# leetcode解题思路

1.

## 26. Remove Duplicates from Sorted Array

Question | My Submissions

Total Accepted: 118738 Total Submissions: 358980 Difficulty: Easy

Given a sorted array, remove the duplicates in place such that each element appear only *once* and return the new length.

Do not allocate extra space for another array, you must do this in place with constant memory.

For example,

Given input array *nums* = `[1,1,2]`,

Your function should return length = `2`, with the first two elements of *nums* being `1` and `2` respectively. It doesn't matter what you leave beyond the new length.

数组原地去重,要求给定长度之前是去重之后的

[1,2,2,3,3,4]->[1,2,3,4,2,3] length=4

思路：two pointers：p1指向当前空位，p2指向与上一个元素不同的元素(实际上p2可以直接与p2前一个元素比较，因为数组已经排好序，前一个元素一但第一次出现，肯定会作为上一个不同的数)

将与上一个不同的数组元素往前放

```python
class Solution(object):
    def removeDuplicates(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        注意：start之前完全满足条件，start指向当前需要填数的位置，来一个数只需要与num[start-1]比较即可
        """
        if len(nums)<=1:
            return len(nums)
        start = 1#start始终指向下一个期望的位置
        for i in range(len(nums)):
            if nums[i]!=nums[start-1]:
                nums[start] = nums[i]
                start+=1
        return start
```

## 80. Remove Duplicates from Sorted Array II

Question | My Submissions

Total Accepted: 70372 Total Submissions: 217130 Difficulty: Medium

Follow up for "Remove Duplicates":

What if duplicates are allowed at most *twice*?

For example,

Given sorted array *nums* = `[1,1,1,2,2,3]`,

Your function should return length = `5`, with the first five elements of *nums* being `1`, `1`, `2`, `2` and `3`. It doesn't matter what you leave beyond the new length.

思路：

```python
class Solution(object):
    def removeDuplicates(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        重点题型：面试经常考察：
        start之前表示当前已经完全确定的数（即start之前完全满足递增且最多出现2次的要求），start指向当前期望的下一个数，
        如果下一个数与前2个数不相同，则可以直接放到start,如果与前2个数相同，那么直接跳到下一个数
        """
        if len(nums)<=2:
            return len(nums)
        start = 2
        for i in range(2,len(nums)):
            if nums[i]!=nums[start-2]:
                nums[start] = nums[i]
                start+=1
        return start
```

2.

## 118. Pascal's Triangle

Total Accepted: **75451** Total Submissions: **231291** Difficulty: **Easy**

> Given *numRows*, generate the first *numRows* of Pascal's triangle.
>
> For example, given *numRows* = 5,
>
> Return
>
> ```
> [
>      [1],
>     [1,1],
>    [1,2,1],
>   [1,3,3,1],
>  [1,4,6,4,1]
> ]
> ```

思路：在上一层的两边加0，然后第i个位置与第i+1个位置相加即可

3.

## 31. Next Permutation

Total Accepted: **59605** Total Submissions: **228642** Difficulty: **Medium**

> Implement next permutation, which rearranges numbers into the lexicographically next greater permutation of numbers.
>
> If such arrangement is not possible, it must rearrange it as the lowest possible order (ie, sorted in ascending order).
>
> The replacement must be in-place, do not allocate extra memory.
>
> Here are some examples. Inputs are in the left-hand column and its corresponding outputs are in the right-hand column.
>
> `1,2,3` → `1,3,2`
>
> `3,2,1` → `1,2,3`
>
> `1,1,5` → `1,5,1`

思路：

```python
class Solution(object):
    def nextPermutation(self, nums):
        """
        :type nums: List[int]
        :rtype: void Do not return anything, modify nums in-place instead.
        思路：双指针：从后往前找第一个 比后面数小的，定位边界 i，从前往后找 第一个 比边界数nums[i]大的 交换，然后逆置i之后的数
        """
        if len(nums)<=2:
            nums.reverse()
            return
        i,j = len(nums)-2,len(nums)-1
        #第一步：从后往前找第一个nums[i]<nums[j]的
        while i>=0 and nums[i]>=nums[j]:
            i-=1
            j-=1
        #没找到，说明完全逆置，直接反转过来就恢复顺序的序列
        if i==-1:
            nums.reverse()
            return
        #找到i,i之后一定是逆序的，从后往前找第一个比i大的，然后逆转i之后的所有序列
        else:

            k = len(nums)-1
            while nums[k]<=nums[i]:
                k-=1
            print("i:{} j:{} k:{}".format(i,j,k))
            tmp = nums[i]
            nums[i]= nums[k]
            nums[k] = tmp
            nums[i+1:] = reversed(nums[i+1:])#列表赋值可以用iterable
```

4.

## 47. Permutations II

Total Accepted: 63974 Total Submissions: 232522 Difficulty: Medium

> Given a collection of numbers that might contain duplicates, return all possible unique permutations.
>
> For example,
>
> `[1,1,2]` have the following unique permutations:
>
> `[1,1,2]`, `[1,2,1]`, and `[2,1,1]`.
>
> Subscribe to see which companies asked this question
>
> Show Tags
>
> Show Similar Problems

与上面那道题的区别就是先对数组进行排序，然后用相同的方法就可以产生不重复的序列

5.

## 303. Range Sum Query - Immutable

Total Accepted: 20708 Total Submissions: 84549 Difficulty: Easy

> Given an integer array *nums*, find the sum of the elements between indices *i* and *j* (*i* ≤ *j*), inclusive.
>
> **Example:**
>
> ```
> Given nums = [-2, 0, 3, -5, 2, -1]
>
> sumRange(0, 2) -> 1
> sumRange(2, 5) -> -1
> sumRange(0, 5) -> -3
> ```
>
> **Note:**
>
> 1. You may assume that the array does not change.
> 2. There are many calls to *sumRange* function.

思路：采用动态规划:sum[i,j] = sum[j]-sum[i-1]

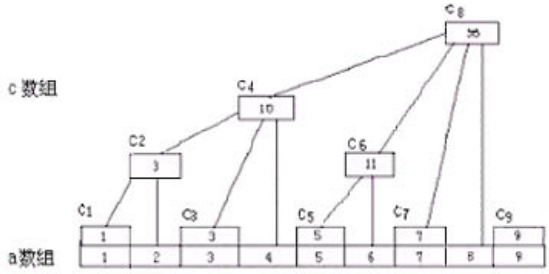
6.

## 307. Range Sum Query - Mutable

Total Accepted: 6997 Total Submissions: 41440 Difficulty: Medium

> Given an integer array *nums*, find the sum of the elements between indices *i* and *j* (*i* ≤ *j*), inclusive.
>
> The *update(i, val)* function modifies *nums* by updating the element at index *i* to *val*.
>
> **Example:**
>
> ```
> Given nums = [1, 3, 5]
>
> sumRange(0, 2) -> 9
> update(1, 2)
> sumRange(0, 2) -> 8
> ```

**树状数组**主要用于查询任意两位之间的所有元素之和，但是每次只能修改一个元素的值；可以在log(n)的复杂度下进行范围修改
树状数组：
http://blog.csdn.net/int64ago/article/details/7429868
主要记住：
1.这个图：我们只维护c数组，标号一定要从1开始




2.c数组与A数组的对应关系：C[n] 负责的A的元素个数为lowbit(n):n&(-n)
C[n]一定监管A[n]
如c1:01 末尾0个数为0，因此负责2^0=1 个A[1]
C2: 10 。。。。为1，因此负责2^1=2个A[1]和A[2]
c3:11 .....0.....2^0=1 个A[3]
。。。。画出这个图后，编程实现插入，求和 和 update（调插入）
插入时 ： 由子节点向父亲节点依次更新
求和时：由父亲节点向子节点依次往下累加：

重要关系式：
1）lowbit是建立树状数组父子节点的桥梁，c[i+lowbit(i)]是找c[i]的父节点,c[i-lowbit(i)]是找c[i]的子节点
2）sum[k] = c[k] + c[k-lowbit[k]] +c[k-lowbit[k-lowbitk]]+....

7.

## 228. Summary Ranges

Total Accepted: 39177 Total Submissions: 167852 Difficulty: Easy

> Given a sorted integer array without duplicates, return the summary of its ranges.
>
> For example, given `[0,1,2,4,5,7]`, return `["0->2","4->5","7"].`

思路：双指针法：p1指向start,p2指向end,str(nums[start]) +"->"+str(nums[end]) if nums[start]!=nums[end] else str(nums[start])

8.

## Wiggle Sort

Total Accepted: **1291** Total Submissions: **3008** Difficulty: **Medium**

Given an unsorted array `nums`, reorder it **in-place** such that `nums[0] <= nums[1] >= nums[2] <= nums[3]...`.

For example, given `nums = [3, 5, 2, 1, 6, 4]`, one possible answer is `[1, 6, 2, 5, 3, 4]`.

思路：要求不严格(不是严格的>或<)贪心，不满足上面条件的就交换
nums[0]<=nums[1]>=nums[2]<=nums[3]>=nums[4]<=nums[5]>=nums[6]
0 1 2 3 4 5 6
对于i为1,3,5...奇数，如果nums[i]<nums[i-1],则交换
对于i为0,2,4,....偶数，如果nums[i]>nums[i-1],则交换
i从1开始，交换后可以保证nums[1]>=nums[0],接下来,i=2,如果nums[2]>nums[1],则需要交换，
交换后nums[2]>nums[1],而nums[1]>=nums[0]，因此nums[2]>nums[0]
这就意味着交换后仍然能保持前面的性质.
因此偶数的交换不影响奇数的交换
因为任意给一个乱序数组，相邻的两个数一定是>=或<=关系，这样就意味着只要交换，就一定能够满足要求
9.

## 324. Wiggle Sort II

Total Accepted: 5807 Total Submissions: 27028 Difficulty: Medium

> Given an unsorted array `nums`, reorder it such that `nums[0] < nums[1] > nums[2] < nums[3]...`.
>
> **Example:**
>
> (1) Given `nums = [1, 5, 1, 1, 6, 4]`, one possible answer is `[1, 4, 1, 5, 1, 6]`.
>
> (2) Given `nums = [1, 3, 2, 2, 3, 1]`, one possible answer is `[2, 3, 1, 3, 1, 2]`.
>
> **Note:**
>
> You may assume all input has valid answer.

思路：要求严格的>或<,这时有重复的交换就不一定满足要求（需要对真个数组进行重排）
简单做法：对整个数组进行排序，然后由大到小插入位置[1,3,5,....0,2,4,6...]等一定符合要求
但是排序时间复杂度为O(nlogn),题目要时间复杂度为O(n)，空间复杂度为O(1)
这就意味着我们要在原数组上进行操作，实际上我们不需要排序如A[1]>A[3]>A[5]。。。>A[0]>A[2]>A[4]>A[6]。。
我们只需要保证A[1],A[3],A[5]。。>中位数 > A[0]和A[2],A[4]等即可
在原数组上操作,i=1:n-1 我们需要将
0 1 2 3 4 5 ... n-1 => 1 3 5 ...0 2 4 6等等
进行坐标映射,具体映射方法:可以多写几个推公式获得
index = [ 2*i+1 if i<size/2 else (2*i%size if size%2==0 else (2*i+1)%size) for i in range(size)]
排序时，可以用三路排序法(3 Pointers):i,j,k <=i都>mid ; >=k 都<mid ；直到处理到j与k相遇

注意：找第k大的数，时间复杂度为O(n)
10.

## 67. Add Binary

Total Accepted: 75856 Total Submissions: 283565 Difficulty: Easy

> Given two binary strings, return their sum (also a binary string).
>
> For example,
>
> a = `"11"`
>
> b = `"1"`
>
> Return `"100"`.

思路： 2 pointers:从后向前处理
while p1>=0 or p2>=0 or carry_on:#注意：这里的循环条件，很多做加法，包括链表都是这个循环条件

11.

## 283. Move Zeroes

Total Accepted: 60142 Total Submissions: 138154 Difficulty: Easy

> Given an array `nums`, write a function to move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.
>
> For example, given `nums = [0, 1, 0, 3, 12]`, after calling your function, `nums` should be `[1, 3, 12, 0, 0]`.
>
> **Note**:
>
> 1. You must do this **in-place** without making a copy of the array.
> 2. Minimize the total number of operations.

思路：
2 pointers : p1 指0,p2往后面找非0的元素，然后交换

12.

# 78. Subsets

Total Accepted: 86592 Total Submissions: 282127 Difficulty: Medium

Given a set of distinct integers, *nums*, return all possible subsets.

**Note:**

- Elements in a subset must be in non-descending order.
- The solution set must not contain duplicate subsets.

For example,

If ***nums*** = `[1,2,3]`, a solution is:

```
[
  [3],
  [1],
  [2],
  [1,2,3],
  [1,3],
  [2,3],
  [1,2],
  []
]
```

思路1：

1 2 3

先初始化为[]

[] res=[]

然后从后往前处理:

3 res =[],([3])

2 res = [],[3],([2],[2,3])

1 res = [],[3],[2],[2,3],([1],[1,3],[1,2],[1,2,3])

后面的括号为新产生的，在前面的基础之上全部在首部添加新来的数

这种方法比递归产生更高效，而且更容易实现

思路2：

采用递归的思想:

1,2,3

1 -> [] => [1]

[2] => [1,2]

[3] => [1,3]

[2,3] => [1,2,3]

2 -> [] => [2]

[3] => [2,3]

3 -> [] => [3]

+[]

13.

# 319. Bulb Switcher

Question My Submissions

Total Accepted: 12122 Total Submissions: 30650 Difficulty: Medium

There are *n* bulbs that are initially off. You first turn on all the bulbs. Then, you turn off every second bulb. On the third round, you toggle every third bulb (turning on if it's off or turning off if it's on). For the *i*th round, you toggle every *i* bulb. For the *n*th round, you only toggle the last bulb. Find how many bulbs are on after *n* rounds.

**Example:**

```
Given n = 3.

At first, the three bulbs are [off, off, off].
After first round, the three bulbs are [on, on, on].
After second round, the three bulbs are [on, off, on].
After third round, the three bulbs are [on, off, off].

So you should return 1, because there is only one bulb is on.
```

思路：第i个灯是否亮取决于从1到i有多少个因子，

比如 2, 1,2 偶数个因子 ->切换偶数次，变成最初的状态灭

如果有奇数个因子，那么最后就亮=》 2 = 1*2 都是成对出现的，只有当4= 2*2 时，两个因数相同，才会有奇数个因子，最终亮，

因此，最终亮的灯的编号为： 1^2 , 2^2 , 3^2,4^2,5^2。。。看《=n有多少个平方数就行了

14.

## 299. Bulls and Cows

Question | My Submissions

Total Accepted: 20590 Total Submissions: 71674 Difficulty: Easy

You are playing the following Bulls and Cows game with your friend: You write down a number and ask your friend to guess what the number is. Each time your friend makes a guess, you provide a hint that indicates how many digits in said guess match your secret number exactly in both digit and position (called "bulls") and how many digits match the secret number but locate in the wrong position (called "cows"). Your friend will use successive guesses and hints to eventually derive the secret number.

For example:

```
Secret number:  "1807"
Friend's guess: "7810"
```

Hint: `1` bull and `3` cows. (The bull is `8`, the cows are `0`, `1` and `7`.)

Write a function to return a hint according to the secret number and friend's guess, use `A` to indicate the bulls and `B` to indicate the cows. In the above example, your function should return `"1A3B"`.

Please note that both secret number and friend's guess may contain duplicate digits, for example:

```
Secret number:  "1123"
Friend's guess: "0111"
```

In this case, the 1st `1` in friend's guess is a bull, the 2nd or 3rd `1` is a cow, and your function should return `"1A1B"`.

You may assume that the secret number and your friend's guess only contain digits, and their lengths are always equal.

```
思路：res_A 一个一个对应，res_B 只跟个数有关，并且 res_B 包含 res_A
"""
res_A = [x==y for x,y in zip(secret,guess)].count(True)
res_B = sum([min(secret.count(x),guess.count(x)) for x in set(guess)])-res_A
return str(res_A)+"A"+str(res_B)+"B"
```

15.

## 290. Word Pattern

Total Accepted: 27704 Total Submissions: 97927 Difficulty: Easy

Given a `pattern` and a string `str`, find if `str` follows the same pattern.

Here **follow** means a full match, such that there is a bijection between a letter in `pattern` and a **non-empty** word in `str`.

**Examples:**

1. pattern = `"abba"`, str = `"dog cat cat dog"` should return true.
2. pattern = `"abba"`, str = `"dog cat cat fish"` should return false.
3. pattern = `"aaaa"`, str = `"dog cat cat dog"` should return false.
4. pattern = `"abba"`, str = `"dog dog dog dog"` should return false.

思路：pattern 和 str 分离保存，
用p_s_map : pattern ->str的映射， s_p_map:str->pattern的映射
这两个映射一定要同时建立,不能单方面的修改一个

```cpp
class Solution {
public:
    bool wordPattern(string pattern, string str) {
        map<char, string> p_s;
        map<string, char> s_p;
        vector<string> strVec;
        auto start = 0;
        while (start <str.length())
        {
            auto end = str.find_first_of(' ', start);
            if (end == string::npos)
                end = str.length();//易错点1：字符串的查找返回的是pos，不是迭代器，如果是string
                    ::npos的话就相当于一个很大的数，下面的步骤会报错
            string tmp(str.begin() + start, str.begin() + end);
            strVec.push_back(tmp);
            start = end + 1;
        }
        if (strVec.size() != pattern.length()) return false;
        for (int i = 0; i < strVec.size(); i++)
        {
            if (s_p.find(strVec[i]) == s_p.end()&&p_s.find(pattern[i])==p_s.end())//注意2：这里是易错点：建立映射时必须同时找不到
            {
                s_p[strVec[i]] = pattern[i];
                p_s[pattern[i]] = strVec[i];
            }
            else if (s_p[strVec[i]] == pattern[i] && strVec[i] == p_s[pattern[i]])//或者同时都在映射里
                continue;
            else
                return false;
        }
        return true;
    }
};
```

16.

## 205. Isomorphic Strings

Question | My Submissions

Total Accepted: **48186** Total Submissions: **167325** Difficulty: **Easy**

Given two strings ***s*** and ***t***, determine if they are isomorphic.

Two strings are isomorphic if the characters in ***s*** can be replaced to get ***t***.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character but a character may map to itself.

For example,

Given `"egg"`, `"add"`, return true.

Given `"foo"`, `"bar"`, return false.

Given `"paper"`, `"title"`, return true.

思路：与上一题的代码完全一样

17.

## 62. Unique Paths

Question | My Submissions

Total Accepted: **79662** Total Submissions: **223884** Difficulty: **Medium**

A robot is located at the top-left corner of a *m* x *n* grid (marked 'Start' in the diagram below).

The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below).

How many possible unique paths are there?


Above is a 3 x 7 grid. How many possible unique paths are there?

**Note:** *m* and *n* will be at most 100.

思路：动规:
opt[i][j] = opt[i][j-1]+opt[i-1][j]

18.

## 279. Perfect Squares

Total Accepted: **26950** Total Submissions: **85398** Difficulty: **Medium**

> Given a positive integer *n*, find the least number of perfect square numbers (for example, `1, 4, 9, 16, ...`) which sum to *n*.
>
> For example, given *n* = `12`, return `3` because `12 = 4 + 4 + 4`; given *n* = `13`, return `2` because `13 = 4 + 9`.
>
> **Credits:**
>
> Special thanks to @jianchao.li.fighter for adding this problem and creating all test cases.
>
> Subscribe to see which companies asked this question

思路：动态规划：

opt[i] = min([1+opt[i-j*j] for j in range(1,int(sqrt(i)))])

19.

## 46. Permutations

Total Accepted: **90089** Total Submissions: **258223** Difficulty: **Medium**

> Given a collection of **distinct** numbers, return all possible permutations.
>
> For example,
>
> `[1,2,3]` have the following permutations:
>
> `[1,2,3]`, `[1,3,2]`, `[2,1,3]`, `[2,3,1]`, `[3,1,2]`, and `[3,2,1]`.
>
> Subscribe to see which companies asked this question

思路：求逆序数：

1）生成下一个逆序数 2）如果完全逆序，就终止 3）注意：在原数组上修改需要copy.deepcopy

20.

## 40. Combination Sum II

Question

Total Accepted: **62290** Total Submissions: **230055** Difficulty: **Medium**

> Given a collection of candidate numbers (***C***) and a target number (***T***), find all unique combinations in ***C*** where the candidate numbers sums to ***T***.
>
> Each number in ***C*** may only be used **once** in the combination.
>
> **Note:**
>
> - All numbers (including target) will be positive integers.
> - Elements in a combination ($a_1$, $a_2$, … , $a_k$) must be in non-descending order. (ie, $a_1 \le a_2 \le \dots \le a_k$).
> - The solution set must not contain duplicate combinations.
>
> For example, given candidate set `10,1,2,7,6,1,5` and target `8`,
>
> A solution set is:
>
> `[1, 7]`
>
> `[1, 2, 5]`
>
> `[2, 6]`
>
> `[1, 1, 6]`

思路：深度优先，注意先排序，然后添加时注意去重

21

## 79. Word Search

Question | My Submissions

Total Accepted: 68273 Total Submissions: 303950 Difficulty: Medium

Given a 2D board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once.

For example,

Given **board** =

```
[
  ['A','B','C','E'],
  ['S','F','C','S'],
  ['A','D','E','E']
]
```

**word** = `"ABCCED"`, -> returns `true`,

**word** = `"SEE"`, -> returns `true`,

**word** = `"ABCB"`, -> returns `false`.

思路：

dfs 递归遍历所有初始可能，只要有一种可能能达到，就返回True

```
#注意：这种写法是有正确结果才返回
if board[i][j]==word[0] and self.search(board,word,1,i,j,search_area,used_cells+[(i,j)])
    return True
'''
if board[i][j]==word[0] :这种写法是：只要找到第一个相等的，不管怎么样都返回
    return self.search(board,word,1,i,j,search_area,used_cells+[(i,j)])
'''
```

22.

## 237. Delete Node in a Linked List

Question | My Submissions

Total Accepted: 66526 Total Submissions: 152267 Difficulty: Easy

Write a function to delete a node (except the tail) in a singly linked list, given only access to that node.

Supposed the linked list is `1 -> 2 -> 3 -> 4` and you are given the third node with value `3`, the linked list should become `1 -> 2 -> 4` after calling your function.

思路：

```
class Solution(object):
    def deleteNode(self, node):
        """
        :type node: ListNode
        :rtype: void Do not return anything, modify node in-place instead.
        注意：此题特别巧妙
        将下一个节点的值复制到当前待删除的节点上，然后将下个节点删除
        """
        if not node:
            return
        node.val = node.next.val
        node.next = node.next.next
```

23.

## 203. Remove Linked List Elements

Total Accepted: 54114 Total Submissions: 193172 Difficulty: Easy

Remove all elements from a linked list of integers that have value ***val***.

**Example**

***Given:*** 1 --> 2 --> 6 --> 3 --> 4 --> 5 --> 6, ***val*** = 6

***Return:*** 1 --> 2 --> 3 --> 4 --> 5

思路：

加头节点，另外要注意是要删除一个还是删除所有的值为val的节点

24.

# 160. Intersection of Two Linked Lists

Total Accepted: **64941** Total Submissions: **215809** Difficulty: **Easy**

> Write a program to find the node at which the intersection of two singly linked lists begins.
>
> For example, the following two linked lists:
>
> ```
> A:          a1 → a2
>                    ↘
>                      c1 → c2 → c3
>                    ↗
> B:     b1 → b2 → b3
> ```
>
> begin to intersect at node c1.
>
> **Notes:**
>
> - If the two linked lists have no intersection at all, return `null`.
> - The linked lists must retain their original structure after the function returns.
> - You may assume there are no cycles anywhere in the entire linked structure.
> - Your code should preferably run in O(n) time and use only O(1) memory.

思路：2个链表一定是在最后公用一段：
先获取2个链表的长度，然后将长的链表头指针移动多出短的链表部分，然后开始一一对应的找

25.

# 318. Maximum Product of Word Lengths

Question My Submissions

Total Accepted: **13768** Total Submissions: **35420** Difficulty: **Medium**

> Given a string array `words`, find the maximum value of `length(word[i]) * length(word[j])` where the two words do not share common letters. You may assume that each word will contain only lower case letters. If no such two words exist, return 0.
>
> **Example 1:**
>
> Given `["abcw", "baz", "foo", "bar", "xtfn", "abcdef"]`
>
> Return `16`
>
> The two words can be `"abcw", "xtfn"`.
>
> **Example 2:**
>
> Given `["a", "ab", "abc", "d", "cd", "bcd", "abcd"]`
>
> Return `4`
>
> The two words can be `"ab", "cd"`.
>
> **Example 3:**
>
> Given `["a", "aa", "aaa", "aaaa"]`
>
> Return `0`
>
> No such pair of words.

思路：
用位操作来表示一个单词中含有的字符，这种方法常用于单词求交集：但不能用于单词比较！！！因为在一个单词中同一个字符可能出现多次！！
'a'是00000000001 'b'是00000000010 'c'是00000000100 依次类推

26.

# 169. Majority Element

Total Accepted: **97758** Total Submissions: **242885** Difficulty: **Easy**

> Given an array of size *n*, find the majority element. The majority element is the element that appears more than `⌊ n/2 ⌋` times.
>
> You may assume that the array is non-empty and the majority element always exist in the array.

注意：摩尔计数法的顺序：先判相等，然后判是否需要换元素，最后减支持度；每来一个元素只执行以上3种操作中的一种

思路：

```
class Solution(object):
    def majorityElement(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        思路：voting :两两不同的数抵消，最后剩下的数就一定是主元素
        这种算法对于存在主元素的数组是有效的，如：

        A A A C C B B C C C B C C

        它肯定能返回主元素C。但是，如果不存在主元素，那么得到的结果就跟遍历顺序有关了。如：

        A A A C C C B

        如果是从左到右，那么结果是B，如果是从右到左，那么结果是A。
        """
        count = 1
        major_elem =nums[0]
        for i in range(1,len(nums)):
            if count==0:#抵消完了，新设置major_elem 和他的支持度1
                major_elem = nums[i]
                count = 1
            elif nums[i]!=major_elem:#不相等，将major_elem的支持度-1
                count-=1
            else:#相等，将major_elem的支持度+1
                count+=1
        return major_elem
```

27.

229. Majority Element II

Question | My Submissions

Total Accepted: 23574 Total Submissions: 93359 Difficulty: **Medium**

> Given an integer array of size *n*, find all elements that appear more than `⌊ n/3 ⌋` times. The algorithm should run in linear time and in O(1) space.

思路：

同样是摩尔计数法，出现次数大于1/3的最多有两个，因此就找出现次数最多的两个就行了(采用摩尔计数法——真正的解一定包含在摩尔计数法里面，为排除伪答案，需要验证)

注意：摩尔计数法的顺序：先判相等，然后判是否需要换元素，最后减支持度；每来一个元素只执行以上3种操作中的一种

28.

331. Verify Preorder Serialization of a Binary Tree

Question | My Submissions

Total Accepted: 6081 Total Submissions: 19323 Difficulty: **Medium**

> One way to serialize a binary tree is to use pre-order traversal. When we encounter a non-null node, we record the node's value. If it is a null node, we record using a sentinel value such as `#`.
>
> ```
>      _9_
>     /   \
>    3     2
>   / \   / \
>  4   1  #  6
> / \ / \   / \
> # # # #   # #
> ```
>
> For example, the above binary tree can be serialized to the string `"9,3,4,#,#,1,#,#,2,#,6,#,#"`, where `#` represents a null node.
>
> Given a string of comma separated values, verify whether it is a correct preorder traversal serialization of a binary tree. Find an algorithm without reconstructing the tree.
>
> Each comma separated value in the string must be either an integer or a character `'#'` representing `null` pointer.
>
> You may assume that the input format is always valid, for example it could never contain two consecutive commas such as `"1,,3"`.
>
> **Example 1:**
>
> `"9,3,4,#,#,1,#,#,2,#,6,#,#"`
>
> Return `true`
>
> **Example 2:**
>
> `"1,#"`
>
> Return `false`
>
> **Example 3:**
>
> `"9,#,#,1"`



思路：

```python
class Solution(object):
    def isValidSerialization(self, preorder):
        """
        :type preorder: str
        :rtype: bool
        思路：根据二叉树上还能挂几个节点，一个根节点(如果不为"#")后面还有2个空位置，
        增加一个子节点，如果是数，可以增加一个空位置；如果是"#"减少一个空位置
        如果序列没遍历完，就始终保持empty_space>=0 并且empty_space最后一定是0
        """
        nodes = preorder.split(',')
        empty_space = 2 if nodes[0]!="#" else 0
        for x in nodes[1:]:
            if empty_space<=0:
                return False
            elif x=="#":
                empty_space-=1
            else:
                empty_space+=1
        return True if empty_space==0 else False
```

29.

## 232. Implement Queue using Stacks

Total Accepted: 35854 Total Submissions: 105634 Difficulty: Easy

> Implement the following operations of a queue using stacks.
>
> - push(x) -- Push element x to the back of queue.
> - pop() -- Removes the element from in front of queue.
> - peek() -- Get the front element.
> - empty() -- Return whether the queue is empty.

思路：
用两个栈，一个专门负责进元素，一个专门负责出元素
1)push : 直接进stack_in
2)pop|peak: 如果stack_out不为空，直接出，每当stack_out为空，就从stack_in中倒腾过来所有元素，然后再出一个（peak:不出，只返回）
3)empty: 当stack_in 和 stack_out 都为空时，此时栈为空

30.

## 225. Implement Stack using Queues

Total Accepted: 32946 Total Submissions: 108165 Difficulty: Easy

> Implement the following operations of a stack using queues.
>
> - push(x) -- Push element x onto stack.
> - pop() -- Removes the element on top of the stack.
> - top() -- Get the top element.
> - empty() -- Return whether the stack is empty.

思路：
用两个队列，其中一个缓存已经进来的元素，另一个始终为空(两个队列始终有一个为空)
push:新加入元素只往为空的队列1加，加完后把另一个刚开始不为空的队列2所有元素全部移到队列1后面，这时队列2为空
pop|top:不为空的队列第一个元素
empty:两个队列都为空时

31.

## 278. First Bad Version

Question | My Submissions

Total Accepted: 34941 Total Submissions: 159104 Difficulty: Easy

> You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.
>
> Suppose you have `n` versions `[1, 2, ..., n]` and you want to find out the first bad one, which causes all the following ones to be bad.
>
> You are given an API `bool isBadVersion(version)` which will return whether `version` is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

思路：

```
int binarySearch(vector<int>&nums,int target)
{
    int l = 0, r = nums.size() - 1, mid = l + (r - l) / 2;
    while (l <= r)
    {
        if (nums[mid] < target)
            l = mid + 1;
        else if (nums[mid] == target)
            return mid;
        else
            r = mid - 1;
        mid = l + (r - l) / 2;
    }
    return -1;
}

//查找最左边界的二分查找
//将>=target的合并
int BinarySearchLeft(vector<int>&nums, int target)
{
    int l = 0, r = nums.size() - 1, mid = l + (r - l) / 2;
    while (l <= r)
    {
        if (nums[mid] < target)
            l = mid + 1;
        else
            r = mid - 1;
        mid = l + (r - l) / 2;
    }
    return (nums[l] == target)?l:- 1;
}

//查找最右边界的二分查找
//将<=target的合并
int BinarySearchRight(vector<int>&nums, int target)
{
    int l = 0, r = nums.size() - 1, mid = l + (r - l) / 2;
    while (l <= r)
    {
        if (nums[mid] > target)
            r = mid - 1;
        else
            l = mid + 1;
        mid = l + (r - l) / 2;
    }
    return (nums[r] == target) ? r : -1;
}
```

32.

## 330. Patching Array

Question | My Submissions

Total Accepted: 4863 Total Submissions: 17119 Difficulty: Medium

Given a sorted positive integer array *nums* and an integer *n*, add/patch elements to the array such that any number in range `[1, n]` inclusive can be formed by the sum of some elements in the array. Return the minimum number of patches required.

**Example 1:**

*nums* = `[1, 3]`, *n* = `6`

Return `1`.

Combinations of *nums* are `[1], [3], [1,3]`, which form possible sums of: `1, 3, 4`.

Now if we add/patch `2` to *nums*, the combinations are: `[1], [2], [3], [1,3], [2,3], [1,2,3]`.

Possible sums are `1, 2, 3, 4, 5, 6`, which now covers the range `[1, 6]`.

So we only need `1` patch.

**Example 2:**

*nums* = `[1, 5, 10]`, *n* = `20`

Return `2`.

The two patches can be `[2, 4]`.

**Example 3:**

*nums* = `[1, 2, 2]`, *n* = `5`

Return `0`.

思路：

采用贪心策略:

0 ...+[1] ->(0,1)

(0,1) +[2] ->(0,3)

(0,3)+[5] ->不连续， 因此[0,i]加上一个数下一个区间仍然连续的条件是：新加的x<=i+1,>i+1时，至少从i+2开始，中间断了i+1

而且新家一个数，最多是i+1,所以新的区间最多能延伸到[0,i+i+1]->[0,2*i+1];若数组下一个数不满足(没有数或者有数不满足x<=i+1),此时则必须新增加一个数，让其范围拓展到[0,2*i+1],然后再继续处理数组

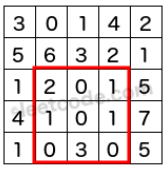
33.

## 304. Range Sum Query 2D - Immutable

Question | My Submissions

Total Accepted: 9181 Total Submissions: 41792 Difficulty: Medium

Given a 2D matrix *matrix*, find the sum of the elements inside the rectangle defined by its upper left corner (*row*1, *col*1) and lower right corner (*row*2, *col*2).




The above rectangle (with the red border) is defined by (row1, col1) = **(2, 1)** and (row2, col2) = **(4, 3)**, which contains sum = **8**.

**Example:**

```
Given matrix = [
  [3, 0, 1, 4, 2],
  [5, 6, 3, 2, 1],
  [1, 2, 0, 1, 5],
  [4, 1, 0, 1, 7],
  [1, 0, 3, 0, 5]
]

sumRegion(2, 1, 4, 3) -> 8
sumRegion(1, 1, 2, 2) -> 11
sumRegion(1, 2, 2, 4) -> 12
```

思路：

每一行采用动态规划，与一维一样

34.

## 332. Reconstruct Itinerary

Question | My Submissions

Total Accepted: 4808 Total Submissions: 20923 Difficulty: Medium

Given a list of airline tickets represented by pairs of departure and arrival airports `[from, to]`, reconstruct the itinerary in order. All of the tickets belong to a man who departs from `JFK`. Thus, the itinerary must begin with `JFK`.

**Note:**

1. If there are multiple valid itineraries, you should return the itinerary that has the smallest lexical order when read as a single string. For example, the itinerary `["JFK", "LGA"]` has a smaller lexical order than `["JFK", "LGB"]`.
2. All airports are represented by three capital letters (IATA code).
3. You may assume all tickets form at least one valid itinerary.

**Example 1:**

`tickets` = `[["MUC", "LHR"], ["JFK", "MUC"], ["SFO", "SJC"], ["LHR", "SFO"]]`

Return `["JFK", "MUC", "LHR", "SFO", "SJC"]`.

**Example 2:**

`tickets` = `[["JFK","SFO"],["JFK","ATL"],["SFO","ATL"],["ATL","JFK"],["ATL","SFO"]]`

Return `["JFK","ATL","JFK","SFO","ATL","SFO"]`.

Another possible reconstruction is `["JFK","SFO","ATL","JFK","ATL","SFO"]`. But it is larger in lexical order.

思路：

简单地回溯法：

因为题目说You may assume all tickets form at least one valid itinerary，所以一定有解，

因此self.dfs一定返回True，而且在选择目的地时一定要排序，这样才能保证第一次搜索到的可行解就是我们需要的解

35.

## 112. Path Sum

Question | My Submissions

Total Accepted: 93430 Total Submissions: 301828 Difficulty: Easy

Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

For example:

Given the below binary tree and `sum = 22`,

```
              5
             / \
            4   8
           /   / \
          11  13  4
         /  \      \
        7    2      1
```

return true, as there exist a root-to-leaf path `5->4->11->2` which sum is 22.

思路：

dfs 深度优先遍历，到达叶子节点的时候判断和是否满足( if root!=None and root.left==None and root.right==None:#到达叶子节点)

36.

## 113. Path Sum II

Question | My Submissions

Total Accepted: 74276 Total Submissions: 266297 Difficulty: Medium

Given a binary tree and a sum, find all root-to-leaf paths where each path's sum equals the given sum.

For example:

Given the below binary tree and `sum = 22`,

```
              5
             / \
            4   8
           /   / \
          11  13  4
         /  \    / \
        7    2  5   1
```

return

```
[
   [5,4,11,2],
   [5,8,4,5]
]
```

思路：

dfs直到叶子节点，在叶子节点处进行判断

37.

## 334. Increasing Triplet Subsequence

Question | My Submissions

Total Accepted: 6303 Total Submissions: 19469 Difficulty: **Medium**

> Given an unsorted array return whether an increasing subsequence of length 3 exists or not in the array.
>
> Formally the function should:
>
> > Return true if there exists *i, j, k*
> > such that *arr[i] < arr[j] < arr[k]* given 0 ≤ *i* < *j* < *k* ≤ *n*-1 else return false.
>
> Your algorithm should run in O(*n*) time complexity and O(*1*) space complexity.
>
> **Examples:**
>
> Given `[1, 2, 3, 4, 5]`,
>
> return `true`.
>
> Given `[5, 4, 3, 2, 1]`,
>
> return `false`.

思路：用(a,b)来表示两个距离最小的递增序列，显然来一个数x,x<=a则更新a;x在(a,b] 考虑更新b,x>b，则返回True
初始化时为(None,None)
来一个数，更新a
再来一个数，如果比a小，则不能构成递增序列，仍然更新a;如果比a大，则可以构成递增序列，更新b
如果(a,b)都没构建完成（说明没找到两个的递增序列），此时肯定也不存在3个的递增序列
构建完成后，按以上规则进行更新，如果出现比b大的则说明一定存在递增的三元组

38.

## 300. Longest Increasing Subsequence

Question | My Submissions

Total Accepted: 20169 Total Submissions: 59937 Difficulty: **Medium**

> Given an unsorted array of integers, find the length of longest increasing subsequence.
>
> For example,
>
> Given `[10, 9, 2, 5, 3, 7, 101, 18]`,
>
> The longest increasing subsequence is `[2, 3, 7, 101]`, therefore the length is `4`. Note that there may be more than one LIS combination, it is only necessary for you to return the length.
>
> Your algorithm should run in O($n^2$) complexity.

思路：
动态规划:
opt[i]表示以nums[i]为最后一个元素的递增序列的长度，显然如果前面有比nums[i]小的，就可以在其基础上再加nums[i],从而导致递增序列的长度再加1
opt[i] = max([opt[j]+1 for j in range(i) if nums[i]>nums[j]]+[1])#注意：最后一个+[1]是为了防止前面没有比nums[i]小的

39.

## 162. Find Peak Element

Question | My Submissions

Total Accepted: 58534 Total Submissions: 178382 Difficulty: **Medium**

> A peak element is an element that is greater than its neighbors.
>
> Given an input array where `num[i] ≠ num[i+1]`, find a peak element and return its index.
>
> The array may contain multiple peaks, in that case return the index to any one of the peaks is fine.
>
> You may imagine that `num[-1] = num[n] = -∞`.
>
> For example, in array `[1, 2, 3, 1]`, 3 is a peak element and your function should return the index number 2.
>
> click to show spoilers.
>
> **Note:**
>
> Your solution should be in logarithmic complexity.
>
> **Credits:**
>
> Special thanks to @ts for adding this problem and creating all test cases.
>
> Subscribe to see which companies asked this question

思路：

```cpp
//2nd pass
/*
1)证明：如果num[-1] = num[n] = -∞.那么峰值一定存在。
因为nums[-1] = -∞.因此 nums[0]>nums[-1],如果峰值不存在，那么nums[0]<nums[1],同理 nums[1]<nums[2] 因为如果nums[2]>nums[1],nums[2]就是峰值
因此，nums[-1]<nums[0]<nums[1]<nums[2]<..<nums[length-1]>nums[length](nums[length] = -∞),因此nums[length-1]一定为峰值
*/
class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        if(nums.size()==1)
            return 0;
        int l = 0,r = nums.size()-1,mid = l+(r-l)/2;
        while(l<=r)
        {
            int mid_l_val = mid==0?INT_MIN:nums[mid-1];
            int mid_r_val = mid==nums.size()-1?INT_MIN:nums[mid+1];
            if(nums[mid]>=mid_l_val&&nums[mid]>=mid_r_val)//峰值,因为题目中没有重复的数，因此遇到重复的数只可能发生在边界
                (如果数组第一个或最后一个为INT_MIN)
                return mid;
            else if(nums[mid]<mid_l_val)//nums[mid]<nums[mid-1],让r = mid-1,尽可能包含峰值
                r = mid-1;
            else if(nums[mid]<mid_r_val)//nums[mid]<nums[mid+1],让l = mid+1,尽可能包含峰值
                l = mid+1;
            mid = l+(r-l)/2;
        }
        return -1;
    }
};
```

40.

## 53. Maximum Subarray

Question | My Submissions

Total Accepted: **101472** Total Submissions: **280414** Difficulty: **Medium**

> Find the contiguous subarray within an array (containing at least one number) which has the largest sum.
>
> For example, given the array `[-2,1,-3,4,-1,2,1,-5,4]`,
>
> the contiguous subarray `[4,-1,2,1]` has the largest sum = `6`.
>
> click to show more practice.
>
> **More practice:**
>
> If you have figured out the O(*n*) solution, try coding another solution using the divide and conquer approach, which is more subtle.
>
> Subscribe to see which companies asked this question

思路：

经典问题：最大连续子数组和问题，标准的动态规划

思路：动态规划;

opt[i] = max(opt[i-1]+A[i],A[i])

41.

## 152. Maximum Product Subarray

Total Accepted: **53666** Total Submissions: **246971** Difficulty: **Medium**

> Find the contiguous subarray within an array (containing at least one number) which has the largest product.
>
> For example, given the array `[2,3,-2,4]`,
>
> the contiguous subarray `[2,3]` has the largest product = `6`.

思路：

思路：

因为最大乘积遇到-1时会进行翻转，可能最大变最小，最小变最大；因此需要维护2个数组，opt_max保存以当前节点为终点的连续最大乘积，opt_min保存以当前节点为终点的连续最小乘积

到当前数时最大的乘积一定在:

opt_max = max(opt_max[i-1]*nums[i],opt_min[i-1]*nums[i],nums[i])

opt_min = min(opt_max[i-1]*nums[i],opt_min[i-1]*nums[i],nums[i])

然后遍历opt_max从中找最大即可(注意不要包含最初初始化为1的值)-------之所以要遍历是因为这个最大至少当前节点的最大乘积，而不是整个数组的最大乘积

42.

## 198. House Robber

Question | My Submissions

Total Accepted: **57797** Total Submissions: **172841** Difficulty: **Easy**

> You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security system connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.
>
> Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight **without alerting the police**.

思路：

不能取相邻的两个数：

动态规划:opt[i] = max{opt[i-2]+A[i],opt[i-1]}

43.

213. House Robber II

Question My Submissions

Total Accepted: 23191 Total Submissions: 77697 Difficulty: Medium

> **Note:** This is an extension of House Robber.
>
> After robbing those houses on that street, the thief has found himself a new place for his thievery so that he will not get too much attention. This time, all houses at this place are **arranged in a circle.** That means the first house is the neighbor of the last one. Meanwhile, the security system for these houses remain the same as for those in the previous street.
>
> Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight **without alerting the police**.

思路：

相对于上一题，仅仅增加了数组的第一个元素和最后一个元素不能同时选择的条件，

因此分两次进行动态规划，

一次去掉第一个元素，保留最后一个元素；

一次去掉最后一个元素，保留第一个元素

44.

292. Nim Game

Question My Submissions

Total Accepted: 51721 Total Submissions: 99943 Difficulty: Easy

> You are playing the following Nim Game with your friend: There is a heap of stones on the table, each time one of you take turns to remove 1 to 3 stones. The one who removes the last stone will be the winner. You will take the first turn to remove the stones.
>
> Both of you are very clever and have optimal strategies for the game. Write a function to determine whether you can win the game given the number of stones in the heap.
>
> For example, if there are 4 stones in the heap, then you will never win the game: no matter 1, 2, or 3 stones you remove, the last stone will always be removed by your friend.
>
> **Hint:**
>
> 1. If there are 5 stones in the heap, could you figure out a way to remove the stones such that you will always be the winner?

思路：

不管是先下棋还是后下棋，只要让剩余的棋子的数量为4的倍数，那么就一定能赢

如果先下，那么剩余棋子肯定不为4的倍数，只要对手采取4的策略，就一定会输

45.

242. Valid Anagram

Total Accepted: 65636 Total Submissions: 159702 Difficulty: Easy

> Given two strings *s* and *t*, write a function to determine if *t* is an anagram of *s*.
>
> For example,
>
> *s* = "anagram", *t* = "nagaram", return true.
>
> *s* = "rat", *t* = "car", return false.
>
> **Note:**
>
> You may assume the string contains only lowercase alphabets.

思路：

注意题意理解：Valid Anagram 的含义是 验证其中的一个字符串能否通过移位得到另一个字符串，每个字符及其个数都不变

思路：用一个hash_map存储第一个字符串中每个字符及其出现的次数~然后遍历另一个字符串，看能否都减为0，若都为0，则表明两个字符串完全相等

s_arr[] 每个字符直接hash到其对应的ascII码的位置

46.

## 49. Group Anagrams

Question | My Submissions

Total Accepted: 68122 Total Submissions: 254011 Difficulty: Medium

Given an array of strings, group anagrams together.

For example, given: `["eat", "tea", "tan", "ate", "nat", "bat"]`,

Return:

```
[
  ["ate", "eat","tea"],
  ["nat","tan"],
  ["bat"]
]
```

**Note:**

1. For the return value, each *inner* list's elements must follow the lexicographic order.
2. All inputs will be in lower-case.

思路：

```
class Solution(object):
    def groupAnagrams(self, strs):
        """
        :type strs: List[str]
        :rtype: List[List[str]]
        思路：
        1）先对整体所有单词按字母序排序，这样可以保证最后每个list的结果都按字母序
        2)将排序后的单词作为key,这样所有的相同字母组合构成的单词都对应一个key
        """
        strs.sort()
        word_map = collections.defaultdict(list)
        for x in strs:
            key = "".join(sorted(list(x)))
            word_map[key].append(x)
        return [val for val in word_map.values()]
```

47.

## 297. Serialize and Deserialize Binary Tree

Question | My Submissions

Total Accepted: 14926 Total Submissions: 55403 Difficulty: Medium

Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment.

Design an algorithm to serialize and deserialize a binary tree. There is no restriction on how your serialization/deserialization algorithm should work. You just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure.

For example, you may serialize the following tree

```
    1
   / \
  2   3
     / \
    4   5
```

as `"[1,2,3,null,null,4,5]"`, just the same as how LeetCode OJ serializes a binary tree. You do not necessarily need to follow this format, so please be creative and come up with different approaches yourself.

**Note:** Do not use class member/global/static variables to store states. Your serialize and deserialize algorithms should be stateless.

思路：

序列和反序列化都需要借助辅助队列

序列化：

1）用层序遍历，如果一个节点可能有孩子(不为空)就作为种子加到队列que里面

2）结果里面只要是是que里面的孩子，就放在res里面

3）最后去除res最后所有的None，用json dumps将res转化为字符串

反序列化：

1）用json loads 将字符串转化为res

2)从res取元素时，同样用一个队列保存所有有孩子的节点(节点不为空)，队列里面从队首出父亲节点到res里面领孩子

48.

## 274. H-Index

Question | My Submissions

Total Accepted: 27691 Total Submissions: 95770 Difficulty: Medium

Given an array of citations (each citation is a non-negative integer) of a researcher, write a function to compute the researcher's h-index.

According to the definition of h-index on Wikipedia: "A scientist has index *h* if *h* of his/her *N* papers have **at least** *h* citations each, and the other *N − h* papers have **no more than** *h* citations each."

For example, given `citations = [3, 0, 6, 1, 5]`, which means the researcher has `5` papers in total and each of them had received `3, 0, 6, 1, 5` citations respectively. Since the researcher has `3` papers with **at least** `3` citations each and the remaining two with **no more than** `3` citations each, his h-index is `3`.

**Note**: If there are several possible values for `h`, the maximum one is taken as the h-index.

**Hint:**

1. An easy approach is to sort the array first.
2. What are the possible values of h-index?
3. A faster approach is to use extra space.

思路：

动态规划：

H-index :opt[i]表示H-index为i时，引用次数>=i有多少篇

1）按题目要求，显然H-index取值范围为0-len(citations)

2)opt[i] = opt[i+1]+count[i] count[i]表示引用次数=i的有多少篇

3）opt从后向前计算，初始化时最后一个初始化为:引用次数>len(citations)的文章数，然后opt下标从Len(citations)-0开始计算

4）满足要求的i为opt[i]>=i,即有i篇文章每篇文章引用次数都>=i

49.

## 275. H-Index II

Total Accepted: 19480 Total Submissions: 60145 Difficulty: Medium

**Follow up** for H-Index: What if the `citations` array is sorted in ascending order? Could you optimize your algorithm?

**Hint:**

1. Expected runtime complexity is in *O*(log *n*) and the input is sorted.

思路：仔细思考H-index的含义：值 >= （大于等于该值的次数） 的（值）

尽量找相等的

如果一个数组从小到大排列：

0 1 2 3 4

那么：值 大，范围往左移，（大于等于该值的次数）才会变大

思路：

```
class Solution(object):
    def hIndex(self, citations):
        """
        :type citations: List[int]
        :rtype: int
        """
        #H-index范围为0-len(citations)
        if not citations:
            return 0
        left,right = 0 ,len(citations)-1
        n = len(citations)
        mid = (left+right)/2
        while left<=right:
            if citations[mid]==n-mid:
                return  n-mid
            elif citations[mid]>n-mid:
                right = mid-1
            else:
                left = mid+1
            mid = (left+right)/2
        return n-left
```

50.

## 284. Peeking Iterator

Question | My Submissions

Total Accepted: 16250 Total Submissions: 49056 Difficulty: Medium

> Given an Iterator class interface with methods: `next()` and `hasNext()`, design and implement a PeekingIterator that support the `peek()` operation -- it essentially peek() at the element that will be returned by the next call to next().
>
> Here is an example. Assume that the iterator is initialized to the beginning of the list: `[1, 2, 3]`.
>
> Call `next()` gets you 1, the first element in the list.
>
> Now you call `peek()` and it returns 2, the next element. Calling `next()` after that ***still*** return 2.
>
> You call `next()` the final time and it returns 3, the last element. Calling `hasNext()` after that should return false.

思路：

刚开始初始化iterator时，先让iterator往后移一步，把下一个元素缓存到self.next_val，
以后只要next，就更新self.next_val

51.

## 173. Binary Search Tree Iterator

Question | My Submissions

Total Accepted: 42823 Total Submissions: 126887 Difficulty: Medium

> Implement an iterator over a binary search tree (BST). Your iterator will be initialized with the root node of a BST.
>
> Calling `next()` will return the next smallest number in the BST.
>
> **Note:** `next()` and `hasNext()` should run in average O(1) time and uses O(*h*) memory, where *h* is the height of the tree.
>
> **Credits:**
>
> Special thanks to @ts for adding this problem and creating all test cases.
>
> Subscribe to see which companies asked this question

思路：

利用二叉排序树的中序遍历是递增序列的性质，采用一个栈缓存中序遍历序列，每当next取出一个值，
就将其右孩子的所有左孩子压入

52.

## 41. First Missing Positive

Total Accepted: 59970 Total Submissions: 254094 Difficulty: Hard

> Given an unsorted integer array, find the first missing positive integer.
>
> For example,
>
> Given `[1,2,0]` return `3`,
>
> and `[3,4,-1,1]` return `2`.
>
> Your algorithm should run in *O*(*n*) time and uses constant space.

思路：

```cpp
/*2nd pass
思路：本题需要找第一个缺失的正数，因此nums里面只放正确的数，0 位置放 1，1位置放 2， 2位置放3， n-1位置放n
本题与上一题的区别在于nums里面只放需要的数，上一题从0开始，本题从1开始（正数）
*/

class Solution {
public:
    int firstMissingPositive(vector<int>& nums) {
         for(int i=0;i<nums.size();i++)
        {
            int target_pos= nums[i]-1;
            while(target_pos>=0&&target_pos<nums.size()&&nums[target_pos]!=nums[i])
            {
                swap(nums[i],nums[target_pos]);
                target_pos= nums[i]-1;
            }
        }
        int res = nums.size()+1;
        for(int i=0;i<nums.size();i++)
        {
            if(i!=nums[i]-1)
            {
                res = i+1;
                break;
            }
        }
        return res;
    }
};
```

53.

## 268. Missing Number

Question | My Submissions

Total Accepted: 42115 Total Submissions: 106717 Difficulty: Medium

> Given an array containing *n* distinct numbers taken from `0, 1, 2, ..., n`, find the one that is missing from the array.
>
> For example,
>
> Given *nums* = `[0, 1, 3]` return `2`.
>
> **Note**:
>
> Your algorithm should run in linear runtime complexity. Could you implement it using only constant extra space complexity?

```cpp
//2nd pass
/*思路：
将i位置上的数与Nums[i]位置上的数交换，如果在合理范围内就交换；这样尽量保证nums[i]放到第i个位置上；
每次交换至少能保证1个数放到合适的位置上，这样交换n次就能将所有有合适位置的数都放到合适的位置上，然后遍历一遍，
遇到第一个nums[i]!=i的数时，就返回
*/
class Solution {
public:
    int missingNumber(vector<int>& nums) {
        for(int i=0;i<nums.size();i++)
        {
            int target_pos= nums[i];
            while(target_pos>=0&&target_pos<nums.size()&&nums[target_pos]!=nums[i])
            {
                swap(nums[i],nums[target_pos]);
                target_pos= nums[i];
            }
        }
        int res = nums.size();
        for(int i=0;i<nums.size();i++)
        {
            if(i!=nums[i])
            {
                res = i;
                break;
            }
        }
        return res;
    }
};
```

54.

## 257. Binary Tree Paths

Question | My Submissions

Total Accepted: 38184 Total Submissions: 139044 Difficulty: Easy

Given a binary tree, return all root-to-leaf paths.

For example, given the following binary tree:

```
   1
 /   \
2     3
 \
  5
```

All root-to-leaf paths are:

```
["1->2->5", "1->3"]
```

思路：
dfs，到叶子节点就把中间节点join,然后放到结果中，特别注意初始化时要把树根先放到path中

55.

## 210. Course Schedule II

Question | My Submissions

Total Accepted: 22555 Total Submissions: 108377 Difficulty: Medium

There are a total of *n* courses you have to take, labeled from `0` to `n - 1`.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: `[0,1]`

Given the total number of courses and a list of prerequisite **pairs**, return the ordering of courses you should take to finish all courses.

There may be multiple correct orders, you just need to return one of them. If it is impossible to finish all courses, return an empty array.

For example:

```
2, [[1,0]]
```

There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correct course order is `[0,1]`

```
4, [[1,0],[2,0],[3,1],[3,2]]
```

There are a total of 4 courses to take. To take course 3 you should have finished both courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0. So one correct course order is `[0,1,2,3]`. Another correct ordering is `[0,2,1,3]`.

```python
class Solution(object):
    def findOrder(self, numCourses, prerequisites):
        """
        :type numCourses: int
        :type prerequisites: List[List[int]]
        :rtype: List[int]
        思路：此题暴露了你很多问题：
        1）如何判断一个有向图中有没有环存在？不断删除叶子节点(出度为0的节点)，如果最终全部删除完了，那么肯定就没有环，否则有环
        2)如何设计数据结构？设计的数据结构要根据具体编程实际方便，比如这一题，如果删除一个节点会导致与其相邻节点的出度减去1，这时为了不重新遍历一遍，就要把出度包含这个节点的所有节点存在一起！！！
        """
        res,course_select,out_d = [],[0]*numCourses,[0]*numCourses
        depend_map = collections.defaultdict(list)
        for x,y in prerequisites:
            depend_map[y].append(x)#x的出度都与y相关，删除y会导致所有x的出度都减小1
            out_d[x]+=1
        count = 0
        while count<numCourses:
            i = 0
            #选择一门课程，出度为0,是叶子节点
            for i in range(numCourses+1):
                if i<numCourses and out_d[i]==0 and course_select[i]==0:#没有选择过的叶子节点
                    break
            if i>=numCourses:#没有满足要求的节点,但课程还没选完
                res = []
                break
            else:
                #选择课程i，修改与i相关的出度-1
                res.append(i)
                course_select[i] =1
                if depend_map[i]:
                    for x in depend_map[i]:
                        out_d[x]-=1
                count+=1
        return res
```



56.

## 207. Course Schedule

Question | My Submissions

Total Accepted: 33966 Total Submissions: 129401 Difficulty: Medium

There are a total of *n* courses you have to take, labeled from `0` to `n - 1`.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: `[0,1]`

Given the total number of courses and a list of prerequisite **pairs**, is it possible for you to finish all courses?

For example:

```
2, [[1,0]]
```

There are a total of 2 courses to take. To take course 1 you should have finished course 0. So it is possible.

```
2, [[1,0],[0,1]]
```

There are a total of 2 courses to take. To take course 1 you should have finished course 0, and to take course 0 you should also have finished course 1. So it is impossible.

思路：参考第210题，如果不断删除叶子节点，最终如果还有剩余而且课程没选完，那么久肯定有环存在，此时就无解

57.

## 240. Search a 2D Matrix II

Question | My Submissions

Total Accepted: 30830 Total Submissions: 91197 Difficulty: Medium

Write an efficient algorithm that searches for a value in an *m* x *n* matrix. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

For example,

Consider the following matrix:

```
[
  [1,   4,  7, 11, 15],
  [2,   5,  8, 12, 19],
  [3,   6,  9, 16, 22],
  [10, 13, 14, 17, 24],
  [18, 21, 23, 26, 30]
]
```

Given **target** = `5`, return `true`.

Given **target** = `20`, return `false`.

## 74. Search a 2D Matrix

Question | My Submissions

Total Accepted: 72845 Total Submissions: 218125 Difficulty: Medium

Write an efficient algorithm that searches for a value in an *m* x *n* matrix. This matrix has the following properties:

- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

For example,

Consider the following matrix:

```
[
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
```

Given **target** = `3`, return `true`.

思路：

以上2个题都是经典题， 并且解法完全相同

从右上角开始, 比较target 和 matrix[i][j]的值. 如果小于target, 则该行不可能有此数, 所以i++; 如果大于target, 则该列不可能有此数,

所以j--. 遇到边界则表明该矩阵不含target.

只要满足行有序，列有序即可使用。算法复杂度O(m+n)（行数+列数）

另：74题特殊解，两次二分查找：

//思路：利用二分查找，查找第一个比Target大的值(left)，确定行；然后再在该行二分查找Target小的值，看能否找到

//需要注意的是:二分查找的位置就是第一个比target大的值，也是该target应该插入的位置

58.

## 241. Different Ways to Add Parentheses

Question | My Submissions

Total Accepted: 18048 Total Submissions: 53934 Difficulty: Medium

Given a string of numbers and operators, return all possible results from computing all the different possible ways to group numbers and operators. The valid operators are `+`, `-` and `*`.

**Example 1**

Input: `"2-1-1"`.

```
((2-1)-1) = 0
(2-(1-1)) = 2
```

Output: `[0, 2]`

**Example 2**

Input: `"2*3-4*5"`

```
(2*(3-(4*5))) = -34
((2*3)-(4*5)) = -14
((2*(3-4))*5) = -10
(2*((3-4)*5)) = -10
(((2*3)-4)*5) = 10
```

Output: `[-34, -14, -10, -10, 10]`

思路：

这道题的思路很巧妙，有点类似于Unique Binary Search Trees II的计算：递归遍历法

遍历所有运算符，在每个运算符的位置拆分成两部分，运算符左边的计算一个结果，运算符右边的计算一个结果

然后把所有的结果都存储起来

```python
class Solution(object):
    def diffWaysToCompute(self, input):
        """
        :type input: str
        :rtype: List[int]
        思路：
        这道题的思路很巧妙，有点类似于Unique Binary Search Trees II的计算
        遍历所有运算符，在每个运算符的位置拆分成两部分，运算符左边的计算一个结果，运算符右边的计算一个结果
        然后把所有的结果都存储起来
        """
        res = []
        for i in range(len(input)):
            if input[i]=="+" or input[i]=="-" or input[i]=="*":
                left_results = self.diffWaysToCompute(input[:i])
                right_results = self.diffWaysToCompute(input[i+1:])
                for x in left_results:
                    for y in right_results:
                        if input[i]=="+":
                            res.append(x+y)
                        elif input[i]=="-":
                            res.append(x-y)
                        else:
                            res.append(x*y)
        if res==[]:
            res.append(int(input))
        return res
```

59.

## 95. Unique Binary Search Trees II

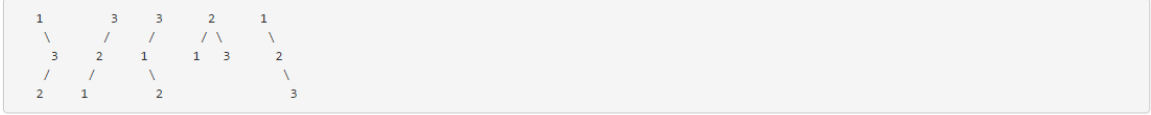
Question | My Submissions

Total Accepted: 51256 Total Submissions: 177577 Difficulty: Medium

Given *n*, generate all structurally unique **BST's** (binary search trees) that store values 1...*n*.

For example,

Given *n* = 3, your program should return all 5 unique BST's shown below.



```python
class Solution(object):
    def generateTrees(self, n):
        """
        :type n: int
        :rtype: List[TreeNode]
        思路：和241道题完全相同的方法
        """
        if n==0:
            return []
        if n==1:
            return [TreeNode(1)]
        return self.genTrees(1,n)

    def genTrees(self,begin,end):
        if begin>end:
            return [None]
        if begin==end:
            return [TreeNode(begin)]
        res = []
        for i in range(begin,end+1):
            lchild_possible = self.genTrees(begin,i-1)
            rchild_possible = self.genTrees(i+1,end)
            for lchild in lchild_possible:
                for rchild in rchild_possible:
                    root = TreeNode(i)
                    root.left,root.right = lchild,rchild
                    res.append(root)
        return res
```

60.

## 238. Product of Array Except Self

Question | My Submissions

Total Accepted: 37458 Total Submissions: 89605 Difficulty: **Medium**

Given an array of *n* integers where *n* > 1, `nums`, return an array `output` such that `output[i]` is equal to the product of all the elements of `nums` except `nums[i]`.

Solve it **without division** and in O(*n*).

For example, given `[1,2,3,4]`, return `[24,12,8,6]`.

**Follow up:**

Could you solve it with constant space complexity? (Note: The output array **does not** count as extra space for the purpose of space complexity analysis.)

思路：

举个例子：[1,4,3,8,2]

1)先正序将第i个数左边的数的乘积都保存起来left->[1,1,4,12,96]

2)再逆序将第i个数右边的数的乘积也都保存起来right->[192,48,16,2,1]

3)然后再将left和right数组中的每个数对应相乘

61.

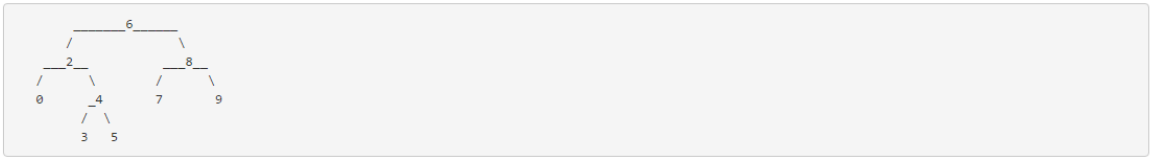
## 235. Lowest Common Ancestor of a Binary Search Tree

Question | My Submissions

Total Accepted: 58344 Total Submissions: 153735 Difficulty: **Easy**

Given a binary search tree (BST), find the lowest common ancestor (LCA) of two given nodes in the BST.

According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes v and w as the lowest node in T that has both v and w as descendants (where we allow **a node to be a descendant of itself**)."




For example, the lowest common ancestor (LCA) of nodes `2` and `8` is `6`. Another example is LCA of nodes `2` and `4` is `2`, since a node can be a descendant of itself according to the LCA definition.

思路：

在二叉查找树种，寻找两个节点的最低公共祖先。

1、如果a、b都比根节点小，则在左子树中递归查找公共节点。

2、如果a、b都比根节点大，则在右子树中查找公共祖先节点。

3、如果a、b一个比根节点大，一个比根节点小，或者有一个等于根节点，则根节点即为最低公共祖先。

```python
class Solution(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        在二叉查找树种，寻找两个节点的最低公共祖先。
        1、如果a、b都比根节点小，则在左子树中递归查找公共节点。
        2、如果a、b都比根节点大，则在右子树中查找公共祖先节点。
        3、如果a、b一个比根节点大，一个比根节点小，或者有一个等于根节点，则根节点即为最低公共祖先。
        """
        if not root:
            return None
        if (root.val-p.val)*(root.val-q.val)<=0:
            return root
        if min(p.val,q.val)>root.val:
            return self.lowestCommonAncestor(root.right,p,q)
        if max(p.val,q.val)<root.val:
            return self.lowestCommonAncestor(root.left,p,q)
```

62.

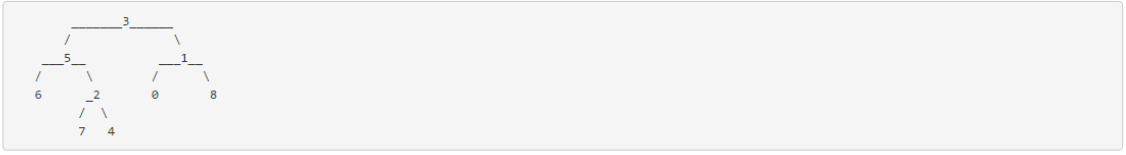
## 236. Lowest Common Ancestor of a Binary Tree

Question | My Submissions

Total Accepted: 36119 Total Submissions: 125845 Difficulty: **Medium**

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes v and w as the lowest node in T that has both v and w as descendants (where we allow **a node to be a descendant of itself**)."




For example, the lowest common ancestor (LCA) of nodes `5` and `1` is `3`. Another example is LCA of nodes `5` and `4` is `5`, since a node can be a descendant of itself according to the LCA definition.

思路：

```python
class Solution(object):
    def lowestCommonAncestor(self, root, p, q):
        """
        :type root: TreeNode
        :type p: TreeNode
        :type q: TreeNode
        :rtype: TreeNode
        """
        if not root:
            return None
        if root==p or root==q:
            return root
        lchild = self.lowestCommonAncestor(root.left,p,q)
        rchild = self.lowestCommonAncestor(root.right,p,q)
        if lchild!=None and rchild!=None:
            return root
        return lchild if lchild else rchild
```

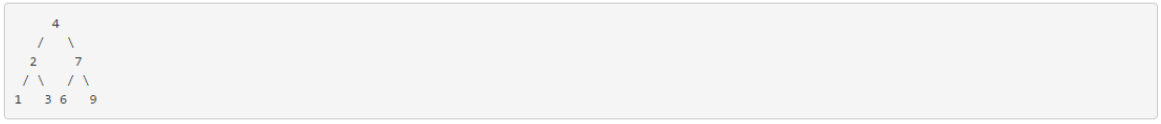
63.

## 226. Invert Binary Tree

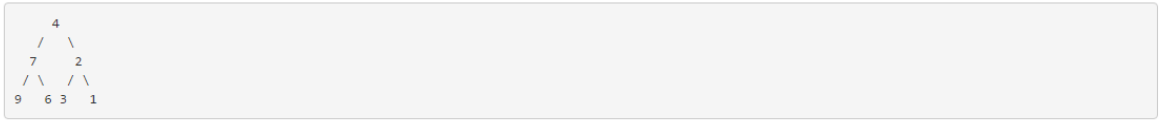
Question | My Submissions

Total Accepted: 75814 Total Submissions: 171088 Difficulty: **Easy**

Invert a binary tree.




to




思路：

如果一个节点不为空，将此节点的左右孩子相互交换，然后递归进行，最后返回这个节点

64.

## 230. Kth Smallest Element in a BST

Question | My Submissions

Total Accepted: 39362 Total Submissions: 107677 Difficulty: Medium

Given a binary search tree, write a function `kthSmallest` to find the **k**th smallest element in it.

**Note:**

You may assume k is always valid, 1 ≤ k ≤ BST's total elements.

**Follow up:**

What if the BST is modified (insert/delete operations) often and you need to find the kth smallest frequently? How would you optimize the kthSmallest routine?

**Hint:**

1. Try to utilize the property of a BST.
2. What if you could modify the BST node's structure?
3. The optimal runtime complexity is O(height of BST).

思路：

```
8  class Solution(object):
9      def kthSmallest(self, root, k):
10         """
11         :type root: TreeNode
12         :type k: int
13         :rtype: int
14         思路：BST的中序遍历是递增的，考察BST的非递归遍历
15         """
16         stack = [root]
17         cnt = 0
18         cur = root
19         while cur.left:#注意二叉树中序遍历的典型操作：1）开始时疯狂的压入左孩子
20             stack.append(cur.left)
21             cur = cur.left
22         while stack:
23             cur = stack.pop()
24             cnt+=1
25             if cnt==k:
26                 return cur.val
27             if cur.right!=None:#注意2）弹出一个节点有右孩子时，切换到右孩子，然后疯狂的压入左孩子
28                 cur = cur.right
29                 while cur:
30                     stack.append(cur)
31                     cur = cur.left
```

65.

## 101. Symmetric Tree

Question | My Submissions

Total Accepted: 98413 Total Submissions: 293564 Difficulty: Easy

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

For example, this binary tree is symmetric:

```
    1
   / \
  2   2
 / \ / \
3  4 4  3
```

But the following is not:

```
    1
   / \
  2   2
   \   \
   3    3
```

思路：

两个对称的节点：要么同时为None,要么都不为None，且值相等；
除此之外，均可断定不相等

66.

## 7. Reverse Integer

Question | My Submissions

Total Accepted: 127569 Total Submissions: 540765 Difficulty: Easy

Reverse digits of an integer.

**Example1:** x = 123, return 321

**Example2:** x = -123, return -321

click to show spoilers.

**Have you thought about this?**

Here are some good questions to ask before coding. Bonus points for you if you have already thought through this!

If the integer's last digit is 0, what should the output be? ie, cases such as 10, 100.

Did you notice that the reversed integer might overflow? Assume the input is a 32-bit integer, then the reverse of 1000000003 overflows. How should you handle such cases?

For the purpose of this problem, assume that your function returns 0 when the reversed integer overflows.

**Update (2014-11-10):**

思路：
依次取个位，然后把数字重新组合起来（特别注意的是：溢出判断:采用预警机制，在计算之前判断是否*10会溢出）

67.

## 202. Happy Number

Question | My Submissions

Total Accepted: 58710 Total Submissions: 163522 Difficulty: Easy

Write an algorithm to determine if a number is "happy".

A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers.

**Example:** 19 is a happy number

$1^2 + 9^2 = 82$

$8^2 + 2^2 = 68$

$6^2 + 8^2 = 100$

$1^2 + 0^2 + 0^2 = 1$

思路：同计算循环小数一样, 如果出现循环, 则无需继续计算,直接返回false即可.

68.

## 38. Count and Say

Question | My Submissions

Total Accepted: 74130 Total Submissions: 262054 Difficulty: Easy

The count-and-say sequence is the sequence of integers beginning as follows:

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

`1` is read off as `"one 1"` or `11`.

`11` is read off as `"two 1s"` or `21`.

`21` is read off as `"one 2"`, then `one 1"` or `1211`.

Given an integer *n*, generate the $n^{th}$ sequence.

Note: The sequence of integers will be represented as a string.

思路：
Two Pointers 双指针法

69.

## 88. Merge Sorted Array

Question | My Submissions

Total Accepted: 91744 Total Submissions: 306914 Difficulty: Easy

Given two sorted integer arrays *nums1* and *nums2*, merge *nums2* into *nums1* as one sorted array.

**Note:**

You may assume that *nums1* has enough space (size that is greater or equal to *m* + *n*) to hold additional elements from *nums2*. The number of elements initialized in *nums1* and *nums2* are *m* and *n* respectively.

思路：

归并排序中合并的步骤，不过由于是要合并到A中，也就是不申请额外空间，因此，为了减少移动A重元素的次数，考虑从后往前逐步合并：从后往前遍历A和B数组，每次把大的数字从A中m+n位置逐步往前放。

时间复杂度：O(m+n)

空间复杂度：O(1)

70.

231. Power of Two

Question | My Submissions

Total Accepted: 60721 Total Submissions: 170381 Difficulty: Easy

> Given an integer, write a function to determine if it is a power of two.
>
> **Credits:**
>
> Special thanks to @jianchao.li.fighter for adding this problem and creating all test cases.
>
> Subscribe to see which companies asked this question

思路：

思路1：2的指数>0,2^0=1,2^1=2。。。

因此:n<=0,return False ;n>0,看n能否整除2，即n/2==n/2.0为整数,为整数继续判断n/2

时间复杂度:O(lgn)

```
    if n<=0:
  return False
if n==1 or n==2:
  return True
if n/2==n/2.0:
  return self.isPowerOfTwo(n/2)
return False
```

思路2：2的指数，则在其二进制表达中肯定只有一个1，如

2^0=1  0000001

2^1=2  0000010

2^2=4  0000100

...

2^-1 一定和它所有位数都相反，因此只用判断n&(n-1)==0

```
return n>0 and (n&(n-1)==0)
```

71.

224. Basic Calculator

Question | My Submissions

Total Accepted: 24408 Total Submissions: 113870 Difficulty: Medium

> Implement a basic calculator to evaluate a simple expression string.
>
> The expression string may contain open `(` and closing parentheses `)`, the plus `+` or minus sign `-`, **non-negative** integers and empty spaces ` `.
>
> You may assume that the given expression is always valid.
>
> Some examples:
>
> ```
> "1 + 1" = 2
> " 2-1 + 2 " = 3
> "(1+(4+5+2)-3)+(6+8)" = 23
> ```
>
> **Note:** **Do not** use the `eval` built-in library function.

1 ) 因为只有"+","-"和括号运算，没有乘除，所以运算符的优先级仅仅由括号决定，否则从左算到右

因此，先解决的第一个问题就是，如何计算一个不含括号，只有加减的表达式?如： “-11+2-4+3”

方法：把所有的数不管正负，全都看成带上符号的操作数：(-11)+(+2)+(-4)+(+3)

用res表示当前操作数,op存储下一个操作数的符号:初始化：res =0,op=1

那么执行计算的过程：

cur=- 是符号 op=-1 res = 0

cur=1 是数字，继续往后读

cur=1 是数字，继续往后读，后面不是数字，cur=11,当前计算结果：res+=(-1*11) ,op=-1,res=-11

cur=+ 是符号，op=1,res=-11

cur=2 是数字，res+=op*cur,res+=(1*2) res=-9

cur=- 是符号，op=-1

cur=4 是数字，后面没有数字，res+=(op(-1)*cur(4)) res = -9-4=-13

cur=+ 是符号, op=1

cur=3 是数字，后面没有数字，res+=(op(1)*cur(3)) res=-13+3 = -10

最终结果为 -10

2)知道以上怎么计算了后，如果有括号，借助栈，将前面的res和op分别压入栈，重置res=0,op=1，开始计算括号里面的值；

如果遇到")"就知道当前括号里面的已经计算完了~ 然后把前一个入栈的res_pre 和 op_pre出栈，那么res = res_pre+op_pre*cur_res(括号里面的计算结果)

重点：每当遇到(就相当于重新开始一轮计算

3)需要注意的是：有效的字符有数字，'+','-','(',')',遇到空格需要直接跳过不处理

72.

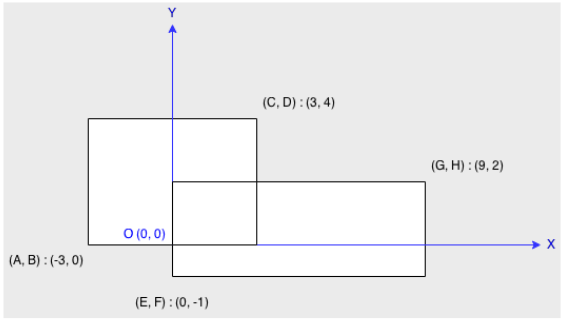
## 223. Rectangle Area

Question | My Submissions

Total Accepted: 32066 Total Submissions: 109351 Difficulty: Easy

Find the total area covered by two **rectilinear** rectangles in a **2D** plane.

Each rectangle is defined by its bottom left corner and top right corner as shown in the figure.




Assume that the total area is never beyond the maximum possible value of **int**.

思路：

```
class Solution(object):
    def computeArea(self, A, B, C, D, E, F, G, H):
        """
        :type A: int
        :type B: int
        :type C: int
        :type D: int
        :type E: int
        :type F: int
        :type G: int
        :type H: int
        :rtype: int
        """
        S = abs(C-A)*abs(D-B)+abs(G-E)*abs(H-F)
        if C<E or B>H or G<A or F>D:#不相交
            return S
        #一定相交
        #求重叠区域，两个左边界取最大，两个右边界取最小
        left_margin,right_margin = max(A,E),min(C,G)
        #两个上边界取最小，两个下边界取最大
        top_margin,bottom_margin = min(D,H),max(B,F)
        #计算重叠面积
        overlap = (right_margin-left_margin)*(top_margin-bottom_margin)
        return S-overlap
```

73.（基本结论题）

## 191. Number of 1 Bits

Question | My Submissions

Total Accepted: 79611 Total Submissions: 211918 Difficulty: Easy

Write a function that takes an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight).

For example, the 32-bit integer '11' has binary representation `00000000000000000000000000001011`, so the function should return 3.

思路:

思路1：不断右移与1相与取低位并计数

```
cnt = 0
while n:
    if n&1:
        cnt+=1
    n>>=1
return cnt
```

思路2：n&(n-1) 会将一个数二进制最右边的1变为0(非常重要的结论！！！会经常用到)

因此直接统计变多少次可以将n变为0即可

```
cnt = 0
while n:
    n = n&(n-1)
    cnt+=1
return cnt
```

74.

## 190. Reverse Bits

Question | My Submissions

Total Accepted: 55675 Total Submissions: 191026 Difficulty: Easy

> Reverse bits of a given 32 bits unsigned integer.
>
> For example, given input 43261596 (represented in binary as **00000010100101000001111010011100**), return 964176192 (represented in binary as **00111001011110000010100101000000**).
>
> **Follow up**:
> If this function is called many times, how would you optimize it?
>
> Related problem: Reverse Integer

思路：建议移动Mask的方式：

将mask每次左移，依次与n相与取n的比特位，取到之后右移，将取到的这一位放在最低位,结果存在lowbit中

然后res左移一位腾出位置与lowbit相与，因为是32位整数，所以初始化获取最低一位，循环31次即可

75.

## 201. Bitwise AND of Numbers Range

Question | My Submissions

Total Accepted: 30249 Total Submissions: 101963 Difficulty: Medium

> Given a range [m, n] where 0 <= m <= n <= 2147483647, return the bitwise AND of all numbers in this range, inclusive.
>
> For example, given the range [5, 7], you should return 4.
>
> **Credits:**
> Special thanks to @amrsaqr for adding this problem and creating all test cases.
>
> Subscribe to see which companies asked this question

思路：

101 - 5　　5&6&7= 4 = 100　实际上就是找5和7都为1 的前缀，逐步去掉5和7后面的比特位，直到两个相等，这时然后再将去掉的比特位都补0，　这时就是最终的结果

110 - 6

111 - 7

76.

## 187. Repeated DNA Sequences

Question | My Submissions

Total Accepted: 39065 Total Submissions: 159589 Difficulty: Medium

> All DNA is composed of a series of nucleotides abbreviated as A, C, G, and T, for example: "ACGAATTCCG". When studying DNA, it is sometimes useful to identify repeated sequences within the DNA.
>
> Write a function to find all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule.
>
> For example,
>
> ```
> Given s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT",
>
> Return:
> ["AAAAACCCCC", "CCCCCAAAAA"].
> ```

思路：

1个编码需要2bit:　A:00 T:01 C:10 G:11　10个编码需要20bit

2^20种可能，每种组合用1个int存，这样大小为1024*1024个int型，用map(int(一种序列对应一个数)：int（出现次数）)来存储

只要出现次数多于1次就添加到结果中，然后将其出现次数变为负无穷，防止重复加入

需要注意的是：生成第一个code后，以后每个字符code在前面一个基础上进行修改

77.搜狐校招笔试题：

## 206. Reverse Linked List

Total Accepted: 91855 Total Submissions: 237845 Difficulty: Easy

> Reverse a singly linked list.

思路：

单链表的逆置，需要特别注意的是：逆置之后头节点最后的指针一定要指向空，否则链表会死循环！！！

78.

92. Reverse Linked List II

Question | My Submissions

Total Accepted: 66455 Total Submissions: 241737 Difficulty: Medium

Reverse a linked list from position *m* to *n*. Do it in-place and in one-pass.

For example:

Given `1->2->3->4->5->NULL`, *m* = 2 and *n* = 4,

return `1->4->3->2->5->NULL`.

**Note:**

Given *m*, *n* satisfy the following condition:

1 ≤ *m* ≤ *n* ≤ length of list.

思路：
链表逆转的拓展，注意最后要修改尾指针的next

79.

147. Insertion Sort List

Total Accepted: 66242 Total Submissions: 229440 Difficulty: Medium

Sort a linked list using insertion sort.

思路：
头节点+Two Pointers：p1指向已经排好序的链表，p2指向当前待排序的节点

80.

222. Count Complete Tree Nodes

Question | My Submissions

Total Accepted: 28664 Total Submissions: 117371 Difficulty: Medium

Given a **complete** binary tree, count the number of nodes.

**Definition of a complete binary tree from Wikipedia:**

In a complete binary tree every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. It can have between 1 and $2^h$ nodes inclusive at the last level h.

Subscribe to see which companies asked this question

思路：
完全二叉树：除最后一层外，每一层上的节点数均达到最大值；在最后一层上只缺少右边的若干结点。
意味着：如果往左的深度和往右的深度一样，那么这棵树一定是满二叉树，满二叉树节点数=2^h-1（h为树深）

81.

221. Maximal Square

Question | My Submissions

Total Accepted: 25600 Total Submissions: 110687 Difficulty: Medium

Given a 2D binary matrix filled with 0's and 1's, find the largest square containing all 1's and return its area.

For example, given the following matrix:

```
1 0 1 0 0
1 0 1 1 1
1 1 1 1 1
1 0 0 1 0
```

Return 4.

思路：
动态规划:opt[i][j]表示以matrix[i][j]为右下角的(里面都为1的)正方形边长
opt[i][j] = min(opt[i-1][j],opt[i][j-1],opt[i-1][j-1])+1 if matrix[i-1][j-1]=='1' else 0

82.

219. Contains Duplicate II

Question | My Submissions

Total Accepted: 49322 Total Submissions: 167491 Difficulty: Easy

Given an array of integers and an integer *k*, find out whether there are two distinct indices *i* and *j* in the array such that **nums[i] = nums[j]** and the difference between *i* and *j* is at most *k*.

思路：
参考查找一个字符串中最大不重复子串，用Hash_map保存一个数字上次出现的位置，下次再次出现时比较当前位置与上次出现的位置，如果小于等于k,说明存在；否则更新hash_map中该字符的位置为当前出现的位置

83.基础性算法(快速选择算法)

215. Kth Largest Element in an Array

Question My Submissions

Total Accepted: 46676 Total Submissions: 144150 Difficulty: Medium

> Find the **k**th largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.
>
> For example,
>
> Given `[3,2,1,5,6,4]` and k = 2, return 5.
>
> **Note:**
>
> You may assume k is always valid, 1 ≤ k ≤ array's length.

思路：

1.堆排序算法：O(klgn),每次从堆顶取一个元素，然后heapfy,取k次，复杂度O(klgn)

2.快速选择算法：从无序数组中找第k大的元素，时间复杂度O(n),最多有n个成为pivot

思路：像快排一样，每次找个pivot,大于pivot的分一堆，小于pivot的分一堆，等于pivot（可能有重复），这样分依次就像这样:

【nums_lt】 pivot1,pivot2,pivot3 【nums_gt】

如果:k<=len(nums_gt) ，说明一定在nums_gt里面找第k个就行了

如果:k>len(nums)-len(nums_lt),说明在nums_lt找第 k-(len(nums)-len(nums_lt))就行了

其它情况说明k就指向pivot,直接返回即可

84.

204. Count Primes

Total Accepted: 55635 Total Submissions: 234601 Difficulty: Easy

> **Description:**
>
> Count the number of prime numbers less than a non-negative number, ***n***.
>
> **Credits:**
>
> Special thanks to @mithmatt for adding this problem and creating all test cases.

思路：

注意:题目求：[0,n-1]有多少个质数(前闭后闭)，分配[0,n-1]共n个空间，每个标志i是否为质数，初始化为[False(0),False(1),True(2),True(3)..]

然后用筛选法开始删除不是质数的数：

重要结论:

1）如果i为质数，那么i*i,i*i+i,i*i+2i,i*i+3i.....肯定都不为质数

2）由1结论可知，如果要判断n是否为质数，i最多只用循环到sqrt(n)即可，因为i到sqrt(n),即可筛除i*i(n),i*i+i,i*i+2i....等

85.

209. Minimum Size Subarray Sum

Question My Submissions

Total Accepted: 33686 Total Submissions: 127890 Difficulty: Medium

> Given an array of **n** positive integers and a positive integer **s**, find the minimal length of a subarray of which the sum ≥ **s**. If there isn't one, return 0 instead.
>
> For example, given the array `[2,3,1,2,4,3]` and `s = 7`,
>
> the subarray `[4,3]` has the minimal length under the problem constraint.

思路：

思路一：用个队列，新加元素后，考虑队列是否可以从头弹出元素，同时更新最小长度值

思路二：Two Pointers

86.

100. Same Tree

Question My Submissions

Total Accepted: 116464 Total Submissions: 271237 Difficulty: Easy

> Given two binary trees, write a function to check if they are equal or not.
>
> Two binary trees are considered equal if they are structurally identical and the nodes have the same value.
>
> Subscribe to see which companies asked this question

思路：

1.两棵树相等，要么都为空

2.如果都不为空的话，节点的值完全相等

87.

## 153. Find Minimum in Rotated Sorted Array

Question | My Submissions

Total Accepted: 83986 Total Submissions: 234954 Difficulty: Medium

> Suppose a sorted array is rotated at some pivot unknown to you beforehand.
>
> (i.e., `0 1 2 4 5 6 7` might become `4 5 6 7 0 1 2`).
>
> Find the minimum element.
>
> You may assume no duplicate exists in the array.

思路：

```
class Solution(object):
    def findMin(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        right = len(nums)-1
        left = 0
        if nums[right]>=nums[left]:#那么一定没有逆序
            return nums[0]
        mid = (left+right)/2
        while left+1!=right:#普通的逆序情况,可以证明：最终right与left一定相邻，且left指向左边最大值，right指向右边最小值
            if nums[mid]>nums[left]:
                left = mid#left始终维持在递增区间，且逐渐增大
            if nums[mid]<nums[right]:
                right = mid#right始终维持在较小的递增区间，且逐渐减小
            mid = (left+right)/2
        return nums[right]
```

88.

## 90. Subsets II

Question | My Submissions

Total Accepted: 63006 Total Submissions: 209320 Difficulty: Medium

> Given a collection of integers that might contain duplicates, ***nums***, return all possible subsets.
>
> **Note:**
>
> - Elements in a subset must be in non-descending order.
> - The solution set must not contain duplicate subsets.
>
> For example,
>
> If ***nums*** = `[1,2,2]`, a solution is:
>
> ```
> [
>   [2],
>   [1],
>   [1,2,2],
>   [2,2],
>   [1,2],
>   []
> ]
> ```

思路：

/*2nd pass

思路：先排序 1,3,4,4,5,

先生成[],[5]

从4开始往前生成:

4(与5不同,直接全部复制[],[5]，然后在前面插入4): [],[5],[4],[4,5]

4(与前一个4相同,复制增加的部分[4],[4,5],然后在前面插入4):[],[5],[4],[4,5],[4,4],[4,4,5]

3(与前一个4不同，全部复制，前面插入3):[],[5],[4],[4,5],[4,4],[4,4,5]， [3],[3,5],[3,4],[3,4,5],[3,4,4],[3,4,4,5]

1 同3

因此，只有与前面一个数不同才会全部复制(修改复制长度)，否则只用复制新增的部分(复制长度不变)

89.

## 309. Best Time to Buy and Sell Stock with Cooldown

Question | My Submissions

Total Accepted: 11033 Total Submissions: 30508 Difficulty: Medium

> Say you have an array for which the $i^{th}$ element is the price of a given stock on day *i*.
>
> Design an algorithm to find the maximum profit. You may complete as many transactions as you like (ie, buy one and sell one share of the stock multiple times) with the following restrictions:
>
> - You may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).
> - After you sell your stock, you cannot buy stock on next day. (ie, cooldown 1 day)
>
> **Example:**
>
> ```
> prices = [1, 2, 3, 0, 2]
> maxProfit = 3
> transactions = [buy, sell, cooldown, buy, sell]
> ```

思路：
*动态规划
* has_stock_porfit[i]:第i天结束后手上有一支股票的最大利润
* no_stock_profit[i]:第i天结束后手上没有股票的最大利润(因为手上没股票可以认为是当天买，当天卖)，可以认为是第i天一定卖了
*
* 关系式:
* has_stock_profit[i] = max{has_stock_profit[i-1]（对应hold）,no_stock_profit[i-2]-prices[i]（第i-2天手上没股票（i-2天卖了），冷静一天，第i天买股票(还没卖出，会导致最大利润减小)）}
* no_stock_profit[i] = max{no_stock_profit[i-1]（对应hold）,has_stock_profit[i-1]+prices[i]（卖出手上的股票）}

90.

## 216. Combination Sum III

Question | My Submissions

Total Accepted: **27496** Total Submissions: **78811** Difficulty: **Medium**

Find all possible combinations of ***k*** numbers that add up to a number ***n***, given that only numbers from 1 to 9 can be used and each combination should be a unique set of numbers.

Ensure that numbers within the set are sorted in ascending order.

***Example 1:***

Input: ***k*** = 3, ***n*** = 7

Output:

```
[[1,2,4]]
```

***Example 2:***

Input: ***k*** = 3, ***n*** = 9

Output:

```
[[1,2,6], [1,3,5], [2,3,4]]
```

思路：常规思路，dfs即可

91.

## 111. Minimum Depth of Binary Tree

Question | My Submissions

Total Accepted: **98238** Total Submissions: **323356** Difficulty: **Easy**

Given a binary tree, find its minimum depth.

The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.

Subscribe to see which companies asked this question

思路：
* 空节点的树高为0，单个节点的树高为1
* 如果一个节点有两个孩子：
* left&&right：min(minDepth(left),minDepth(right))+1
* left&&!right:minDepth(left)+1
* !left&&right:minDepth(right)+1
* !left&&!right:1
* 需要特别注意的是：
*  1
*  / \
*  NULL 2
* 最小树高为2，而不是1！！！！

92.

## 104. Maximum Depth of Binary Tree

Question | My Submissions

Total Accepted: **130557** Total Submissions: **275366** Difficulty: **Easy**

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

思路：
与上一题思路完全一样，不过换成Max

93.

## 172. Factorial Trailing Zeroes

Question | My Submissions

Total Accepted: 53465 Total Submissions: 166273 Difficulty: Easy

> Given an integer *n*, return the number of trailing zeroes in *n*!.
>
> **Note:** Your solution should be in logarithmic time complexity.

思路：
这里我们要求n!末尾有多少个0，因为我们知道0是2和5相乘得到的，而在1到n这个范围内，2的个数要远多于5的个数，所以这里只需计算从1到n这个范围内有多少个5就可以了。
思路已经清楚，下面就是一些具体细节，这个细节还是很重要的。
我在最开始的时候就想错了，直接返回了n / 5，但是看到题目中有要求需要用O(logn)的时间复杂度，就能够想到应该没这么简单。举个例子：
例1
n=15。那么在15! 中 有3个5(来自其中的5, 10, 15)， 所以计算n/5就可以。
例2
n=25。与例1相同，计算n/5，可以得到5个5，分别来自其中的5, 10, 15, 20, 25，但是在25中其实是包含2个5的，所以应该是25/5+25/5^2 =6个5 。
n=35 包含1个5的有35/5=7个：5,10,15,20,25，30，35，包含2个5的有：35/5^2 =1 ,因此共有8个5

94.

## 143. Reorder List

Question | My Submissions

Total Accepted: 61142 Total Submissions: 271987 Difficulty: Medium

> Given a singly linked list $L: L_0→L_1→…→L_{n-1}→L_n$,
> reorder it to: $L_0→L_n→L_1→L_{n-1}→L_2→L_{n-2}→…$
>
> You must do this in-place without altering the nodes' values.
>
> For example,
> Given `{1,2,3,4}`, reorder it to `{1,4,2,3}`.

思路：
* 1.将链表从中间截断，分为前后两个链表(注意:p1的长度=p2长度 or p1的长度=p2长度+1)
* 2.将后面的链表逆置
* 3.用two pointers将两个链表串连起来

95.

## 338. Counting Bits

Question | My Submissions

Total Accepted: 980 Total Submissions: 1693 Difficulty: Medium

> Given a non negative integer number **num**. For every numbers **i** in the range **0 ≤ i ≤ num** calculate the number of 1's in their binary representation and return them as an array.
>
> **Example:**
> For `num = 5` you should return `[0,1,1,2,1,2]`.
>
> **Follow up:**
>
> - It is very easy to come up with a solution with run time **O(n*sizeof(integer))**. But can you do it in linear time **O(n)** /possibly in a single pass?
> - Space complexity should be **O(n)**.
> - Can you do it like a boss? Do it without using any builtin function like **__builtin_popcount** in c++ or in any other language.

思路：
* 0000 0
* ----------
* 0001 1　　在前面的基础上+1
* ----------1 )
* 001-0 1　　与前2个低位一样，在前2个基础上+1
* 001-1 2
* ----------
* 0100 1
* 0101 2
* 0110 2　　与前4个低位一样,在前4个基础上+1
* 0111 3
* ----------2 )
* 1-000 1
* 1-001 2
* 1-010 2
* 1-011 3
* 　　与前8个低位一样,在前8个基础上+1
* 1-100 2

* 1-101 3
* 1-110 3
* 1-111 4
* ----------3 )

96.

## 12. Integer to Roman

Total Accepted: 59688 Total Submissions: 156219 Difficulty: Medium

> Given an integer, convert it to a roman numeral.
>
> Input is guaranteed to be within the range from 1 to 3999.

思路：

```
class Solution(object):
    def intToRoman(self, num):
        """
        :type num: int
        :rtype: str
        思路：先写出千，百，十，个形式，然后依次取个位，将其进行拼接
        """
        thousands = ["","M","MM","MMM"]
        hundreds =["","C","CC","CCC","CD","D","DC","DCC","DCCC","CM"]
        tens = ["","X","XX","XXX","XL","L","LX","LXX","LXXX","XC"]
        ones = ["","I","II","III","IV","V","VI","VII","VIII","IX"]
        digits = ones,tens,hundreds,thousands
        res = ""
        index = 0
        while num:
            digit = num%10
            res = digits[index][digit]+res
            num/=10
            index+=1
        return res
```

/*2nd pass
I(1),V(5),X(10),L(50),C(100),D(500),M(1000)
每次取num的低位，第一次取低位对应ones,第二次取低位对应tens,第三次取低位对应hundreds,第四次取低位对应thousands
将字符串从高位到低位进行拼接即可

97.

## 13. Roman to Integer

Total Accepted: 78273 Total Submissions: 201987 Difficulty: Easy

> Given a roman numeral, convert it to an integer.
>
> Input is guaranteed to be within the range from 1 to 3999.

思路：

比如MCMXCVI
I +1
V 表示5>1,+5
C 表示100，>5 +100
X 表示10,10<100,-10
M 表示1000，1000>100,+1000
C 表示100，100<1000,-100
M 表示1000，1000》=1000，+1000
最后结果就是:2106-110 = 1996
只要在前面出现比后面基数的表示范围小的，都减去，否则直接加上即可

98.

## 16. 3Sum Closest

Question | My Submissions

Total Accepted: 72139 Total Submissions: 250871 Difficulty: Medium

Given an array S of n integers, find three integers in S such that the sum is closest to a given number, target. Return the sum of the three integers. You may assume that each input would have exactly one solution.

```
For example, given array S = {-1 2 1 -4}, and target = 1.

The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
```

思路：

```
        思路：2采用3sum的方法，使结果尽可能向target靠近
        """
        nums.sort()#3-sum问题需要刚开始进行排序
        res = sys.maxsize
        for i in range(len(nums)):
            cur_num = nums[i]
            p1,p2 = i+1,len(nums)-1#在这里将3-sum问题转化为2-sum问题
            while p1<p2:#在two-sum中也是这么做的
                cur_sum = cur_num+nums[p1]+nums[p2]
                cur_dis = abs(cur_sum-target)#尽可能使结果向target靠近，在靠近过程中，更新距离
                res = cur_sum if cur_dis<abs(res-target) else res
                if cur_sum>target:
                    p2-=1
                if cur_sum==target:#只要出现相等的情况，那么target就是距离target最近的3-sum,距离为0
                    return target
                if cur_sum<target:
                    p1+=1
        return res
```

99.

## 35. Search Insert Position

Question | My Submissions

Total Accepted: 99267 Total Submissions: 267818 Difficulty: Medium

Given a sorted array and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You may assume no duplicates in the array.

Here are few examples.

`[1,3,5,6]`, 5 → 2

`[1,3,5,6]`, 2 → 1

`[1,3,5,6]`, 7 → 4

思路：

因为已经排好序，典型的二分查找

特别需要注意的是：对一个已经排序的数组进行二分查找，

如果能找到，就是找到元素的位置；

如果找不到，那么终止时，l的位置就是应该插入的位置

重要结论：二分查找的位置就是第一个比target大的值，也是该target应该插入的位置

100.（腾讯笔试题——大整数乘法）

## 43. Multiply Strings

Total Accepted: 57593 Total Submissions: 248821 Difficulty: Medium

Given two numbers represented as strings, return multiplication of the numbers as a string.

Note: The numbers can be arbitrarily large and are non-negative.

思路：

大整数乘法问题

1）乘数1的第i位（从低位开始）与 乘数2的第j位（从低位开始）相乘，结果是res[i+j]位

不考虑进位，直接累加上去，

2）然后从低位到高位统一处理进位问题，

3）最后从高位开始去0

4）将所得的结果反转

```
   11
*  23
--------
   33-----(i=1,j=0,i+j=1)---(1) ,(i=0,j=0)
  22------(i=1,j=1),(i=0,j=1;i+j=1)-----(2)    i+j相等所以(1)和(2)相加
"""
```

/*2nd pass
思路：长度为len1的数与长度为len2的数相乘，最终结果的长度最多为len1+len2
因此开辟一个len1+len2长的数组
如果：
res[p1+p2] = nums[p1]*nums[p2]
那么最低位为 len1-1+len2_1,最低位后面有一个空格，但是我们想让这个空格留给进位用，因此将
res[len1+len2-2-(p1+p2)] = nums[p1]*nums[p2]
这样，res的最左边为低位，最右边为高位；高位的右边有一个空格用来处理进位
依次从左向右处理进位，然后去掉高位的0，最终结果反向输出

101.

## 66. Plus One

Total Accepted: **92782** Total Submissions: **279715** Difficulty: **Easy**

> Given a non-negative number represented as an array of digits, plus one to the number.
>
> The digits are stored such that the most significant digit is at the head of the list.

参考上题：
思路：与大整数乘法的思路很接近：
1）因为可能有进位，所以需要补一个高位
2）将低位加1
3）开始统一处理进位
4）去除高位的0

102.

## 60. Permutation Sequence

Question My Submissions

Total Accepted: **51527** Total Submissions: **206839** Difficulty: **Medium**

> The set `[1,2,3,…,n]` contains a total of *n*! unique permutations.
>
> By listing and labeling all of the permutations in order,
> We get the following sequence (ie, for *n* = 3):
>
> 1. `"123"`
> 2. `"132"`
> 3. `"213"`
> 4. `"231"`
> 5. `"312"`
> 6. `"321"`
>
> Given *n* and *k*, return the $k^{th}$ permutation sequence.

思路：
注意观察：
1 2 3
1 3 2
---
2 1 3
2 3 1
---
3 1 2
3 2 1
给定n,k
第i=0 个数字，确定分组 [k/(n-(i+1))!]向上取整，求出当前分组号(根据分组号既可以确定当前i个数)
然后，将第i个可用的数从可用数数组中去掉，更新K,k为在当前分组中，第2个数的分组号
总体思路：
k->确定分组号->确定当前位->更新K

103.

## 77. Combinations

Total Accepted: 70889 Total Submissions: 209447 Difficulty: Medium

Given two integers *n* and *k*, return all possible combinations of *k* numbers out of 1 ... *n*.

For example,

If *n* = 4 and *k* = 2, a solution is:

```
[
  [2,4],
  [3,4],
  [2,3],
  [1,2],
  [1,3],
  [1,4],
]
```

思路：
简单地dfs,回溯法

104.

## 64. Minimum Path Sum

Question My Submissions

Total Accepted: 66829 Total Submissions: 193614 Difficulty: Medium

Given a *m* x *n* grid filled with non-negative numbers, find a path from top left to bottom right which *minimizes* the sum of all numbers along its path.

**Note:** You can only move either down or right at any point in time.

思路：
简单的动态规划opt[i][j] = min(opt[i-1][j],opt[i][j-1])+grid[i][j]
需要注意的是初始条件：
需要将第一行和第一列从前往后累加，这是因为题目要求从左上角到右下角的路径，动态规划之前先要保证已有的基础(第一行和第一列)上，
每个都能到达左上角。这道题与求字符串之间的编辑距离很像

105.

## 102. Binary Tree Level Order Traversal

Total Accepted: 96590 Total Submissions: 298588 Difficulty: Easy

Given a binary tree, return the *level order* traversal of its nodes' values. (ie, from left to right, level by level).

For example:

Given binary tree `{3,9,20,#,#,15,7}`,

```
    3
   / \
  9  20
    /  \
   15   7
```

return its level order traversal as:

```
[
  [3],
  [9,20],
  [15,7]
]
```

思路：
层序遍历需要用队列和计数器，而不是栈！

## 107. Binary Tree Level Order Traversal II

Question | My Submissions

Total Accepted: 75287 Total Submissions: 224260 Difficulty: Easy

Given a binary tree, return the *bottom-up level order* traversal of its nodes' values. (ie, from left to right, level by level from leaf to root).

For example:

Given binary tree `{3,9,20,#,#,15,7}`,

```
    3
   / \
  9  20
    /  \
   15   7
```

return its bottom-up level order traversal as:

```
[
  [15,7],
  [9,20],
  [3]
]
```

思路：
与上题完全相同，将结果倒序输出即可

106.

## 110. Balanced Binary Tree

Question | My Submissions

Total Accepted: 104054 Total Submissions: 308171 Difficulty: Easy

Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of *every* node never differ by more than 1.

思路：

```
class Solution(object):
    def isBalanced(self, root):
        """
        :type root: TreeNode
        :rtype: bool
        思路：
        1.注意求二叉树高的递归方法（常考）:1+max(self.getHeight(root.left),self.getHeight(root.right))
        2.如果一棵树是平衡二叉树：则 1）左右子树高度差<=1  2）左子树是平衡二叉树 3）右子树也是平衡二叉树
        """
        if not root:
            return True
        return  abs(self.getHeight(root.left)-self.getHeight(root.right))<=1 and self.isBalanced(root.left) and self.isBalanced(root.right)

    def getHeight(self,root):
        if not root:
            return 0
        return 1+max(self.getHeight(root.left),self.getHeight(root.right))
```

107.

## 59. Spiral Matrix II

Total Accepted: 49861 Total Submissions: 144382 Difficulty: Medium

Given an integer $n$, generate a square matrix filled with elements from 1 to $n^2$ in spiral order.

For example,

Given $n$ = `3`,

You should return the following matrix:

```
[
 [ 1, 2, 3 ],
 [ 8, 9, 4 ],
 [ 7, 6, 5 ]
]
```

思路：

left,right,top,bottom = 0,n-1,0,n-1
row:top  col:[left,right]
row:[top+1,bottom] col:right
row: bottom col:[right-1,left]
row:[bottom-1,top+1] cols:left
每经过一轮,update边界:
left+=1,right-=1 top+=1 bottom-=1
要填n^2个数，总共循环math.pow(n,2)次

108.

## 108. Convert Sorted Array to Binary Search Tree

Total Accepted: **71638** Total Submissions: **194377** Difficulty: **Medium**

> Given an array where elements are sorted in ascending order, convert it to a height balanced BST.

思路：
要求生成平衡二叉排序树：
*则尽可能的用中值作为二叉树的根节点，这样对于已经排好序的数组，中值左边肯定比中值小，中值右边肯定比中值大
*左边的中值作为左子树的根节点，右边的中值作为右子树的根节点，递归进行下去即可

109.

## 119. Pascal's Triangle II

Total Accepted: **70824** Total Submissions: **221985** Difficulty: **Easy**

> Given an index *k*, return the $k^{th}$ row of the Pascal's triangle.
>
> For example, given *k* = 3,
>
> Return `[1,3,3,1]`.

思路：
1 k=0
1 1 k=1
1 2 1 k=2
1 3 3 1 k=3
观察1：求第k行，只需要k+1个空间
观察2：第k行的，第0和第k个都为1
观察3：从后往前更新：A[i] = A[i]+A[i-1]
如果从前往后，需要保存下一个值，用于下次更新
1 2 1 ->1 3(把2更新成3) 3(需要2和1更新，但此时2已经被修改为3)，因此需要从后往前更新

110.

## 103. Binary Tree Zigzag Level Order Traversal

Question | Editorial Solution | My Submissions

Total Accepted: **57343** Total Submissions: **202309** Difficulty: **Medium**

> Given a binary tree, return the *zigzag level order* traversal of its nodes' values. (ie, from left to right, then right to left for the next level and alternate between).
>
> For example:
>
> Given binary tree `{3,9,20,#,#,15,7}`,

```
    3
   / \
  9  20
    /  \
   15   7
```

> return its zigzag level order traversal as:

```
[
  [3],
  [20,9],
  [15,7]
]
```

思路：
1.二叉树的层序遍历：队列和计数器
2.分奇偶进行翻转即可

111.

## 155. Min Stack

Total Accepted: 66678 Total Submissions: 305472 Difficulty: Easy

> Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.
>
> - push(x) -- Push element x onto stack.
> - pop() -- Removes the element on top of the stack.
> - top() -- Get the top element.
> - getMin() -- Retrieve the minimum element in the stack.

思路：

1)用一个栈行使正常栈的功能，用于压入，弹出，访问栈顶元素

2)用一个辅助栈保存当前最小的元素，如果压入元素不大于当前元素，那么也压入最小伴随栈；出栈的时候，如果是最小伴随栈当前的最下元素出栈，

那么也把最小伴随栈的当前栈顶元素弹出

如：

Main Stack: 2 3 1 4 5

Auxliary Stack:[2] [2] [2,1] [2,1] [2,1]

112.

## 130. Surrounded Regions

Total Accepted: 50272 Total Submissions: 315221 Difficulty: Medium

> Given a 2D board containing `'X'` and `'O'`, capture all regions surrounded by `'X'`.
>
> A region is captured by flipping all `'O'`s into `'X'`s in that surrounded region.
>
> For example,
>
> ```
> X X X X
> X O O X
> X X O X
> X O X X
> ```
>
> After running your function, the board should be:
>
> ```
> X X X X
> X X X X
> X X X X
> X O X X
> ```

思路：

用"*"标记一定可以逃出去的点，

从边缘为"O"的修改为"*",作为seed,开始bfs或dfs：如果seed旁边有"O"的修改为"*",然后将其加入种子队列，直到队列为空为止

然后将所有的"*"修改为"O",剩下的修改为"X"

若不用"*"则bfs过程加入种子时，有可能成环，死循环

113.

## 200. Number of Islands

Question | Editorial Solution | My Submissions

Total Accepted: 40990 Total Submissions: 150084 Difficulty: Medium

Given a 2d grid map of `'1'` s (land) and `'0'` s (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

***Example 1:***

```
11110
11010
11000
00000
```

Answer: 1

***Example 2:***

```
11000
11000
00100
00011
```

Answer: 3

思路：

与上题相同的思路：

1.遍历整个grid,遇到1，修改为"*",加入seed_que,开始bfs的过程（bfs遇到1，都修改为"*"）,bfs完成之后cnt+=1

2.继续遍历过程，再次遇到1，说明刚刚bfs扩展不到这里，重新将这个作为种子，再次进行Bfs,完成之后cnt+=1

直到找不到为1的节点，遍历完成

114.

## 148. Sort List

Total Accepted: 67565 Total Submissions: 275031 Difficulty: Medium

> Sort a linked list in $O(n \log n)$ time using constant space complexity.

思路：

对链表进行归并排序

注意：

1）特别需要注意的是如何分割链表，

比如：1->2->3->4->NULL

分割之后应该为：L1:1->2->NULL L2:3->4->NULL

所以分割链表应该做2件事：1）给前一个链表最后添加结束符 2）返回后一个链表的头节点

分割链表可以参考two pointers,p1走1步，p2走2步；这样等走2步的到达结尾，走一步的p1刚好到达中间;从p1开始分

2）合并链表L1,L2时：为保持一致性，建议新增一个链表L3,将L1和L2都往L3上合并

115.

## 151. Reverse Words in a String

Total Accepted: 95903 Total Submissions: 612527 Difficulty: Medium

> Given an input string, reverse the string word by word.
>
> For example,
>
> Given s = "`the sky is blue`",
>
> return "`blue is sky the`".

思路：

此题为典型的单词翻转问题：先去除句子左右的空格

1）从后往前找第一个非空单词

2）用words_stack搜集单词，当遇到下一个空格时，表明此单词已经结束，将words_stack的单词输出到sentence_stack中，然后加个空格

3）注意最后一个单词不能靠空格触发放到sentence_stack中，需要在循环结束后手动放到sentence_stack

116.

## 165. Compare Version Numbers

Question | Editorial Solution | My Submissions

Total Accepted: 49146 Total Submissions: 285408 Difficulty: Easy

Compare two version numbers *version1* and *version2*.

If *version1* > *version2* return 1, if *version1* < *version2* return -1, otherwise return 0.

You may assume that the version strings are non-empty and contain only digits and the `.` character.

The `.` character does not represent a decimal point and is used to separate number sequences.

For instance, `2.5` is not "two and a half" or "half way to version three", it is the fifth second-level revision of the second first-level revision.

Here is an example of version numbers ordering:

```
0.1 < 1.1 < 1.2 < 13.37
```

思路：
* 1)查找"."的位置，如果找到就将"."之前的数转化为整数；如果没找到就将全部的数转换为整数；对两个数进行比较
* 2）然后将其移到"."之后的一个位置或者结尾（注意：如果一个字符串已经到结尾，那么始终维持在结尾位置）
* 3）等到两个数都到结尾后，进行相等比较，否则只根据当前大小进行>或<比较

117.

## 150. Evaluate Reverse Polish Notation

Question | Editorial Solution | My Submissions

Total Accepted: 62148 Total Submissions: 266548 Difficulty: Medium

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are `+`, `-`, `*`, `/`. Each operand may be an integer or another expression.

Some examples:

```
["2", "1", "+", "3", "*"] -> ((2 + 1) * 3) -> 9
["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6
```

思路：
* 遇到数字直接压入栈，遇到运算符弹出栈里面前2个数（先弹出来数字1，后弹出来数字2），用数字2 运算符 数字1 ，然后将结果压入栈；
* 最后栈中一定仅剩一个数，这个数就是最终的运算结果

运算过程参照下图：

The infix expression "5 + ((1 + 2) × 4) − 3" can be written down like this in RPN:

5 1 2 + 4 × + 3 −

The expression is evaluated left-to-right, with the inputs interpreted as shown in t
after the *Operation* given in the middle column has taken place):

| Input | Operation | Stack | Comment |
|---|---|---|---|
| 5 | Push value | 5 | |
| 1 | Push value | 1<br>5 | |
| 2 | Push value | 2<br>1<br>5 | |
| + | Add | 3<br>5 | Pop two values (1, 2) and push result (3) |
| 4 | Push value | 4<br>3<br>5 | |
| × | Multiply | 12<br>5 | Pop two values (3, 4) and push result (12) |
| + | Add | 17 | Pop two values (5, 12) and push result (17) |
| 3 | Push value | 3<br>17 | |
| − | Subtract | 14 | Pop two values (17, 3) and push result (14) |
| | Result | 14 | |

118.

## 166. Fraction to Recurring Decimal

Total Accepted: **28692** Total Submissions: **192194** Difficulty: **Medium**

> Given two integers representing the numerator and denominator of a fraction, return the fraction in string format.
>
> If the fractional part is repeating, enclose the repeating part in parentheses.
>
> For example,
>
> - Given numerator = 1, denominator = 2, return "0.5".
> - Given numerator = 2, denominator = 1, return "2".
> - Given numerator = 2, denominator = 3, return "0.(6)".

思路：

如果余数已经出现过，那么就表明肯定出现了循环,因此需要保存每次余数，同时为了获得最后的结果，也要保存每一次的商

1）维护一个商数组和一个余数的字典

2) 每当用cur_remainder/denominator 求出当前商后，用cur_remainder%denominator求出当前的余数，看余数是否出现，如果出现，则表明出现了

循环，根据上次余数出现的位置和当前的位置之间就是循环体

举例：

4/333

pos ：0 1 2 3

res ：0 0 1 2

remainder: 4 40 67 4-》注意：这里的4在pos=0时出现过，因此pos=1到当前pos=3就是循环体(012)

注意：当前remainder到下一步要*10，相当于除法中的补0

注意一些边界情形：

1）res[0]一定是整数位，如：80/5 = 16 所有的整数位都在res[0]中保存

2）需要判断循环节的位置，如果没有循环(余数没有重复)，res[1:]如果不为空，那么res[1:]就是所有的小数部分(否则没有小数部分也就不用

在整数部分后加".")

3)如果有循环节，最终结果由整数部分(res[0]),小数部分(从res的下标1到循环开始的位置)，循环部分(循环节开始和终止之间的res)组成

结果拼接方法为: 整数部分 + "." + 小数部分 + "("+ 循环部分+")"(如果小数部分为空则不加)

119.

## 168. Excel Sheet Column Title

Total Accepted: **56761** Total Submissions: **265366** Difficulty: **Easy**

> Given a positive integer, return its corresponding column title as appear in an Excel sheet.
>
> For example:
>
> ```
> 1 -> A
> 2 -> B
> 3 -> C
> ...
> 26 -> Z
> 27 -> AA
> 28 -> AB
> ```

思路：

实际上就是将10进制数(每位0-9)->26进制（每位0-25,对应A-Z）

---

注意从0开始计数，28(从1开始计数)->27(从0开始计数，0相当于1)

（28-1）%26 = 1 ----------> B

(28-1)/26 = 1(十进制意义上的1，相当于0) ----->1-1 =0 ----A

AB

因此最终结果为 AB

120.

## 171. Excel Sheet Column Number

Total Accepted: 73022 Total Submissions: 177454 Difficulty: Easy

> Related to question Excel Sheet Column Title
>
> Given a column title as appear in an Excel sheet, return its corresponding column number.
>
> For example:

```
A -> 1
B -> 2
C -> 3
...
Z -> 26
AA -> 27
AB -> 28
```

思路：将26进制转为10进制数:
举个例子:
AB
B->2*(26^0) = 2
A->1*(26^1) = 26
res = 2+26 =28

121.

## 208. Implement Trie (Prefix Tree)

Total Accepted: 32894 Total Submissions: 129991 Difficulty: Medium

> Implement a trie with `insert`, `search`, and `startsWith` methods.
>
> **Note:**
>
> You may assume that all inputs are consist of lowercase letters `a-z`.

思路：
1.python中实现前缀树，存放孩子用dict
2.注意前缀树节点中要有一个标志位，表明当前节点是不是表示一个单词的结尾

122.

## 211. Add and Search Word - Data structure design

Question | Editorial Solution | My Submissions

Total Accepted: 22554 Total Submissions: 111330 Difficulty: Medium

> Design a data structure that supports the following two operations:

```
void addWord(word)
bool search(word)
```

> search(word) can search a literal word or a regular expression string containing only letters `a-z` or `.`. A `.` means it can represent any one letter.
>
> For example:

```
addWord("bad")
addWord("dad")
addWord("mad")
search("pad") -> false
search("bad") -> true
search(".ad") -> true
search("b..") -> true
```

思路：
构建前缀数，并在前缀树中进行递归匹配

123.

## 220. Contains Duplicate III

Question | Editorial Solution | My Submissions

Total Accepted: 25507 Total Submissions: 138419 Difficulty: Medium

> Given an array of integers, find out whether there are two distinct indices *i* and *j* in the array such that the difference between **nums[i]** and **nums[j]** is at most *t* and the difference between *i* and *j* is at most *k*.

思路：
//关键:给定val，在大小为k的集合里查找离val最近的数(O(lgk)才能AC，若为O(k)会超时)
用cpp中multiset构建BST来存储k个元素

124.

## 337. House Robber III

Question | Editorial Solution | My Submissions

Total Accepted: 5985 Total Submissions: 16206 Difficulty: Medium

> The thief has found himself a new place for his thievery again. There is only one entrance to this area, called the "root." Besides the root, each house has one and only one parent house. After a tour, the smart thief realized that "all houses in this place forms a binary tree". It will automatically contact the police if two directly-linked houses were broken into on the same night.
>
> Determine the maximum amount of money the thief can rob tonight without alerting the police.
>
> **Example 1:**

```
    3
   / \
  2   3
   \   \
    3   1
```

> Maximum amount of money the thief can rob = 3 + 3 + 1 = **7**.
>
> **Example 2:**

```
     3
    / \
   4   5
  / \   \
 1   3   1
```

> Maximum amount of money the thief can rob = 4 + 5 = **9**.

思路：
动态规划:
opt[cur_root][0]:表示当前结点不能抢的最大收益
opt[cur_root][1]:表示当前结点能抢时的最大收益
显然有:
opt[cur_root][0] = rob(cur_root.left,CanRob=True)+rob(cur_root.right,CanRob=True)
opt[cur_root][1] = max(opt[cur_root][0],#可抢时不抢
root.val+rob(root.left,CanRob=False)+rob(root.right,CanRob=False)#可以抢时抢
)
递归计算时肯定会超时，因为会有很多重复计算，因此采用一个opt来保存当前的已经计算的中间结果.
最终的结果就是opt[root][1]---因为刚开始时，根节点是可抢可不抢的.

125.

## 335. Self Crossing

Question | Editorial Solution | My Submissions

Total Accepted: 4171 Total Submissions: 23133 Difficulty: Medium

You are given an array *x* of `n` positive numbers. You start at point `(0,0)` and moves `x[0]` metres to the north, then `x[1]` metres to the west, `x[2]` metres to the south, `x[3]` metres to the east and so on. In other words, after each move your direction changes counter-clockwise.

Write a one-pass algorithm with `O(1)` extra space to determine, if your path crosses itself, or not.

**Example 1:**

```
Given x = [2, 1, 1, 2],
┌───┐
│   │
└───┼──>
    │

Return true (self crossing)
```

**Example 2:**

```
Given x = [1, 2, 3, 4],
┌──────┐
│      │
│
│
└─────────────>

Return false (not self crossing)
```

**Example 3:**

```
Given x = [1, 1, 1, 1],
┌───┐
│   │
└───┼>

Return true (self crossing)
```



```python
class Solution(object):
    def isSelfCrossing(self, x):
        """
        :type x: List[int]
        :rtype: bool
        思路：
        1）总结出相交的模式，然后依次轮流判断是否满足
        1 2 3 4 5 2 8 7
        ->判断 1 2 3 4 5 2 不满足模式
        ->判断 2 3 4 5 2 8 不满足模式
        ->判断 3 4 5 2 8 7 不满足模式
        ->判断 4 5 2 8 7 0 不满足模式
        ->判断 5 2 8 7 0 0 不满足模式
        ....
        2)每6个数为一组表示一个模式：
                    b                           b
          +-----------------+           +-----------------+
          |                 |           |                 |
          |                 |           |                 | a
        c |                 |         c |                 |
          |                 | a         |                 |    f
          +------------>    |           |                 | <----+
                    d       |           |                 |      | e
                            |           |                 |
                                        +-----------------------+
                                                        d
        1）d从左边与a相交                2）f从右边与a相交

        相交分为以上2种情况：
        1)第一种情况需要4条线，第二种情况需要6跳线；为统一起见，6个数判定一次，4跳线相交的模式可以认为e=f=0(补0)
        2)a,b,c三条线不管怎么都不会相交,因此约束条件从第4跳线d开始
        3)以上两种情形都要求:d>=b>0(注意b=0说明是补的0，这时不能认为是模式),为保证d能交上a:a>=c ;为保证f能交上a: a>=c-e and f>=d-b
        """
        if len(x)<=3:#如果少于3个数，肯定不会相交
            return False
        return any([(d>=b>0 and (a>=c  or (a>=c-e>0 and f>=d-b)))#相交模式
                for a,b ,c,d,e,f in ((x[i:i+6]+[0]*6)[:6]
                for i in range(len(x)))])
```

126.

## 23. Merge k Sorted Lists

Question | Editorial Solution | My Submissions

Total Accepted: 81208 Total Submissions: 350139 Difficulty: Hard

Merge *k* sorted linked lists and return it as one sorted list. Analyze and describe its complexity.

思路：模仿归并排序，将链表的列表进行递归二分，将分到的两边的列表进行合并

127.

## 315. Count of Smaller Numbers After Self

Question | Editorial Solution | My Submissions

Total Accepted: 8834 Total Submissions: 29376 Difficulty: Hard

You are given an integer array *nums* and you have to return a new *counts* array. The *counts* array has the property where `counts[i]` is the number of smaller elements to the right of `nums[i]`.

**Example:**

```
Given nums = [5, 2, 6, 1]

To the right of 5 there are 2 smaller elements (2 and 1).
To the right of 2 there is only 1 smaller element (1).
To the right of 6 there is 1 smaller element (1).
To the right of 1 there is 0 smaller element.
```

Return the array `[2, 1, 1, 0]`.

思路：

/*

思路：

1）用二叉排序树进行计数，每个BST节点记录以当前节点为根节点的子树上<当前节点的有多少个，=当前节点的有多少个；这样就有点儿类似于以二叉排序树的根节点对其进行分段统计

2）将输入的节点逆序插入BST，当插入节点的同时对<=当前节点的进行计数

举例：9 5 2 6 4

1)插入4

4(lt_cnt=0,equal_cnt=1)

---

ret = 0

2)插入6

4(lt_cnt=0,equal_cnt=1)

\

6(lt_cnt=0,equal_cnt=1)

---

ret = 4(equal_cnt=1)+4(lt_cnt=0) =1

3)插入2

4(lt_cnt=1,equal_cnt=1)

/ \

2(lt_cnt=0,equal_cnt=1) 6(lt_cnt=0,equal_cnt=1)

---

ret = 0

4)插入5

4(lt_cnt=1,equal_cnt=1)

/ \

2(lt_cnt=0,equal_cnt=1) 6(lt_cnt=1,equal_cnt=1)

/

5(lt_cnt=0,equal_cnt=1)

---

ret = 4(lt_cnt=1,equal_cnt=1) = 2 （从根节点往右走才计算）

5)插入9

4(lt_cnt=1,equal_cnt=1)

/ \

2(lt_cnt=0,equal_cnt=1) 6(lt_cnt=1,equal_cnt=1)

/ \

5(lt_cnt=0,equal_cnt=1) 9(lt_cnt=0,equal_cnt=1)

---

ret = 4(lt_cnt=1,equal_cnt=1) + 6(lt_cnt=1,equal_cnt=1) =4 （从根节点往右走才计算）

128.

## 312. Burst Balloons

Question | Editorial Solution | My Submissions

Total Accepted: 7773 Total Submissions: 21448 Difficulty: Hard

Given `n` balloons, indexed from `0` to `n-1`. Each balloon is painted with a number on it represented by array `nums`. You are asked to burst all the balloons. If the you burst balloon `i` you will get `nums[left] * nums[i] * nums[right]` coins. Here `left` and `right` are adjacent indices of `i`. After the burst, the `left` and `right` then becomes adjacent.

Find the maximum coins you can collect by bursting the balloons wisely.

**Note:**

(1) You may imagine `nums[-1] = nums[n] = 1`. They are not real therefore you can not burst them.

(2) 0 ≤ `n` ≤ 500, 0 ≤ `nums[i]` ≤ 100

**Example:**

Given `[3, 1, 5, 8]`

Return `167`

```
    nums = [3,1,5,8] --> [3,5,8] -->   [3,8]   -->  [8]  --> []
   coins =  3*1*5      +  3*5*8    +  1*3*8      + 1*8*1   = 167
```

思路：
动态规划:
opt(i,j)表示在（i,j）范围内(不含第i,j个气球)能得到的最大分数。
给定nums数组，前后补1后(nums[0]=1,nums[nums.size(0-1]=1)，则nums数组所能得到的最大分数就是opt(0,nums.size()-1);
动规方程:
opt(i,j)= max(opt(i,j),opt(i,m)+opt(m,j)+A[i]*A[m]*A[j])（不含i,j）
含义：opt(i,j)的最大值为从(i,j)中挑一个气球m最后爆炸，则opt(i,m)和opt(m,j)可分为两个独立的子问题，气球m最后爆炸得分为nums[i]*nums[m]*nums[j]
关键：如果气球m不是最后一个爆炸，则opt(i,m)和opt(m,j)不独立！
注意:i,j是当前气球范围的边界(可能是nums[0],nums[nums.size()-1]，也可能是以其它气球作为边界)

由opt(i,j)的含义可知：
初始化时,opt(i,i)=0,opt(i,i+1)=0---范围内没有气球　opt(i,i+2)=nums[i]*nums[i+1]*nums[i+2](只有一个气球)

观察递推表达式：
(i,m) -> (i,j)

|
(m,j)
j从下往上计算，i从左往右，m从左往右
*/

129.

## 71. Simplify Path ★

Total Accepted: 50138 Total Submissions: 229577 Difficulty: Medium

Given an absolute path for a file (Unix-style), simplify it.

For example,

**path** = `"/home/"`, => `"/home"`

**path** = `"/a/./b/../../c/"`, => `"/c"`

思路：
用栈来保存路径中间的目录，把路径中/ /之间的内容全部抽取出来：
1）如果没有以"/"结尾则在后面添加"/"结尾
2)如果为""或者"."或者".."是不能加入路径的，必须就地处理，""或者"."直接跳过，".."直接回退
3）最后根据路径栈中的元素重建路径，如果路径本身为空，则返回"/",否则在每个目录前加"/"输出

130.

## 342. Power of Four ★

Total Accepted: **984** Total Submissions: **2800** Difficulty: **Easy**

> Given an integer (signed 32 bits), write a function to check whether it is a power of 4.
>
> **Example:**
>
> Given num = 16, return true. Given num = 5, return false.
>
> **Follow up**: Could you solve it without loops/recursion?

思路：
1）要是4的指数首先得是2的指数,因此满足n&(n-1)==0
2）其次，其二进制末尾的0个数还得是2的倍数，二进制末尾的0个数用 log2(n&(-n))可求得，因此满足 log2(n&(-n))%2==0

131.

## 224. Basic Calculator ★

Question | Editorial Solution | My Submissions

Total Accepted: **27040** Total Submissions: **122207** Difficulty: **Hard**

> Implement a basic calculator to evaluate a simple expression string.
>
> The expression string may contain open `(` and closing parentheses `)`, the plus `+` or minus sign `-`, **non-negative** integers and empty spaces ` `.
>
> You may assume that the given expression is always valid.
>
> Some examples:
>
> ```
> "1 + 1" = 2
> " 2-1 + 2 " = 3
> "(1+(4+5+2)-3)+(6+8)" = 23
> ```
>
> **Note:** **Do not** use the `eval` built-in library function.

思路：
1）因为只有"+","-"和括号运算，没有乘除，所以运算符的优先级仅仅由括号决定，否则从左算到右
因此，先解决的第一个问题就是，如何计算一个不含括号，只有加减的表达式?如： “-11+2-4+3”
方法：把所有的数不管正负，全都看成带上符号的操作数：(-11)+(+2)+(-4)+(+3)
用res表示当前操作数,op存储下一个操作数的符号:初始化：res =0,op=1
那么执行计算的过程：
cur=- 是符号 op=-1 res = 0
cur=1 是数字，继续往后读
cur=1 是数字，继续往后读，后面不是数字，cur=11,当前计算结果：res+=(-1*11) ,op=-1,res=-11
cur=+ 是符号，op=1,res=-11
cur=2 是数字，res+=op*cur,res+=(1*2) res=-9
cur=- 是符号，op=-1
cur=4 是数字，后面没有数字，res+=(op(-1)*cur(4)) res = -9-4=-13
cur=+ 是符号, op=1
cur=3 是数字，后面没有数字，res+=(op(1)*cur(3)) res=-13+3 = -10
最终结果为 -10
2)知道以上怎么计算了后，如果有括号，借助栈，将前面的res和op分别压入栈，重置res=0,op=1，开始计算括号里面的值；
如果遇到")"就知道当前括号里面的已经计算完了~ 然后把前一个入栈的res_pre 和 op_pre出栈，那么res = res_pre+op_pre*cur_res(括号里面的计算结果)
重点：每当遇到(就相当于重新开始一轮计算
3)需要注意的是：有效的字符有数字，'+','-','(',')',遇到空格需要直接跳过不处理

132.

## 227. Basic Calculator II ★

Question | Editorial Solution | My Submissions

Total Accepted: **21280** Total Submissions: **85504** Difficulty: **Medium**

> Implement a basic calculator to evaluate a simple expression string.
>
> The expression string contains only **non-negative** integers, `+`, `-`, `*`, `/` operators and empty spaces ` `. The integer division should truncate toward zero.
>
> You may assume that the given expression is always valid.
>
> Some examples:
>
> ```
> "3+2*2" = 7
> " 3/2 " = 1
> " 3+5 / 2 " = 5
> ```
>
> **Note:** **Do not** use the `eval` built-in library function.

思路：把"+-"视为单目运算符，就是下一个运算数的符号；把乘除视为双目运算符，就是两个数之间的操作
没有乘除时，把所有的运算数与之前面的符号视为一个有符号数，直接压入栈
这样：当有乘除的时候，先计算弹出上一个入栈的有符号数，然后和当前的有符号数进行乘除运算，然后再压入栈~
所以：现在问题的核心转为如何从一个表达式中提取所有带符号的操作数：
1)初始操作数的默认运算符为+，知道遇到'-'改变默认运算符
2)提取数字完后，和当前的运算符一起构成一个有符号数，然后判断有没有乘除：如果有乘除，出栈运算后入栈，然后清除乘除标志位，把当前符号设置为正；
没有乘除，直接入栈，把当前符号设置为正！！（注意：也就是说，只要发生一次入栈，符号标志位都必须回正，因为符号只作用于当前操作数，如果是因为乘除入栈，还得清楚乘除标志位）
3)最后将栈里面的所有数进行累加

133.

## 42. Trapping Rain Water ★

Question | Editorial Solution | My Submissions

Total Accepted: **64753** Total Submissions: **201176** Difficulty: **Hard**

Given *n* non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

For example,

Given `[0,1,0,2,1,0,1,3,2,1,2,1]`, return `6`.


The above elevation map is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped. **Thanks Marcos** for contributing this image!

思路：
单纯只考虑有2根柱子height[l],height[r]的情形：
两根柱子之间能容纳的水由water_level = min(height[l],height[r])来决定，但是由于中间还会有很多石柱子挡着，
能容纳的水就等于sum(water_level-height[i])(i：[l+1,r-1])
但是上面这种情形实际上是考虑(l,r)之间的所有石柱子都比water_level地的情形下，如果中间有一个比water_level高的石柱子呢，
比如[2,0,0，5,0,0，3]这种情形开始l=2,r=3,warer_level=2;但是后来遇到5之后，water_level实际上已经提升为3了，
因此由地的一端开始移动，当遇到比水平面高的柱子时，这是就要更新water_level,重新转换为以上的问题

134.

## 30. Substring with Concatenation of All Words ★

Question | Editorial Solution | My Submissions

Total Accepted: **53518** Total Submissions: **257202** Difficulty: **Hard**

You are given a string, **s**, and a list of words, **words**, that are all of the same length. Find all starting indices of substring(s) in **s** that is a concatenation of each word in **words** exactly once and without any intervening characters.

For example, given:

**s**: `"barfoothefoobarman"`

**words**: `["foo", "bar"]`

You should return the indices: `[0,9]`.

(order does not matter).

思路：
因为不用关注词出现的先后顺序，因此可以用词袋模型：
将words中的所有单词及其次数放到一个all_words袋子中，每次从s中一个开始位置start起，往后找words.size()个单词，并用一个新的袋子has_words存储出现的单词及其次数，
如果出现的单词在all_words中找不到，说明当前start找不到，start往后移动重新开始找

135.

## 33. Search in Rotated Sorted Array ★

Question | Editorial Solution | My Submissions

Total Accepted: **99425** Total Submissions: **328646** Difficulty: **Hard**

Suppose a sorted array is rotated at some pivot unknown to you beforehand.

(i.e., `0 1 2 4 5 6 7` might become `4 5 6 7 0 1 2`).

You are given a target value to search. If found in the array return its index, otherwise return -1.

You may assume no duplicate exists in the array.

Subscribe to see which companies asked this question

## 81. Search in Rotated Sorted Array II ★

Question | Editorial Solution | My Submissions

Total Accepted: **61442** Total Submissions: **192505** Difficulty: **Medium**

> Follow up for "Search in Rotated Sorted Array":
>
> What if *duplicates* are allowed?
>
> Would this affect the run-time complexity? How and why?
>
> Write a function to determine if a given target is in the array.

当有重复元素时，这时题目只会要求你判断target是否存在，这时只用将nums[mid]与nums[right]进行比较，既可以确定target落在哪个区间，
若nums[mid]==nums[right],将right往后移动
比如：{1，3，1，1，1，1}
如果A[right] <= A[mid] 条件就不能确定[start mid]区间为递增有序序列，我们就把该条件分成两个子条件：
A[right] < A[mid] 则 [start mid]区间为递增有序序列
A[right] = A[mid] 则[right mid]区间不能确定，那就right--，往下一步看看即可
以上2题代码完全相同！！

```
6  class Solution {
7  public:
8      int search(vector<int>& nums, int target) {
9          int left=0,right=nums.size()-1;
10         int mid = left+(right-left)/2;
11         while(left<=right)
12         {
13             if(nums[mid]==target)return mid;
14             else if(nums[mid]>nums[right])//中间元素>左边元素，[left,mid]是一个递增区间
15             {
16                 if(target>=nums[left]&&target<=nums[mid])
17                     right = mid-1;//因为前面已经确定mid也不是target,所以right可以跳过mid
18                 else
19                     left = mid+1;
20             }
21             else if(nums[mid]<nums[right])//中间元素<左边元素，[mid,right]是一个递增区间
22             {
23                 if(target>=nums[mid]&&target<=nums[right])
24                     left = mid+1;
25                 else
26                     right = mid-1;
27             }
28             else//nums[mid]==nums[right]
29                 right--;
30             mid = left+(right-left)/2;
31         }
32         return -1;
33     }
34 };
```

136.

## 153. Find Minimum in Rotated Sorted Array ★

Total Accepted: **89094** Total Submissions: **246488** Difficulty: **Medium**

> Suppose a sorted array is rotated at some pivot unknown to you beforehand.
>
> (i.e., `0 1 2 4 5 6 7` might become `4 5 6 7 0 1 2`).
>
> Find the minimum element.
>
> You may assume no duplicate exists in the array.

# 154. Find Minimum in Rotated Sorted Array II ★

Total Accepted: **49817** Total Submissions: **145328** Difficulty: **Hard**

> *Follow up* for "Find Minimum in Rotated Sorted Array":
> What if *duplicates* are allowed?
>
> Would this affect the run-time complexity? How and why?

Suppose a sorted array is rotated at some pivot unknown to you beforehand.

(i.e., `0 1 2 4 5 6 7` might become `4 5 6 7 0 1 2`).

Find the minimum element.

The array may contain duplicates.

注意：以上两题都是完全相同的代码：

```
class Solution {
public:
    int findMin(vector<int>& nums) {
        if(nums[nums.size()-1]>nums[0])//not rotated
            return nums[0];
        int l = 0,r =nums.size()-1,mid = l+(r-l)/2;
        //rotated
        while(l<=r)
        {
            if(nums[mid]>nums[r])//一定在左侧上升沿,因为nums[r]<nums[mid]，因此mid肯定不是最小值
                l = mid+1;
            else if(nums[mid]<nums[r])//一定在右侧上升沿
                r = mid;
            else//no idea
                r--;//mid==r,可以排除r
            mid = l+(r-l)/2;
        }
        return nums[l];

    }
};
```

137.

# 343. Integer Break ★

Question | Editorial Solution | My Submissions

Total Accepted: **3941** Total Submissions: **9685** Difficulty: **Medium**

> Given a positive integer *n*, break it into the sum of **at least** two positive integers and maximize the product of those integers. Return the maximum product you can get.
>
> For example, given *n* = 2, return 1 (2 = 1 + 1); given *n* = 10, return 36 (10 = 3 + 3 + 4).
>
> **Note**: you may assume that *n* is not less than 2.
>
> **Hint:**
>
> 1. There is a simple O(n) solution to this problem.
> 2. You may check the breaking results of *n* ranging from 7 to 10 to discover the regularities.

思路：

1)不需要任何数学知识，直接动规：

```
class Solution {
public:
    int integerBreak(int n) {
        vector<int>opt(n+1,INT_MIN);
        opt[1] = 1;
        opt[2] = 1;
        opt[3] = 2;
        opt[4] = 4;
        for(int i=5;i<=n;i++)
            for(int j=1;j<=i/2;j++)
            opt[i] = max(opt[i],max(j,opt[j])*max(opt[i-j],i-j));//j是不拆分，opt[j]是拆分j的最大乘积;实际上如果j>=5，opt[j]>j,因此一定要拆分。这一步的意思是：把i拆分成两部分，考虑这两部分拆分，不拆分能得到的最大乘积
```

```
        return opt[n];
    }
```
2)如果具备以下数学知识：
1. 2^3<3^2,  和为6
   4^3 < 3^4  和为 12
   5^3 < 3^5
   6^3 < 3^6
   因此，应该尽可能拆分成更多的3，乘积才能最大
   breaking it into 3s turns out to be the most efficient
2.
3*（n-3）>=n
n>=4.5,因此只要n>=5,都应该拆分成3，但是n<=4时，拆分不一定能保证最大
可以将动态规划中
```
        for(int j=1;j<=i/2;j++)
        opt[i] = max(opt[i],max(j,opt[j])*max(opt[i-j],i-j));
```
简化为:
```
    opt[i] = 3*max(opt[i-3],i-3);//相当于直接选择3进行拆分
```

138.

341. Flatten Nested List Iterator ★

Question | Editorial Solution | My Submissions

Total Accepted: **2612** Total Submissions: **12847** Difficulty: **Medium**

Given a nested list of integers, implement an iterator to flatten it.

Each element is either an integer, or a list -- whose elements may also be integers or other lists.

**Example 1:**

Given the list `[[1,1],2,[1,1]]`.

By calling *next* repeatedly until *hasNext* returns false, the order of elements returned by *next* should be: `[1,1,2,1,1]`.

**Example 2:**

Given the list `[1,[4,[6]]]`.

By calling *next* repeatedly until *hasNext* returns false, the order of elements returned by *next* should be: `[1,4,6]`.

思路：
将列表展开实际上dfs的过程，先用dfs将列表里面的内容用一个vector存起来，然后每次从vector最开始取一个元素；
其思路类似于bst的iterator

139.

124. Binary Tree Maximum Path Sum ★

Question | Editorial Solution | My Submissions

Total Accepted: **63514** Total Submissions: **272619** Difficulty: **Hard**

Given a binary tree, find the maximum path sum.

For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path does not need to go through the root.

For example:

Given the below binary tree,

```
       1
      / \
     2   3
```

Return `6`.

思路：
以当前结点为根可以选择2个分支,更新最终的结果
如果root需要返回值，则root返回给上层用，此时root作为上层的一个分支，只能选择一个分支返回
(选2个分支就不是沿着parent-child的路线)，如果root->val+max(left,right)<0,则连root也舍弃，全部都不要

```
      1
     / \
    -2  3
    /\
   4 5
```

以-2作为根节点时，沿着parent-child的路线：值为-2+4+5（更新值）
以1作为根节点时，沿着（5->-2，是-2节点的返回值）->1->3 的路线取得最大值，注意：4不能加上，否则就不是parent-child的路线

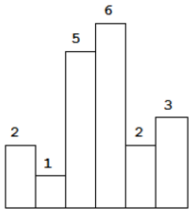
140.

## 84. Largest Rectangle in Histogram ★

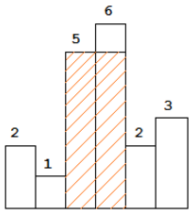
Question | Editorial Solution | My Submissions

Total Accepted: **58775** Total Submissions: **243077** Difficulty: **Hard**

Given *n* non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of largest rectangle in the histogram.




Above is a histogram where width of each bar is 1, given height = `[2,1,5,6,2,3]`.




The largest rectangle is shown in the shaded area, which has area = `10` unit.

For example,

Given heights = `[2,1,5,6,2,3]`,

return `10`.



思路：
维持一个递增的子序列，为什么？
因为递增的子序列以当前元素开始的最大面积一下子就能求出来！！
比如：1 3 5 7 4
最大 1 3 5 7 ，以7为开始的最大面积为1*7，以5开始的最大面积为2*5,以3开始的最大面积为3*3,以1开始的最大面积为1*4,可以直接计算！！！
当来一个4，要保证递增还要保证连续性，弹出5,7，同时计算7*1=7,5*2=10，
然后再压入 1 3 4 4 4！！！为什么弹出5,7的同时要压入4,4？？因为1）要保证递增 2）同时需要保证连续性 3）由原来的5,7（现在的4,4）开始的最大面积已经计算过了，以后用到原来5,7开始的最大面积取决于后面最小的值，因此直接把5,7设置为当前的最小值4,4

141.

## 85. Maximal Rectangle ★

Total Accepted: **41328** Total Submissions: **176096** Difficulty: **Hard**

Given a 2D binary matrix filled with 0's and 1's, find the largest rectangle containing all ones and return its area.

思路：
假设把矩阵沿着某一行分开，然后把分开的行作为底面，将自底面往上的矩阵看成一个直方图（histogram）。直方图的中每个项的高度就是从底面行开始往上的1的数量。根据Largest Rectangle in Histogram就可以求出当前行作为矩阵下边缘的一个最大矩阵。接下来如果对每一行都做一次Largest Rectangle in Histogram，从其中选出最大的矩阵，那么它就是整个矩阵中面积最大的子矩阵。时间复杂度O(n^3),空间复杂度O(n)

142.

## 87. Scramble String ★

Total Accepted: **45548** Total Submissions: **171987** Difficulty: **Hard**

Given a string *s1*, we may represent it as a binary tree by partitioning it to two non-empty substrings recursively.

思路：
按题目描述：
两个字符串互为scramble string 的必要条件是：
1)组成两字符串的字符和个数完全相同！！！--------用Map做
2)两个字符串可以分别进行分割s1_1,s1_2;s2_1,s2_2;其中至少有一种分割使得
（s1_1，s2_1)与(s1_2,s2_2)或者(s1_1,s2_2)与(s1_2,s2_1)也能满足1）条件
递归进行！！

143.

# 347. Top K Frequent Elements ★

Total Accepted: **419** Total Submissions: **935** Difficulty: **Medium**

> Given a non-empty array of integers, return the ***k*** most frequent elements.
>
> For example,
>
> Given `[1,1,1,2,2,3]` and k = 2, return `[1,2]`.
>
> **Note:**
>
> - You may assume *k* is always valid, 1 ≤ *k* ≤ number of unique elements.
> - Your algorithm's time complexity **must be** better than O(*n* log *n*), where *n* is the array's size.

思路：
1)用Map对字符的出现次数进行统计
2）根据字符的出现次数建立大根堆，从堆中取出topK个元素即可

144.