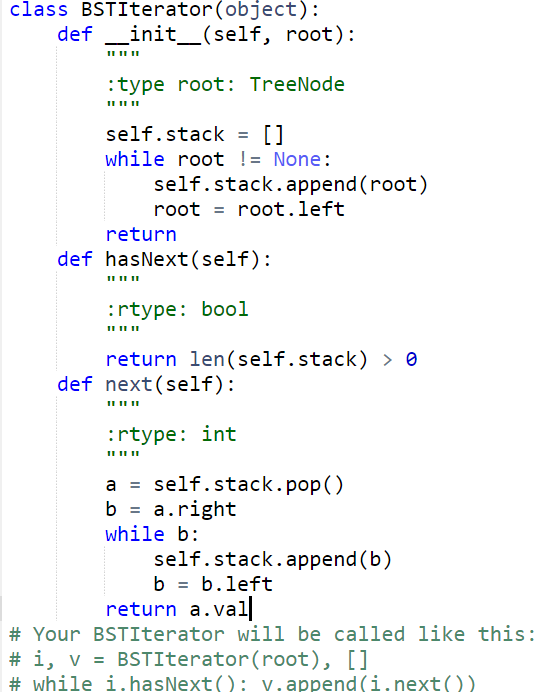


1. Binary Tree Preorder Traversal

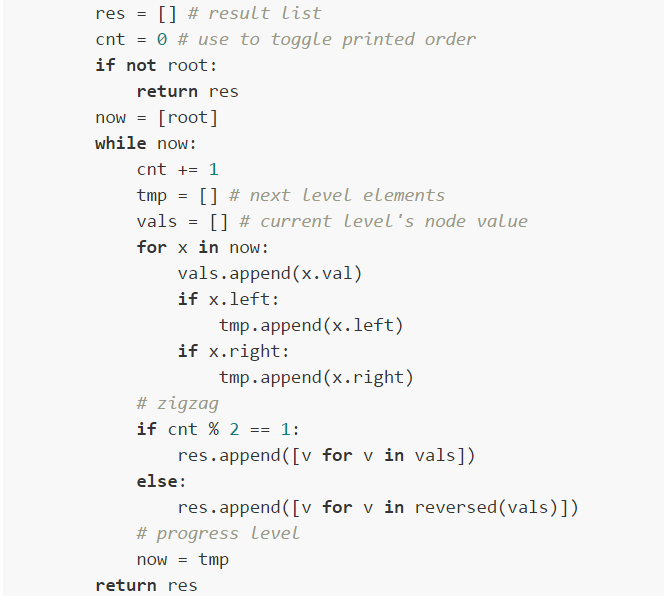
明显的stack题

1. Binary Search Tree Iterator

就是要以inorder的顺序输出binary Search Tree



1. Binary Tree Zigzag Level Order Traversal

The idea is simple. I use a counter to determine whether to print the current level in normal order or reversed order. And be sure to put ****node.val**** instead of ****node**** to the final result.