Python 数据结构与算法题

第一章 顺序表

难度：Easy

1.1 删除排序数组中的重复数字 Remove Duplicates from Sorted Array

给定一个排序数组，在原数组中删除重复出现的数字，使得每个元素只出现一次，并且返回新的数组的长度。

不要使用额外的数组空间，必须在原地没有额外空间的条件下完成。

样例
给出数组 A =[1,1,2]，你的函数应该返回长度 2，此时 A=[1,2]。

LintCode: http://www.lintcode.com/zh-cn/problem/remove-duplicates-from-sorted-array/
LeetCode: https://leetcode.com/problems/remove-duplicates-from-sorted-array/#/description

http://blog.csdn.net/nerv3x3/article/details/3465656
http://blog.csdn.net/u013291394/article/details/50407435
http://blog.csdn.net/lilong_dream/article/details/19757047

1.2 两数之和 Two Sum

给一个整数数组，找到两个数使得他们的和等于一个给定的数 *target*。

你需要实现的函数 `twoSum` 需要返回这两个数的下标, 并且第一个下标小于第二个下标。注意这里下标的范围是 1 到 *n*，不是以 `0` 开头。

注意事项
你可以假设只有一组答案。

样例
给出 numbers = `[2, 7, 11, 15]`, target = `9`, 返回 `[1, 2]`.

LintCode: http://www.lintcode.com/zh-cn/problem/two-sum/
LeetCode: https://leetcode.com/problems/two-sum/#/description

http://blog.csdn.net/nerv3x3/article/details/37339471
http://blog.csdn.net/u013291394/article/details/50269289

1.3 删除元素 Remove Element

给定一个数组和一个值，在原地删除与值相同的数字，返回新数组的长度。

元素的顺序可以改变，并且对新的数组不会有影响。

样例
给出一个数组 [0,4,4,0,0,2,4,4]，和值 4

返回 4 并且 4 个元素的新数组为[0,0,0,2]
LintCode: http://www.lintcode.com/zh-cn/problem/remove-element/
LeetCode: https://leetcode.com/problems/remove-element/#/description

http://blog.csdn.net/nerv3x3/article/details/2920214
http://blog.csdn.net/u013291394/article/details/50407437

1.4 加一 Plus One

给一个由包含一串数字的列表组成的非负整数加上一。

注意点：

- 列表前面的数字表示高位
- 注意最高位也可能进位

例子：

输入: [1, 2, 3, 4, 9]

输出: [1, 2, 3, 5, 0]

LintCode: http://www.lintcode.com/zh-cn/problem/plus-one/
LeetCode: https://leetcode.com/problems/plus-one/#/description

http://blog.csdn.net/nerv3x3/article/details/37336603
http://blog.csdn.net/u013291394/article/details/50534662

1.5 爬楼梯 Climbing Stairs

假设你正在爬楼梯，需要 n 步你才能到达顶部。但每次你只能爬一步或者两步，你能有多少种不同的方法爬到楼顶部？

样例

比如 n=3，1+1+1=1+2=2+1=3，共有 3 中不同的方法

返回 3

LintCode: http://www.lintcode.com/zh-cn/problem/climbing-stairs/
LeetCode: https://leetcode.com/problems/climbing-stairs/#/description

http://blog.csdn.net/nerv3x3/article/details/37329935
http://blog.csdn.net/u013291394/article/details/50544768

1.6 落单的数 Single Number

给出 2*n + 1 个的数字，除其中一个数字之外其他每个数字均出现两次，找到这个数字。

样例
给出 [1,2,2,1,3,4,3]，返回 4

LintCode: http://www.lintcode.com/problem/single-number/
LeetCode: https://leetcode.com/problems/single-number/#/description

http://blog.csdn.net/nerv3x3/article/details/3465512
http://blog.csdn.net/u013291394/article/details/51298527
http://blog.csdn.net/monkeyduck/article/details/37991201

难度：Medium

1.7 删除排序数组中的重复数字 Remove Duplicates from Sorted Array II

其他条件不变，如果可以允许出现两次重复将如何处理？

样例

给出数组 *nums* = `[1,1,1,2,2,3]`,

你的函数应当返回长度 length = `5`，且前 5 个元素分别为 `1`, `1`, `2`, `2` and `3`.

LintCode: http://www.lintcode.com/zh-cn/problem/remove-duplicates-from-sorted-array-ii/
LeetCode: https://leetcode.com/problems/remove-duplicates-from-sorted-array-ii/#/description

http://blog.csdn.net/nerv3x3/article/details/39453377
http://blog.csdn.net/nerv3x3/article/details/38929163
http://blog.csdn.net/u013291394/article/details/50577447

1.8 搜索旋转排序数组 Search in Rotated Sorted Array

假设有一个排序的按未知的旋转轴旋转的数组(比如，0 1 2 4 5 6 7 可能成为 4 5 6 7 0 1 2)。给定一个目标值进行搜索，如果在数组中找到目标值返回数组中的索引位置，否则返回-1。

你可以假设数组中不存在重复的元素。

样例

给出[4, 5, 1, 2, 3]和 target=1，返回 2

给出[4, 5, 1, 2, 3]和 target=0，返回 -1

LintCode: http://www.lintcode.com/zh-cn/problem/search-in-rotated-sorted-array/

LeetCode: https://leetcode.com/problems/search-in-rotated-sorted-array/#/description

http://blog.csdn.net/nerv3x3/article/details/39453385

http://blog.csdn.net/u013291394/article/details/50439774

1.9 搜索旋转排序数组 Search in Rotated Sorted Array II

其他条件不变，假如有重复元素又将如何？

是否会影响运行时间复杂度？

如何影响？

为何会影响？

写出一个函数判断给定的目标值是否出现在数组中。

样例

给出[3,4,4,5,7,0,1,2]和 target=4，返回 true

LintCode: http://www.lintcode.com/zh-cn/problem/search-in-rotated-sorted-array-ii/

LeetCode: https://leetcode.com/problems/search-in-rotated-sorted-array-ii/#/description

http://www.jiuzhang.com/solutions/search-in-rotated-sorted-array-ii/

http://blog.csdn.net/u013291394/article/details/50582829

1.10 三数之和 3Sum

给出一个有 n 个整数的数组 S，在 S 中找到三个整数 a, b, c，找到所有使得 a + b + c = 0 的三元组。

注意事项

在三元组(a, b, c)，要求 a <= b <= c。

结果不能包含重复的三元组。

样例

如 S = {-1 0 1 2 -1 -4}, 你需要返回的三元组集合的是：

(-1, 0, 1)

(-1, -1, 2)

LintCode: http://www.lintcode.com/zh-cn/problem/3sum/
LeetCode: https://leetcode.com/problems/3sum/#/description

http://blog.csdn.net/u013291394/article/details/50358081
https://github.com/williamwhe/leetcode/blob/master/3sum.py

1.11 最接近的三数之和 3Sum Closest

给一个包含 *n* 个整数的数组 *S*, 找到和与给定整数 *target* 最接近的三元组，返回这三个数的和。

注意事项

只需要返回三元组之和，无需返回三元组本身

样例

例如 *S* = `[-1, 2, 1, -4]` and target = `1`. 和最接近 1 的三元组是 -1 + 2 + 1 = 2.

LintCode: http://www.lintcode.com/zh-cn/problem/3sum-closest/
LeetCode: https://leetcode.com/problems/3sum-closest/#/description

http://blog.csdn.net/u013291394/article/details/50359700

1.12 四数之和 4Sum

给一个包含 n 个数的整数数组 S，在 S 中找到所有使得和为给定整数 target 的四元组(a, b, c, d)。

注意事项

四元组(a, b, c, d)中，需要满足 a <= b <= c <= d

答案中不可以包含重复的四元组。

样例

例如，对于给定的整数数组 S=[1, 0, -1, 0, -2, 2] 和 target=0. 满足要求的四元组集合为：

(-1, 0, 0, 1)

(-2, -1, 1, 2)

(-2, 0, 0, 2)

LintCode: http://www.lintcode.com/zh-cn/problem/4sum/

LeetCode: https://leetcode.com/problems/4sum/#/description

http://blog.csdn.net/u013291394/article/details/50374323
https://github.com/williamwhe/leetcode/blob/master/4sum.py

1.13 下一个排列 Next Permutation

给定一个整数数组来表示排列，找出其之后的一个排列。

注意事项

排列中可能包含重复的整数

样例

给出排列`[1,3,2,3]`，其下一个排列是`[1,3,3,2]`

给出排列`[4,3,2,1]`，其下一个排列是`[1,2,3,4]`

LintCode: http://www.lintcode.com/zh-cn/problem/next-permutation/
LeetCode: https://leetcode.com/problems/next-permutation/#/description

http://blog.csdn.net/u013291394/article/details/50432456

1.14 第 K 个排列 Permutation Sequence

给定 *n* 和 *k*，求 `123..n` 组成的排列中的第 *k* 个排列。

注意事项

1 ≤ n ≤ 9

样例

对于 `n = 3`, 所有的排列如下：

```
123
132
213
231
312
321
```

如果 `k = 4`, 第 4 个排列为，`231`.

LintCode: http://www.lintcode.com/zh-cn/problem/permutation-sequence/
LeetCode: https://leetcode.com/problems/permutation-sequence/#/description

http://blog.csdn.net/u013291394/article/details/50538700
https://github.com/williamwhe/leetcode/blob/master/permutation_seq.py

1.15 判断数独是否合法 Valid Sudoku

请判定一个数独是否有效。

该数独可能只填充了部分数字，其中缺少的数字用 . 表示。

注意事项

一个合法的数独（仅部分填充）并不一定是可解的。我们仅需使填充的空格有效即可。

说明

什么是 数独?

- http://sudoku.com.au/TheRules.aspx
- http://baike.baidu.com/subview/961/10842669.htm

样例

The following partially filed sudoku is valid.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

LintCode: http://www.lintcode.com/zh-cn/problem/valid-sudoku/
LeetCode: https://leetcode.com/problems/valid-sudoku/#/description

http://blog.csdn.net/u013291394/article/details/50450198
https://github.com/williamwhe/leetcode/blob/master/valid-sudoku.py

1.16 旋转图像 Rotate Image

给定一个 N×N 的二维矩阵表示图像，90 度顺时针旋转图像。

样例

给出一个矩形[[1,2],[3,4]]，90 度顺时针旋转后，返回[[3,1],[4,2]]
LintCode: http://www.lintcode.com/zh-cn/problem/rotate-image/
LeetCode: https://leetcode.com/problems/rotate-image/#/description

http://blog.csdn.net/nerv3x3/article/details/37968757
http://blog.csdn.net/u013291394/article/details/50482248
http://blog.csdn.net/monkeyduck/article/details/39431989

1.17 格雷编码 Gray Code

格雷编码是一个二进制数字系统，在该系统中，两个连续的数值仅有一个二进制的差异。

给定一个非负整数 `n`，表示该代码中所有二进制的总数，请找出其格雷编码顺序。一个格雷编码顺序必须以 `0` 开始，并覆盖所有的 2^n 个整数。

注意事项

对于给定的 `n`，其格雷编码顺序并不唯一。

根据以上定义， `[0,2,3,1]` 也是一个有效的格雷编码顺序。

样例
给定 `n = 2`， 返回 `[0,1,3,2]`。其格雷编码顺序为：

```
00 - 0
01 - 1
11 - 3
10 - 2
```

LintCode: http://www.lintcode.com/zh-cn/problem/gray-code/
LeetCode: https://leetcode.com/problems/gray-code/#/description

http://blog.csdn.net/u013291394/article/details/50589865

1.18 矩阵归零 Set Matrix Zeroes

给定一个 m×n 矩阵，如果一个元素是 0，则将其所在行和列全部元素变成 0。

需要在原矩阵上完成操作。

样例
给出一个矩阵

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

```
  [0, 3]
]
```

返回

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

LintCode: http://www.lintcode.com/zh-cn/problem/set-matrix-zeroes/
LeetCode: https://leetcode.com/problems/set-matrix-zeroes/#/description

http://blog.csdn.net/nerv3x3/article/details/37338299
http://blog.csdn.net/u013291394/article/details/50550587

1.19 加油站 Gas Station

在一条环路上有 *N* 个加油站，其中第 *i* 个加油站有汽油 `gas[i]`，并且从第_i_个加油站前往第_i_+1 个加油站需要消耗汽油 `cost[i]`。

你有一辆油箱容量无限大的汽车，现在要从某一个加油站出发绕环路一周，一开始油箱为空。

求可环绕环路一周时出发的加油站的编号，若不存在环绕一周的方案，则返回`-1`。

注意事项

数据保证答案唯一。

样例

现在有 4 个加油站，汽油量 `gas[i]=[1, 1, 3, 1]`，环路旅行时消耗的汽油量 `cost[i]=[2, 2, 1, 1]`。则出发的加油站的编号为 2。

LintCode: http://www.lintcode.com/zh-cn/problem/gas-station/
LeetCode: https://leetcode.com/problems/gas-station/#/description

http://blog.csdn.net/nerv3x3/article/details/4258848
http://blog.csdn.net/u013291394/article/details/51264757

1.20 落单的数 II Single Number II

给出 3*n + 1 个的数字，除其中一个数字之外其他每个数字均出现三次，找到这个数字。

样例

给出 [1,1,2,3,3,3,2,2,4,1]，返回 4

LintCode: http://www.lintcode.com/zh-cn/problem/single-number-ii/
LeetCode: https://leetcode.com/problems/single-number-ii/#/description

http://blog.csdn.net/nerv3x3/article/details/39453407
http://blog.csdn.net/u013291394/article/details/51428529
http://blog.csdn.net/monkeyduck/article/details/39300147

难度：Hard

1.21 两个排序数组的中位数 Median of Two Sorted Arrays

两个排序的数组 A 和 B 分别含有 m 和 n 个数，找到两个排序数组的中位数，要求时间复杂度应为 O(log (m+n))。
样例
给出数组 A = [1,2,3,4,5,6] B = [2,3,4,5]，中位数 3.5

给出数组 A = [1,2,3] B = [4,5]，中位数 3
LintCode: http://www.lintcode.com/zh-cn/problem/median-of-two-sorted-arrays/
LeetCode: https://leetcode.com/problems/median-of-two-sorted-arrays/#/description

http://blog.csdn.net/nerv3x3/article/details/3465956
http://blog.csdn.net/u013291394/article/details/50286057

1.22 最长连续序列 Longest Consecutive Sequence

给定一个未排序的整数数组，找出最长连续序列的长度。
说明
要求你的算法复杂度为 O(n)
样例
给出数组[100, 4, 200, 1, 3, 2]，这个最长的连续序列是 [1, 2, 3, 4]，返回所求长度 4
LintCode: http://www.lintcode.com/zh-cn/problem/longest-consecutive-sequence/
LeetCode: https://leetcode.com/problems/longest-consecutive-sequence/#/description

http://blog.csdn.net/nerv3x3/article/details/37335577
http://blog.csdn.net/u013291394/article/details/51151500

1.23 接雨水 Trapping Rain Water

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.

样例

如上图所示，海拔分别为 `[0,1,0,2,1,0,1,3,2,1,2,1]`，返回 `6`.

LintCode: http://www.lintcode.com/zh-cn/problem/trapping-rain-water/

LeetCode: https://leetcode.com/problems/trapping-rain-water/#/description

http://blog.csdn.net/nerv3x3/article/details/37339357

http://blog.csdn.net/u013291394/article/details/50463200

1.24 分糖果 Candy

有 *N* 个小孩站成一列。每个小孩有一个评级。

按照以下要求，给小孩分糖果：

- 每个小孩至少得到一颗糖果。
- 评级越高的小孩可以得到更多的糖果。

需最少准备多少糖果？

样例

给定评级 = `[1, 2]`，返回 `3`.

给定评级 = `[1, 1, 1]`，返回 `3`.

给定评级 = `[1, 2, 2]`，返回 `4`. ([1,2,1]).

LintCode: http://www.lintcode.com/problem/candy/

LeetCode: https://leetcode.com/problems/candy/#/description

http://blog.csdn.net/nerv3x3/article/details/39453227

http://blog.csdn.net/u013291394/article/details/51277834

第二章 链表

难度：Easy

2.1 逆置链表 Reverse Linked List

一、
翻转一个链表

样例
给出一个链表 1->2->3->null，这个翻转后的链表为 3->2->1->null

LintCode:http://www.lintcode.com/zh-cn/problem/reverse-linked-list/
LeetCode:https://leetcode.com/problems/reverse-linked-list/#/description

http://blog.csdn.net/u013291394/article/details/50605007

二、
翻转链表中第 m 个节点到第 n 个节点的部分

注意事项
m，n 满足 1 ≤ m ≤ n ≤ 链表长度

样例
给出链表 1->2->3->4->5->null， m = 2 和 n = 4，返回 1->4->3->2->5->null

LintCode: http://www.lintcode.com/zh-cn/problem/reverse-linked-list-ii/
LeetCode https://leetcode.com/problems/reverse-linked-list-ii/#/description

http://blog.csdn.net/nerv3x3/article/details/4223089
http://blog.csdn.net/u013291394/article/details/50609288

2.2 删除排序链表中的重复元素 Remove Duplicates from Sorted List

给定一个排序链表，删除所有重复的元素每个元素只留下一个。

样例
给出 `1->1->2->null`，返回 `1->2->null`

给出 `1->1->2->3->3->null`，返回 `1->2->3->null`
LintCode: http://www.lintcode.com/zh-cn/problem/remove-duplicates-from-sorted-list/
LeetCode: https://leetcode.com/problems/remove-duplicates-from-sorted-list/#/description

http://blog.csdn.net/nerv3x3/article/details/3465647

http://blog.csdn.net/u013291394/article/details/50571452

2.3 带环链表 Linked List Cycle

给定一个链表，判断它是否有环。

样例
给出 -21->10->4->5, tail connects to node index 1，返回 true
LintCode: http://www.lintcode.com/zh-cn/problem/linked-list-cycle/
LeetCode: https://leetcode.com/problems/linked-list-cycle/#/description

http://blog.csdn.net/nerv3x3/article/details/3465710
http://blog.csdn.net/u013291394/article/details/51334790
http://blog.csdn.net/monkeyduck/article/details/38108715

难度：Medium

2.4 链表求和 Add Two Numbers

你有两个用链表代表的整数，其中每个节点包含一个数字。数字存储按照在原来整数中相反的顺序，使得第一个数字位于链表的开头。写出一个函数将两个整数相加，用链表形式返回和。

样例
给出两个链表 `3->1->5->null` 和 `5->9->2->null`，返回 `8->0->8->null`
LintCode: http://www.lintcode.com/zh-cn/problem/add-two-numbers/
LeetCode: https://leetcode.com/problems/add-two-numbers/#/description

http://blog.csdn.net/nerv3x3/article/details/35290083
http://blog.csdn.net/u013291394/article/details/50269537

2.5 链表划分 Partition List

给定一个单链表和数值 x，划分链表使得所有小于 x 的节点排在大于等于 x 的节点之前。

你应该保留两部分内链表节点原有的相对顺序。

样例
给定链表 1->4->3->2->5->2->null，并且 x=3

返回 1->2->2->4->3->5->null
LintCode: http://www.lintcode.com/zh-cn/problem/partition-list/
LeetCode: https://leetcode.com/problems/partition-list/#/description

http://blog.csdn.net/nerv3x3/article/details/39453367
http://blog.csdn.net/u013291394/article/details/50585339

2.6 删除排序链表中的重复元素 Remove Duplicates from Sorted List II

给定一个排序链表，删除所有重复的元素只留下原链表中没有重复的元素。

样例
给出 `1->2->3->3->4->4->5->null`，返回 `1->2->5->null`

给出 `1->1->1->2->3->null`，返回 `2->3->null`
LintCode: http://www.lintcode.com/zh-cn/problem/remove-duplicates-from-sorted-list-ii/
LeetCode: https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii/#/description

http://blog.csdn.net/nerv3x3/article/details/38929151
http://blog.csdn.net/u013291394/article/details/50582892

2.7 旋转链表 Rotate List

给定一个链表，旋转链表，使得每个节点向右移动 k 个位置，其中 k 是一个非负数

样例
给出链表 1->2->3->4->5->null 和 k=2

返回 4->5->1->2->3->null
LintCode: http://www.lintcode.com/zh-cn/problem/rotate-list/
LeetCode: https://leetcode.com/problems/rotate-list/#/description

http://blog.csdn.net/nerv3x3/article/details/38929137
http://blog.csdn.net/u013291394/article/details/50525595

2.8 删除链表中倒数第 n 个节点 Remove Nth Node From End of List

给定一个链表，删除链表中倒数第 n 个节点，返回链表的头节点。

注意事项
链表中的节点个数大于等于 n

样例
给出链表 1->2->3->4->5->null 和 n = 2.

删除倒数第二个节点之后，这个链表将变成 1->2->3->5->null.

LintCode: http://www.lintcode.com/zh-cn/problem/remove-nth-node-from-end-of-list/
LeetCode: https://leetcode.com/problems/remove-nth-node-from-end-of-list/#/description

http://blog.csdn.net/nerv3x3/article/details/36897851
http://blog.csdn.net/u013291394/article/details/50370258

2.9 两两交换链表中的节点 Swap Nodes in Pairs

给一个链表，两两交换其中的节点，然后返回交换后的链表。

样例
给出 `1->2->3->4`, 你应该返回的链表是 `2->1->4->3`。
LintCode: http://www.lintcode.com/zh-cn/problem/swap-nodes-in-pairs/
LeetCode: https://oj.leetcode.com/problems/swap-nodes-in-pairs/

http://blog.csdn.net/nerv3x3/article/details/3465506
http://blog.csdn.net/u013291394/article/details/50391999

2.10 复制带随机指针的链表 Copy List with Random Pointer

给出一个链表，每个节点包含一个额外增加的随机指针可以指向链表中的任何节点或空的节点。

返回一个深拷贝的链表。

LintCode: http://www.lintcode.com/zh-cn/problem/copy-list-with-random-pointer/
LeetCode: https://leetcode.com/problems/copy-list-with-random-pointer/#/description

http://blog.csdn.net/nerv3x3/article/details/39453281
http://blog.csdn.net/u013291394/article/details/51308051

2.11 带环链表 Linked List Cycle II

给定一个链表，如果链表中存在环，则返回到链表中环的起始节点的值，如果没有环，返回null。

样例
给出 -21->10->4->5, tail connects to node index 1，返回 10
LintCode: http://www.lintcode.com/zh-cn/problem/linked-list-cycle-ii/
LeetCode: https://leetcode.com/problems/linked-list-cycle-ii/#/description

http://blog.csdn.net/u013291394/article/details/51471133

2.12 重排链表 Reorder List

给定一个单链表 L: $L_0 \rightarrow L_1 \rightarrow \ldots \rightarrow L_{n-1} \rightarrow L_n$,

重新排列后为：$L_0 \rightarrow L_n \rightarrow L_1 \rightarrow L_{n-1} \rightarrow L_2 \rightarrow L_{n-2} \rightarrow \ldots$

必须在不改变节点值的情况下进行原地操作。

样例
给出链表 `1->2->3->4->null`，重新排列后为 `1->4->2->3->null`。
LintCode: http://www.lintcode.com/zh-cn/problem/reorder-list/
LeetCode: https://leetcode.com/problems/reorder-list/#/description

http://blog.csdn.net/nerv3x3/article/details/37337343
http://blog.csdn.net/u013291394/article/details/51356450

难度：Hard

2.13 K 组翻转链表 Reverse Nodes in k-Group

给你一个链表以及一个 *k*,将这个链表从头指针开始每 *k* 个翻转一下。
链表元素个数不是 *k* 的倍数，最后剩余的不用翻转。

样例
给出链表 `1->2->3->4->5`

k = `2`，返回 `2->1->4->3->5`

k = `3`，返回 `3->2->1->4->5`
LintCode: http://www.lintcode.com/zh-cn/problem/reverse-nodes-in-k-group/
LeetCode: https://leetcode.com/problems/reverse-nodes-in-k-group/#/description

http://blog.csdn.net/u013291394/article/details/50401210

2.14 LRU 缓存策略 LRU Cache

为最近最少使用（LRU）缓存策略设计一个数据结构，它应该支持以下操作：获取数据（get）和写入数据（set）。

获取数据 get(key)：如果缓存中存在 key，则获取其数据值（通常是正数），否则返回-1。

写入数据 set(key, value)：如果 key 还没有在缓存中，则写入其数据值。当缓存达到上限，它应该在写入新数据之前删除最近最少使用的数据用来腾出空闲位置。

LeetCode: https://leetcode.com/problems/lru-cache/#/description
LintCode: http://www.lintcode.com/zh-cn/problem/lru-cache/

http://blog.csdn.net/nerv3x3/article/details/2920168
http://blog.csdn.net/u013291394/article/details/51447794

第三章 字符串

难度：Easy

3.1 有效回文串 Valid Palindrome

给定一个字符串，判断其是否为一个回文串。只包含字母和数字，忽略大小写。

注意事项

你是否考虑过，字符串有可能是空字符串？这是面试过程中，面试官常常会问的问题。

在这个题目中，我们将空字符串判定为有效回文。

样例

`"A man, a plan, a canal: Panama"` 是一个回文。

`"race a car"` 不是一个回文。

LintCode: http://www.lintcode.com/zh-cn/problem/valid-palindrome/
LeetCode: https://leetcode.com/problems/valid-palindrome/#/description

http://blog.csdn.net/nerv3x3/article/details/4216759
http://blog.csdn.net/u013291394/article/details/51112773

3.2 字符串查找 Implement strStr()

对于一个给定的 source 字符串和一个 target 字符串，你应该在 source 字符串中找出 target 字符串出现的第一个位置(从 0 开始)。如果不存在，则返回 `-1`。

说明

在面试中我是否需要实现 KMP 算法？

- 不需要，当这种问题出现在面试中时，面试官很可能只是想要测试一下你的基础应用能力。当然你需要先跟面试官确认清楚要怎么实现这个题。

样例

如果 source = `"source"` 和 target = `"target"`，返回 `-1`。

如果 source = `"abcdabcdefg"` 和 target = `"bcd"`，返回 `1`。

LintCode: http://www.lintcode.com/zh-cn/problem/strstr/

LeetCode: https://leetcode.com/problems/implement-strstr/#/description

http://blog.csdn.net/u013291394/article/details/50418809

3.3 二进制求和 Add Binary

给定两个二进制字符串，返回他们的和（用二进制表示）。

样例
a = `"11"`
b = `"1"`
Return `"100"`
LintCode: http://www.lintcode.com/zh-cn/problem/add-binary/
LeetCode: https://leetcode.com/problems/add-binary/#/description

http://blog.csdn.net/nerv3x3/article/details/4137815
http://blog.csdn.net/u013291394/article/details/50540383

3.4 最长公共前缀 Longest Common Prefix

给 k 个字符串，求出他们的最长公共前缀(LCP)
样例
在 "ABCD" "ABEF" 和 "ACEF" 中, LCP 为 "A"

在 "ABCDEFG", "ABCEFG", "ABCEFA" 中, LCP 为 "ABC"
LintCode: http://www.lintcode.com/zh-cn/problem/longest-common-prefix/
LeetCode: https://leetcode.com/problems/longest-common-prefix/#/description

http://blog.csdn.net/nerv3x3/article/details/37335391
http://blog.csdn.net/u013291394/article/details/50354183
https://github.com/williamwhe/leetcode/blob/master/longest-common-prefix.py

3.5 罗马数字转整数 Roman to Integer

给定一个罗马数字，将其转换成整数。

返回的结果要求在 1 到 3999 的范围内。

说明
什么是 *罗马数字*?

- https://en.wikipedia.org/wiki/Roman_numerals
- https://zh.wikipedia.org/wiki/%E7%BD%97%E9%A9%AC%E6%95%B0%E5%AD%97

- http://baike.baidu.com/view/42061.htm

样例
`IV` -> `4`

`XII` -> `12`

`XXI` -> `21`

`XCIX` -> `99`

LintCode: http://www.lintcode.com/zh-cn/problem/roman-to-integer/
LeetCode: https://leetcode.com/problems/roman-to-integer/#/description

http://blog.csdn.net/nerv3x3/article/details/37337795
http://blog.csdn.net/u013291394/article/details/50350386

3.6 报数 Count and Say

报数指的是，按照其中的整数的顺序进行报数，然后得到下一个数。如下所示：

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

`1` 读作 `"one 1"` -> `11`.

`11` 读作 `"two 1s"` -> `21`.

`21` 读作 `"one 2, then one 1"` -> `1211`.

给定一个整数 `n`, 返回 第 `n` 个顺序。

注意事项
整数的顺序将表示为一个字符串。

样例
给定 n = `5`, 返回 `"111221"`.

LintCode: http://www.lintcode.com/zh-cn/problem/count-and-say/
LeetCode: https://leetcode.com/problems/count-and-say/#/description

http://blog.csdn.net/nerv3x3/article/details/39453287
http://blog.csdn.net/u013291394/article/details/50416817
https://github.com/williamwhe/leetcode/blob/master/count-and-say.py

3.7 最后一个单词的长度 Length of Last Word

给定一个字符串， 包含大小写字母、空格`' '`，请返回其最后一个单词的长度。

如果不存在最后一个单词，请返回 `0` 。

注意事项

一个单词的界定是，由字母组成，但不包含任何的空格。

样例

给定 s = `"Hello World"`，返回 `5`。

LintCode: http://www.lintcode.com/zh-cn/problem/length-of-last-word/
LeetCode: https://leetcode.com/problems/length-of-last-word/#/description

http://blog.csdn.net/nerv3x3/article/details/3465976
http://blog.csdn.net/u013291394/article/details/50519919

难度：Medium

3.8 字符串转化成整型 String to Integer (atoi)

讲一个字符串转化成 int 类型。题目非常简单，但要额外考虑的因素非常多，如下面的一些字符串的处理：”+123”, ” -23 “, “231ji2”, null, ” “等等。

注意点：

- 字符串可能为空
- 字符串可能全是空格，或前后有空格
- 要考虑正负号
- 要考虑对于 32 位整数是否溢出
- 要考虑不是数字的字符，如果首字母不是数字，返回为 0，在字符串中的将它及它之后的字符全部忽略
- 异常情况全部返回 0

例子：

输入: str=” +123”
输出: 123

输入: str=”-123fe2”
输出: -123

LeetCode: https://leetcode.com/problems/string-to-integer-atoi/#/description

http://blog.csdn.net/nerv3x3/article/details/3465509
http://blog.csdn.net/u013291394/article/details/50325341

3.9 最长回文子串 Longest Palindromic Substring

给出一个字符串（假设长度最长为 1000），求出它的最长回文子串，你可以假定只有一个满足条件的最长回文串。

样例

给出字符串 `"abcdzdcab"`，它的最长回文子串为 `"cdzdc"`。

LintCode: http://www.lintcode.com/zh-cn/problem/longest-palindromic-substring/

LeetCode: https://leetcode.com/problems/longest-palindromic-substring/#/description

http://blog.csdn.net/u013291394/article/details/50300661

https://github.com/williamwhe/leetcode/blob/master/longest-palind.py

3.10 整数转罗马数字 Integer to Roman

给定一个整数，将其转换成罗马数字。

返回的结果要求在 1-3999 的范围内。

说明

什么是 *罗马数字*?

- https://en.wikipedia.org/wiki/Roman_numerals
- https://zh.wikipedia.org/wiki/%E7%BD%97%E9%A9%AC%E6%95%B0%E5%AD%97
- http://baike.baidu.com/view/42061.htm

样例

`4` -> `IV`

`12` -> `XII`

`21` -> `XXI`

`99` -> `XCIX`

更多案例，请戳 http://literacy.kent.edu/Minigrants/Cinci/romanchart.htm

LintCode: http://www.lintcode.com/zh-cn/problem/integer-to-roman/

LeetCode: https://leetcode.com/problems/integer-to-roman/#/description

http://blog.csdn.net/nerv3x3/article/details/37334871

http://blog.csdn.net/u013291394/article/details/50349868

3.11 简化路径 Simplify Path

给定一个文档(Unix-style)的完全路径，请进行路径简化。

样例

`"/home/"`, => `"/home"`

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

LintCode: http://www.lintcode.com/zh-cn/problem/simplify-path/

LeetCode: https://leetcode.com/problems/simplify-path/#/description

http://blog.csdn.net/nerv3x3/article/details/39453399

http://blog.csdn.net/u013291394/article/details/50548001

难度：Hard

3.12 正则表达式匹配 Regular Expression Matching

实现支持'.'和'*'的正则表达式匹配。

'.'匹配任意一个字母。

'*'匹配零个或者多个前面的元素。

匹配应该覆盖整个输入字符串，而不仅仅是一部分。

需要实现的函数是：bool isMatch(const char *s, const char *p)

样例

isMatch("aa","a") → false

isMatch("aa","aa") → true

isMatch("aaa","aa") → false

isMatch("aa", "a*") → true

isMatch("aa", ".*") → true

isMatch("ab", ".*") → true

isMatch("aab", "c*a*b") → true

LintCode: http://www.lintcode.com/zh-cn/problem/regular-expression-matching/

LeetCode: https://leetcode.com/problems/regular-expression-matching/#/description

http://blog.csdn.net/nerv3x3/article/details/2920237

http://blog.csdn.net/u013291394/article/details/50346505

3.13 通配符匹配 Wildcard Matching

判断两个可能包含通配符“？”和“*”的字符串是否匹配。匹配规则如下：

'?' 可以匹配任何单个字符。

'*' 可以匹配任意字符串（包括空字符串）。

两个串完全匹配才算匹配成功。

函数接口如下：

```
bool isMatch(const char *s, const char *p)
```

样例

一些例子：

```
isMatch("aa","a") → false

isMatch("aa","aa") → true

isMatch("aaa","aa") → false

isMatch("aa", "*") → true

isMatch("aa", "a*") → true

isMatch("ab", "?*") → true

isMatch("aab", "c*a*b") → false
```

LintCode: http://www.lintcode.com/zh-cn/problem/wildcard-matching/
LeetCode: https://leetcode.com/problems/wildcard-matching/#/description

http://blog.csdn.net/nerv3x3/article/details/2921852
http://blog.csdn.net/u013291394/article/details/50563953
https://github.com/williamwhe/leetcode/blob/master/wildcard.py

3.14 有效数字 Valid Number

给定一个字符串，验证其是否为数字。

样例

`"0"` => `true`

`" 0.1 "` => `true`

`"abc"` => `false`

`"1 a"` => `false`

`"2e10"` => `true`

LintCode: http://www.lintcode.com/zh-cn/problem/valid-number/
LeetCode: https://leetcode.com/problems/valid-number/#/description

http://blog.csdn.net/nerv3x3/article/details/41628867
http://blog.csdn.net/u013291394/article/details/50563950

第四章 栈和队列

难度：Easy

4.1 有效的括号序列 Valid Parentheses

给定一个字符串所表示的括号序列，包含以下字符： `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`， 判定是否是有效的括号序列。

样例

括号必须依照 `"()"` 顺序表示， `"()[]{}"` 是有效的括号，但 `"([)]"`则是无效的括号。

LintCode: http://www.lintcode.com/zh-cn/problem/valid-parentheses/

LeetCode: https://leetcode.com/problems/valid-parentheses/#/description

http://blog.csdn.net/nerv3x3/article/details/36896909
http://blog.csdn.net/u013291394/article/details/50374281
https://github.com/williamwhe/leetcode/blob/master/valid-parentheses.py

难度：Medium

4.2 逆波兰表达式求值 Evaluate Reverse Polish Notation

求逆波兰表达式的值。

在逆波兰表达法中，其有效的运算符号包括 `+`, `-`, `*`, `/` 。每个运算对象可以是整数，也可以是另一个逆波兰计数表达。

样例

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

LeetCode: https://oj.leetcode.com/problems/evaluate-reverse-polish-notation/

http://blog.csdn.net/nerv3x3/article/details/3465981
http://blog.csdn.net/u013291394/article/details/51439660

4.3 简化路径 Simplify Path

给定一个文档(Unix-style)的完全路径，请进行路径简化。
样例
`"/home/"`, => `"/home"`

`"/a/./b/../../c/"`, => `"/c"`
LintCode: http://www.lintcode.com/zh-cn/problem/simplify-path/
LeetCode: https://leetcode.com/problems/simplify-path/#/description

http://blog.csdn.net/nerv3x3/article/details/39453399
http://blog.csdn.net/u013291394/article/details/50548001

难度：Hard

4.4 求最长有效匹配括号子串的长度 Longest Valid Parentheses

找出一个只包含"("和")"的字符串中最长的有效子字符串的长度。有效的意思是指该子字符串中的括号都能正确匹配。

注意点：注意空字符串

样例：
输入: s = "(()"
输出: 2

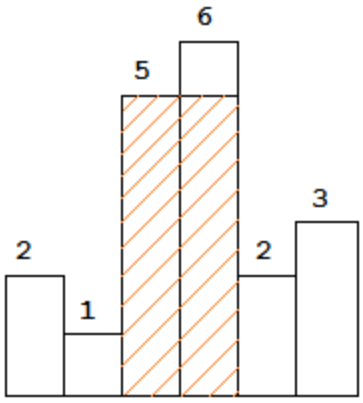
输入: s = ")()())"
输出: 4

LeetCode: https://leetcode.com/problems/longest-valid-parentheses/#/description

http://blog.csdn.net/u013291394/article/details/50436937
https://github.com/williamwhe/leetcode/blob/master/longest-palindrome.py

4.5 直方图最大矩形覆盖 Largest Rectangle in Histogram

给定一个柱状图，求它能包含的最大的矩形的面积。如下图中阴影部分就是要求的矩形。



注意点：

所有的柱的宽度都为 1

例子：

输入: heights = [2,1,5,6,2,3]

输出: 10

样例
给出 height = [2,1,5,6,2,3]，返回 10
LintCode: http://www.lintcode.com/zh-cn/problem/largest-rectangle-in-histogram/
LeetCode: https://leetcode.com/problems/largest-rectangle-in-histogram/#/description

http://blog.csdn.net/u013291394/article/details/50850225

第五章 树

5.1 二叉树的遍历

难度：Easy

5.1.1 二叉树的层次遍历 Binary Tree Level Order Traversal II

给出一棵二叉树，返回其节点值从底向上的层次序遍历（按从叶节点所在层到根节点所在的层遍历，然后逐层从左往右遍历）

样例

给出一棵二叉树 `{3,9,20,#,#,15,7}`,

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

按照从下往上的层次遍历为：

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

LintCode: http://www.lintcode.com/zh-cn/problem/binary-tree-level-order-traversal-ii/
LeetCode: https://leetcode.com/problems/binary-tree-level-order-traversal-ii/

http://blog.csdn.net/nerv3x3/article/details/37329485
http://blog.csdn.net/u013291394/article/details/50683731
https://github.com/williamwhe/leetcode/blob/master/binary-tree-level-order-traversal-ii.py

5.1.2 比较两个二叉树是否相等 Same Tree

判断两棵二叉树是否相等。两棵二叉树仅在它们的形状相同且每个节点的值相等时才判为相等。

例子：

输入:

```
     2          2
p =  / \   q =  / \
    1   3      1   3
```

输出: True

LeetCode: https://leetcode.com/problems/same-tree/#/description

http://blog.csdn.net/nerv3x3/article/details/3465524
http://blog.csdn.net/u013291394/article/details/50625470
http://blog.csdn.net/monkeyduck/article/details/37995813

5.1.3 对称二叉树 Symmetric Tree

判断一棵树是否是镜面对称的。最好同时提供递归和迭代的解法。

注意点：

无

例子：

输入:

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

输出: True

输入:

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

输出: False

LeetCode: https://oj.leetcode.com/problems/symmetric-tree/

http://blog.csdn.net/nerv3x3/article/details/37339069
http://blog.csdn.net/u013291394/article/details/50625474

5.1.4 平衡二叉树 Balanced Binary Tree

给定一个二叉树,确定它是高度平衡的。对于这个问题,一棵高度平衡的二叉树的定义是：一棵二叉树中每个节点的两个子树的深度相差不会超过 1。

样例

给出二叉树 A=`{3,9,20,#,#,15,7}`, B=`{3,#,20,15,7}`

```
A)  3            B)    3
   / \                  \
  9  20                  20
    /  \                 / \
   15   7               15  7
```

二叉树 A 是高度平衡的二叉树，但是 B 不是

LintCode: http://www.lintcode.com/zh-cn/problem/balanced-binary-tree/
LeetCode: https://oj.leetcode.com/problems/balanced-binary-tree/

http://blog.csdn.net/nerv3x3/article/details/3465764
http://blog.csdn.net/u013291394/article/details/50707267

难度：Medium

5.1.5 二叉树的前序遍历 Binary Tree Preorder Traversal

给出一棵二叉树，返回其节点值的前序遍历。

样例

给出一棵二叉树 `{1,#,2,3}`,

```
1
 \
  2
 /
3
```

返回 `[1,2,3]`.

LintCode: http://www.lintcode.com/zh-cn/problem/binary-tree-preorder-traversal/

LeetCode: https://leetcode.com/problems/binary-tree-preorder-traversal/#/description

http://blog.csdn.net/nerv3x3/article/details/3465737
http://blog.csdn.net/u013291394/article/details/51387852

5.1.6 二叉树的中序遍历 Binary Tree Inorder Traversal

给出一棵二叉树,返回其中序遍历

样例

给出二叉树 `{1,#,2,3}`,

```
  1
   \
    2
   /
  3
```

返回 `[1,3,2]`.
LintCode: http://www.lintcode.com/zh-cn/problem/binary-tree-inorder-traversal/
LeetCode: https://leetcode.com/problems/binary-tree-inorder-traversal/#/description

http://blog.csdn.net/nerv3x3/article/details/3465748
http://blog.csdn.net/u013291394/article/details/50612207
http://blog.csdn.net/monkeyduck/article/details/38108961

5.1.7 二叉树的层次遍历 Binary Tree Level Order Traversal

给出一棵二叉树，返回其节点值的层次遍历（逐层从左往右访问）

样例

给一棵二叉树 `{3,9,20,#,#,15,7}`：

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

返回他的分层遍历结果：

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

LintCode: http://www.lintcode.com/zh-cn/problem/binary-tree-level-order-traversal/
LeetCode: https://leetcode.com/problems/binary-tree-level-order-traversal/

http://blog.csdn.net/nerv3x3/article/details/37329065
http://blog.csdn.net/u013291394/article/details/50631025

5.1.8 二叉树的锯齿形层次遍历 Binary Tree Zigzag Level Order Traversal

给出一棵二叉树，返回其节点值的锯齿形层次遍历（先从左往右，下一层再从右往左，层与层之间交替进行）

样例

给出一棵二叉树 `{3,9,20,#,#,15,7}`,

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

返回其锯齿形的层次遍历为：

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

LintCode: http://www.lintcode.com/zh-cn/problem/binary-tree-zigzag-level-order-traversal/
LeetCode: https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/

http://blog.csdn.net/u013291394/article/details/50631029

5.1.9 二叉树转单链表 Flatten Binary Tree to Linked List

把一棵二叉树变为链表，也就是一棵所有节点要么没有子节点，要么只有右节点的二叉树。
注意点：
无
例子:
输入:

```
        1
       / \
      2   5
     / \   \
    3   4   6
```

输出:

```
1
 \
  2
   \
    3
     \
      4
       \
        5
         \
          6
```

LeetCode: https://leetcode.com/problems/flatten-binary-tree-to-linked-list/#/description

http://blog.csdn.net/nerv3x3/article/details/39453299
http://blog.csdn.net/u013291394/article/details/50799703

5.1.10 任意（非完美）二叉树添加 next 指针 Populating Next Right Pointers in Each Node II

为二叉树的节点都添加一个 next 指针，指向跟它在同一高度的右边的节点，如果右边没有节点，就指向 None。与 Populating Next Right Pointers in Each Node 的区别就是这里的二叉树可以是不完全二叉树。
注意点：
最好只用常量的空间
例子：
输入:

```
        1
      /  \
     2    3
    / \    \
   4   5    7
```

输出:

```
        1 -> NULL
      /  \
     2 -> 3 -> NULL
    / \    \
   4-> 5 -> 7 -> NULL
```

LeetCode:https://leetcode.com/problems/populating-next-right-pointers-in-each-node-ii/#/description

http://blog.csdn.net/nerv3x3/article/details/36897643
http://blog.csdn.net/u013291394/article/details/50904785

难度：Hard

5.1.11 二叉树的后序遍历 Binary Tree Postorder Traversal

给出一棵二叉树，返回其节点值的后序遍历。

样例

给出一棵二叉树 `{1,#,2,3}`,

```
  1
   \
    2
   /
  3
```

返回 `[3,2,1]`
LintCode: http://www.lintcode.com/zh-cn/problem/binary-tree-postorder-traversal/
LeetCode: https://leetcode.com/problems/binary-tree-postorder-traversal/#/description

http://blog.csdn.net/u013291394/article/details/51387862

5.1.12 复原二叉搜索树 Recover Binary Search Tree

一棵二叉搜索树中的两个节点交换了位置，找出并调整。

注意点：

- 最好只用常量的空间

例子：

输入:

```
  3
 / \
1   2
```

输出:

```
  2
 / \
1   3
```

LeetCode: https://leetcode.com/problems/recover-binary-search-tree/#/description

http://blog.csdn.net/u013291394/article/details/50883736
https://github.com/williamwhe/leetcode/blob/master/rovertree.py

5.2 二叉树的构建

难度：Medium

5.2.1 通过前序、中序遍历构建二叉树 Construct Binary Tree from Preorder and Inorder Traversal

通过一棵二叉树的前序和中序排列来得出它的树形结构。
例子:
输入: preorder = [3,9,20,15,14,7], inorder = [9,3,14,15,20,7]
输出:

```
    3
   / \
  9  20
    /  \
   15   7
  /
 14
```

LeetCode:https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/

http://blog.csdn.net/u013291394/article/details/50678202

5.2.2 通过中序、后序遍历构建二叉树 Construct Binary Tree from Inorder and Postorder Traversal

通过一棵二叉树的中序和后序排列来得出它的树形结构。
注意点：
无

例子:
输入: inorder = [9,3,14,15,20,7], postorder = [9,14,15,7,20,3]
输出:

```
    3
   / \
  9  20
    /  \
   15   7
  /
 14
```

LeetCode:https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/

http://blog.csdn.net/u013291394/article/details/50688456

5.3 二叉查找树

难度：Easy

5.3.1 将有序数组转化为二叉搜索树 Convert Sorted Array to Binary Search Tree

给定一个升序的序列，将它转化为高度平衡的二叉搜索树。

注意点：

同一个序列转化成的二叉搜索树可能有多种

例子:

输入: nums = [1,2,3]

输出:

```
  2
```

```
 / \
1   3
```

LeetCode:https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/#/description

http://blog.csdn.net/nerv3x3/article/details/39453279
http://blog.csdn.net/u013291394/article/details/50697682

难度：Medium

5.3.2 唯一二叉搜索树 Unique Binary Search Trees

给定 1 到 n 这 n 个数，用它们能够构成多少种形状不同的二叉搜索树。
注意点：
这 n 个数都要是二叉搜索树的节点，不能只取部分
例子：
输入: n = 3
输出: 5

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

LeetCode: https://oj.leetcode.com/problems/unique-binary-search-trees/

http://blog.csdn.net/nerv3x3/article/details/37339649
http://blog.csdn.net/u013291394/article/details/50615027
http://blog.csdn.net/monkeyduck/article/details/38079861

5.3.3 唯一二叉搜索树 Unique Binary Search Trees II

给定 1 到 n 这 n 个数，用它们能够构成多少种形状不同的二叉搜索树。将所有的二叉搜索树罗列出来。

注意点：

- 这 n 个数都要是二叉搜索树的节点，不能只取部分

例子：

输入: n = 3

输出:

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

LeetCode: https://leetcode.com/problems/unique-binary-search-trees-ii/

http://blog.csdn.net/u013291394/article/details/50620479

5.3.4 验证二叉搜索树 Validate Binary Search Tree

判断一棵二叉搜索树是否有效。有效是指每个节点的值大于左节点，小于右节点（如果有对应节点的话），且它的左节点和右节点也满足这种条件。

例子：

输入:

```
  2
 / \
1   3
```

输出: True

LeetCode:https://leetcode.com/problems/validate-binary-search-tree/

http://blog.csdn.net/nerv3x3/article/details/3465507
http://blog.csdn.net/u013291394/article/details/50616870

5.4 二叉树的递归

难度：Easy

5.4.1 二叉树最小深度 Minimum Depth of Binary Tree

求一棵二叉树的最小高度，即从根节点到最近叶子节点的路径经过的节点数。

例子:

输入:

```
    3
   / \
  9  20
    /  \
   15   7
  /
 14
```

输出: 2

LeetCode: https://leetcode.com/problems/minimum-depth-of-binary-tree/#/description

http://blog.csdn.net/nerv3x3/article/details/3465902
http://blog.csdn.net/u013291394/article/details/50720623

5.4.2 二叉树最大深度 Maximum Depth of Binary Tree

求一颗二叉树的最大深度，最大深度指跟节点到最底层叶子节点的距离。

例子:

输入:

```
    3
   / \
  9  20
    /  \
   15   7
  /
 14
```

LeetCode: https://leetcode.com/problems/maximum-depth-of-binary-tree/#/description

http://blog.csdn.net/nerv3x3/article/details/3465897
http://blog.csdn.net/u013291394/article/details/50639059
http://blog.csdn.net/monkeyduck/article/details/37991201

5.4.3 求路径和 Path Sum

判断一棵二叉树是否有一条从根节点到某一叶子节点的路径，该路径上所有节点的和为一个特定值。

例子:

输入: sum = 12

```
    3
   / \
  9  20
    /  \
   15   7
  /
 14
```

输出: True (3->9)

LeetCode:https://leetcode.com/problems/path-sum/#/description

http://blog.csdn.net/nerv3x3/article/details/35289105
http://blog.csdn.net/u013291394/article/details/50728086

https://github.com/williamwhe/leetcode/blob/master/path-sum.py

难度：Medium

5.4.4 求路径和 Path Sum II

判断一棵二叉树是否有一条从根节点到某一叶子节点的路径，该路径上所有节点的和为一个特定值。

例子:

输入: sum = 12

```
   3
  / \
 9  20
   /  \
  15   7
 /
14
```

输出: True (3->9)

LeetCode: https://oj.leetcode.com/problems/path-sum-ii/

http://blog.csdn.net/nerv3x3/article/details/35289569
http://blog.csdn.net/u013291394/article/details/50740194

5.4.5 完美二叉树添加 next 指针 Populating Next Right Pointers in Each Node

为二叉树的节点都添加一个 next 指针，指向跟它在同一高度的右边的节点，如果右边没有节点，就指向 None。

注意点：

- 最好只用常量的空间
- 这是一棵完全二叉树

例子：

输入：

```
        1
      /  \
     2    3
    / \  / \
   4  5  6  7
```

输出：

```
        1 -> NULL
      /  \
     2 -> 3 -> NULL
    / \  / \
   4->5->6->7 -> NULL
```

LeetCode:https://oj.leetcode.com/problems/populating-next-right-pointers-in-each-node/

http://blog.csdn.net/nerv3x3/article/details/36897435
http://blog.csdn.net/u013291394/article/details/50894314

5.4.6 求所有从根节点到叶子节点组成的数字的和 Sum Root to Leaf Numbers

一棵树的每个节点都是 0-9 中的某一个数字，现在把从根节点到某一个叶子节点之间所有节点的数字依次连接起来组成一个新的数字。要求所有从根节点到叶子节点组成的数字的和。

例子：

输入：

```
   1
  / \
 2   3
```

输出: 25

LeetCode:https://leetcode.com/problems/sum-root-to-leaf-numbers/#/description

http://blog.csdn.net/nerv3x3/article/details/37338753
http://blog.csdn.net/u013291394/article/details/51193356

难度：Hard

5.4.7 二叉树的最大路径和 Binary Tree Maximum Path Sum

求一棵二叉树中最大的路径和。该路径可以是二叉树中某一节点到树中任意一个节点的所经过的路径，不允许重复经过一个节点，不必经过根节点。

例子:

输入:

```
    1
   / \
  2   3
```

输出: 6

LeetCode:https://oj.leetcode.com/problems/binary-tree-maximum-path-sum/

http://blog.csdn.net/nerv3x3/article/details/31487859
http://blog.csdn.net/u013291394/article/details/51366022

第六章 排序

难度：Easy

6.1 合并排序数组 Merge Sorted Array

合并两个排序的整数数组 A 和 B 变成一个新的数组。

注意事项

你可以假设 A 具有足够的空间（A 数组的大小大于或等于 m+n）去添加 B 中的元素。

样例

给出 A = `[1, 2, 3, empty, empty]`, B = `[4, 5]`

合并之后 A 将变成 `[1,2,3,4,5]`

LintCode: http://www.lintcode.com/zh-cn/problem/merge-sorted-array/
LeetCode: https://leetcode.com/problems/merge-sorted-array/#/description

http://blog.csdn.net/nerv3x3/article/details/3465682
http://blog.csdn.net/u013291394/article/details/50589862
https://github.com/williamwhe/leetcode/blob/master/merge-sorted-array.py

6.2 合并两个排序链表 Merge Two Sorted Lists

将两个排序链表合并为一个新的排序链表

样例

给出 `1->3->8->11->15->null`， `2->null`， 返回 `1->2->3->8->11->15->null`。

LintCode: http://www.lintcode.com/zh-cn/problem/merge-two-sorted-lists/
LeetCode: https://oj.leetcode.com/problems/merge-two-sorted-lists/

http://blog.csdn.net/nerv3x3/article/details/3465680
http://blog.csdn.net/u013291394/article/details/50378014

难度：Medium

6.3 链表插入排序 Insertion Sort List

用插入排序对链表排序

样例

Given `1->3->2->0->null`, return `0->1->2->3->null`

LintCode: http://www.lintcode.com/zh-cn/problem/insertion-sort-list/
LeetCode: https://leetcode.com/problems/Insertion-Sort-List/#/description

http://blog.csdn.net/nerv3x3/article/details/3465729
http://blog.csdn.net/u013291394/article/details/51457362

6.4 链表排序 Sort List

在 O(n log n) 时间复杂度和常数级的空间复杂度下给链表排序。

样例

给出 `1->3->2->null`，给它排序变成 `1->2->3->null`.

LintCode: http://www.lintcode.com/zh-cn/problem/sort-list/
LeetCode: https://leetcode.com/problems/sort-list/#/description

http://blog.csdn.net/nerv3x3/article/details/38143633

6.5 颜色分类 Sort Colors

给定一个包含红，白，蓝且长度为 n 的数组，将数组元素进行分类使相同颜色的元素相邻，并按照红、白、蓝的顺序进行排序。

我们可以使用整数 0，1 和 2 分别代表红，白，蓝。

注意事项

不能使用代码库中的排序函数来解决这个问题。
排序需要在原数组中进行。

样例
给你数组 `[1, 0, 1, 2]`，需要将该数组原地排序为 `[0, 1, 1, 2]`。

LintCode: http://www.lintcode.com/zh-cn/problem/sort-colors/
LeetCode: https://oj.leetcode.com/problems/sort-colors/

http://blog.csdn.net/nerv3x3/article/details/39453411
http://blog.csdn.net/u013291394/article/details/50557290
http://blog.csdn.net/monkeyduck/article/details/39209429

难度：Hard

6.6 合并 k 个排序链表 Merge k Sorted Lists

合并 k 个排序链表，并且返回合并后的排序链表。尝试分析和描述其复杂度。
样例
给出 3 个排序链表[2->4->null,null,-1->null]，返回 -1->2->4->null
LintCode: http://www.lintcode.com/zh-cn/problem/merge-k-sorted-lists/
LeetCode: https://leetcode.com/problems/merge-k-sorted-lists/#/description

http://blog.csdn.net/u013291394/article/details/50391995

6.7 丢失的第一个正整数 First Missing Positive

给出一个无序的正数数组，找出其中没有出现的最小正整数。
样例

如果给出 `[1,2,0]`, return `3`
如果给出 `[3,4,-1,1]`, return `2`
LintCode: http://www.lintcode.com/zh-cn/problem/first-missing-positive/
LeetCode: https://leetcode.com/problems/first-missing-positive/#/description

http://blog.csdn.net/u013291394/article/details/50460294

第七章 查找

难度：Easy

7.1 搜索插入位置 Search Insert Position

给定一个排序数组和一个目标值，如果在数组中找到目标值则返回索引。如果没有，返回到它将会被按顺序插入的位置。

你可以假设在数组中无重复元素。
样例
[1,3,5,6]，5 → 2

[1,3,5,6]，2 → 1

[1,3,5,6]， 7 → 4

[1,3,5,6]，0 → 0
LintCode: http://www.lintcode.com/zh-cn/problem/search-insert-position/
LeetCode: https://leetcode.com/problems/search-insert-position/#/description

http://blog.csdn.net/nerv3x3/article/details/39453389
http://blog.csdn.net/u013291394/article/details/50449119

难度：Medium

7.2 搜索区间 Search for a Range

给定一个包含 *n* 个整数的排序数组，找出给定目标值 *target* 的起始和结束位置。

如果目标值不在数组中，则返回`[-1, -1]`
样例
给出`[5, 7, 7, 8, 8, 10]`和目标值 target=`8`,

返回`[3, 4]`
LintCode: http://www.lintcode.com/zh-cn/problem/search-for-a-range/

LeetCode: https://leetcode.com/problems/search-for-a-range/#/description

http://blog.csdn.net/u013291394/article/details/50447580

7.3 搜索二维矩阵 Search a 2D Matrix

写出一个高效的算法来搜索 $m \times n$ 矩阵中的值。

这个矩阵具有以下特性：

- 每行中的整数从左到右是排序的。
- 每行的第一个数大于上一行的最后一个整数。

样例
考虑下列矩阵：

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

给出 `target = 3`，返回 `true`
LintCode: http://www.lintcode.com/zh-cn/problem/search-a-2d-matrix/
LeetCode: https://leetcode.com/problems/search-a-2d-matrix/

http://blog.csdn.net/u013291394/article/details/50557303

第八章 广度优先搜索

难度：Medium

8.1 单词接龙 Word Ladder

给出两个单词（start 和 end）和一个字典，找到从 start 到 end 的最短转换序列

比如：

每次只能改变一个字母。
变换过程中的中间单词必须在字典中出现。
注意事项：

如果没有转换序列则返回 0。
所有单词具有相同的长度。
所有单词都只包含小写字母。
样例
给出数据如下：

start = "hit"

end = "cog"

dict = ["hot","dot","dog","lot","log"]

一个最短的变换序列是 "hit" -> "hot" -> "dot" -> "dog" -> "cog",

返回它的长度 5

LintCode: http://www.lintcode.com/zh-cn/problem/word-ladder/
LeetCode: https://leetcode.com/problems/word-ladder/

http://blog.csdn.net/u013291394/article/details/51326575

8.2 被围绕的区域 Surrounded Regions

给一个二维的矩阵，包含 `'X'` 和 `'O'`，找到所有被 'X' 围绕的区域，并用 'X' 填充满。
样例
给出二维矩阵：

```
X X X X
X O O X
X X O X
X O X X
```

把被 'X' 围绕的区域填充之后变为：

```
X X X X
X X X X
X X X X
X O X X
```

LintCode: http://www.lintcode.com/zh-cn/problem/surrounded-regions/
LeetCode: https://leetcode.com/problems/surrounded-regions/#/

http://blog.csdn.net/u013291394/article/details/51200609

难度：Hard

8.3 单词接龙 Word Ladder II

给出两个单词（start 和 end）和一个字典，找出所有从 start 到 end 的最短转换序列

比如：

1.每次只能改变一个字母。
2.变换过程中的中间单词必须在字典中出现。
注意事项

- 所有单词具有相同的长度。
- 所有单词都只包含小写字母。

样例
给出数据如下：

start = "hit"

end = "cog"

dict = ["hot","dot","dog","lot","log"]

返回

```
[
    ["hit","hot","dot","dog","cog"],
    ["hit","hot","lot","log","cog"]
  ]
```

LintCode: http://www.lintcode.com/zh-cn/problem/word-ladder-ii/
LeetCode: https://leetcode.com/problems/word-ladder-ii/#/description

http://blog.csdn.net/u013291394/article/details/51464671

第九章 深度优先搜索

难度：Medium

9.1 分割回文串 Palindrome Partitioning

给定一个字符串 s，将 s 分割成一些子串，使每个子串都是回文串。

返回 s 所有可能的回文串分割方案。

样例

给出 s = `"aab"`，返回

```
[
  ["aa", "b"],
  ["a", "a", "b"]
]
```

LintCode: http://www.lintcode.com/zh-cn/problem/palindrome-partitioning/
LeetCode: https://leetcode.com/problems/palindrome-partitioning/

http://blog.csdn.net/u013291394/article/details/51212959

9.2 不同的路径 Unique Paths

有一个机器人的位于一个 *m* × *n* 个网格左上角。

机器人每一时刻只能向下或者向右移动一步。机器人试图达到网格的右下角。

问有多少条不同的路径？

注意事项

n 和 m 均不超过 100

样例

给出 m = `3` 和 n = `3`, 返回 `6`.
给出 m = `4` 和 n = `5`, 返回 `35`.

LintCode: http://www.lintcode.com/zh-cn/problem/unique-paths/
LeetCode: https://leetcode.com/problems/unique-paths/

http://blog.csdn.net/nerv3x3/article/details/38928973
http://blog.csdn.net/u013291394/article/details/50527092

9.3 不同的路径 Unique Paths II

"不同的路径" 的跟进问题：

现在考虑网格中有障碍物，那样将会有多少条不同的路径？

网格中的障碍和空位置分别用 1 和 0 来表示。

注意事项

m 和 n 均不超过 100

样例
如下所示在 3x3 的网格中有一个障碍物：

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

一共有 2 条不同的路径从左上角到右下角。

LintCode: http://www.lintcode.com/zh-cn/problem/unique-paths-ii/
LeetCode: https://leetcode.com/problems/unique-paths-ii/

http://blog.csdn.net/u013291394/article/details/50531113

9.4 恢复 IP 地址 Restore IP Addresses

给一个由数字组成的字符串。求出其可能恢复为的所有 IP 地址。
样例
给出字符串 `"25525511135"`，所有可能的 IP 地址为：

```
[
  "255.255.11.135",
  "255.255.111.35"
]
```

（顺序无关紧要）
LintCode: http://www.lintcode.com/zh-cn/problem/restore-ip-addresses/
LeetCode: https://leetcode.com/problems/restore-ip-addresses/#/description

http://blog.csdn.net/nerv3x3/article/details/31068775

9.5 数字组合 Combination Sum

给出一组候选数字(C)和目标数字(T),找到 C 中所有的组合，使找出的数字和为 T。C 中的数字可以无限制重复被选取。

例如,给出候选数组[2,3,6,7]和目标数字 7，所求的解为：

[7],

[2,2,3]

注意事项

- 所有的数字(包括目标数字)均为正整数。
- 元素组合(*a*1, *a*2, … , *ak*)必须是非降序(ie, *a*1 ≤ *a*2 ≤ … ≤ *ak*)。
- 解集不能包含重复的组合。

样例
给出候选数组[2,3,6,7]和目标数字 7

返回 [[7],[2,2,3]]

LintCode: http://www.lintcode.com/zh-cn/problem/combination-sum/
LeetCode: https://oj.leetcode.com/problems/combination-sum/

http://blog.csdn.net/nerv3x3/article/details/39453235
http://blog.csdn.net/u013291394/article/details/50454109
https://github.com/williamwhe/leetcode/blob/master/Combination-Sum.py

9.6 生成括号 Generate Parentheses

给定 n 对括号，请写一个函数以将其生成新的括号组合，并返回所有组合结果。
样例
给定 `n = 3`, 可生成的组合如下:

`"((()))", "(()())", "(())()", "()(())", "()()()"`
LintCode: http://www.lintcode.com/zh-cn/problem/generate-parentheses/
LeetCode: https://oj.leetcode.com/problems/generate-parentheses/

http://blog.csdn.net/nerv3x3/article/details/2920102
http://blog.csdn.net/u013291394/article/details/50384187

9.7 单词搜索 Word Search

给出一个二维的字母板和一个单词，寻找字母板网格中是否存在这个单词。

单词可以由按顺序的相邻单元的字母组成，其中相邻单元指的是水平或者垂直方向相邻。每个单元中的字母最多只能使用一次。
样例
给出 board =

```
[
  "ABCE",
  "SFCS",
```

"ADEE"

]

word = "ABCCED"， ->返回 true,

word = "SEE"，-> 返回 true,

word = "ABCB"， -> 返回 false.

LintCode: http://www.lintcode.com/zh-cn/problem/word-search/

LeetCode: https://leetcode.com/problems/word-search/#/description

http://blog.csdn.net/u013291394/article/details/50575168

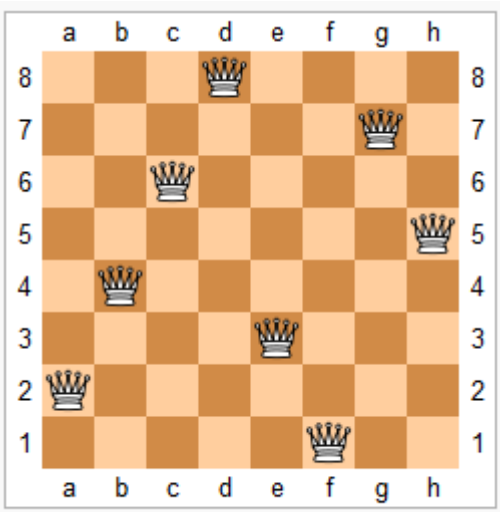
难度：Hard

9.8 N 皇后问题 N-Queens

n 皇后问题是将 n 个皇后放置在 n*n 的棋盘上，皇后彼此之间不能相互攻击。

给定一个整数 n，返回所有不同的 n 皇后问题的解决方案。

每个解决方案包含一个明确的 n 皇后放置布局，其中“Q”和“.”分别表示一个女王和一个空位置。



One solution to the eight queens puzzle

样例

对于 4 皇后问题存在两种解决的方案：

[

["..Q.", // Solution 1

"...Q",

"Q...",

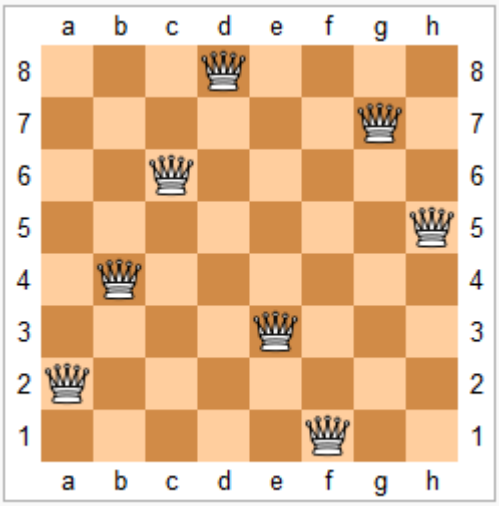
```
    "..Q."],
  ["..Q.", // Solution 2
    "Q...",
    "...Q",
    ".Q.."]
]
```

LintCode: http://www.lintcode.com/zh-cn/problem/n-queens/
LeetCode: https://oj.leetcode.com/problems/n-queens/

http://blog.csdn.net/nerv3x3/article/details/39453349
http://blog.csdn.net/u013291394/article/details/50512818
https://github.com/williamwhe/leetcode/blob/master/nqueen.py

9.9 N 皇后问题 N-Queens II

根据 n 皇后问题，现在返回 n 皇后不同的解决方案的数量而不是具体的放置布局。



One solution to the eight queens puzzle

样例
比如 n=4，存在 2 种解决方案
LintCode: http://www.lintcode.com/zh-cn/problem/n-queens-ii/
LeetCode: https://oj.leetcode.com/problems/n-queens-ii/

http://blog.csdn.net/nerv3x3/article/details/39453357
http://blog.csdn.net/u013291394/article/details/50512928
https://github.com/williamwhe/leetcode/blob/master/nqueen2.py

9.10 数独 Sudoku Solver

通过程序来解决数独问题。
注意点：
有且只有唯一解
例子：
输入：

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

输出：

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

LeetCode: http://leetcode.com/problems/sudoku-solver/

http://blog.csdn.net/u013291394/article/details/50516914

第十章 暴力枚举法

难度：Medium

10.1 子集 Subsets

给定一个含不同整数的集合，返回其所有的子集

注意事项

子集中的元素排列必须是非降序的，解集必须不包含重复的子集

样例

如果 S = `[1,2,3]`，有如下的解：

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

LintCode: http://www.lintcode.com/zh-cn/problem/subsets/
LeetCode: https://leetcode.com/problems/subsets/#/description

http://blog.csdn.net/nerv3x3/article/details/36895699
http://blog.csdn.net/u013291394/article/details/50575160

10.2 子集 Subsets II

给定一个可能具有重复数字的列表，返回其所有可能的子集

注意事项

- 子集中的每个元素都是非降序的

- 两个子集间的顺序是无关紧要的
- 解集中不能包含重复子集

样例
如果 S = `[1,2,2]`，一个可能的答案为：

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

LintCode: http://www.lintcode.com/zh-cn/problem/subsets-ii/
LeetCode: https://leetcode.com/problems/subsets-ii/#/description

http://blog.csdn.net/nerv3x3/article/details/36896337
http://blog.csdn.net/u013291394/article/details/50593967

10.3 全排列 Permutations

给定一个数字列表，返回其所有可能的排列。

注意事项
你可以假设没有重复数字。

样例
给出一个列表`[1,2,3]`，其全排列为：

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

```
]
```

LintCode: http://www.lintcode.com/zh-cn/problem/permutations/
LeetCode: https://leetcode.com/problems/permutations/

http://blog.csdn.net/nerv3x3/article/details/2920199
http://blog.csdn.net/u013291394/article/details/50476076

10.4 全排列 Permutations II

给出一个具有重复数字的列表，找出列表所有不同的排列。
样例
给出列表 `[1,2,2]`，不同的排列有：

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

LintCode: http://www.lintcode.com/zh-cn/problem/permutations-ii/
LeetCode: https://oj.leetcode.com/problems/permutations-ii/

http://blog.csdn.net/nerv3x3/article/details/37993703
http://blog.csdn.net/u013291394/article/details/50478389

10.5 组合 Combinations

组给出两个整数 n 和 k，返回从 1……n 中选出的 k 个数的组合。
样例
例如 n = 4 且 k = 2

返回的解为：

[[2,4],[3,4],[2,3],[1,2],[1,3],[1,4]]
LintCode: http://www.lintcode.com/zh-cn/problem/combinations/
LeetCode: https://oj.leetcode.com/problems/combinations/

http://blog.csdn.net/nerv3x3/article/details/39453269
http://blog.csdn.net/u013291394/article/details/50571448

10.6 电话号码的字母组合 Letter Combinations of a Phone Number

手机按键上每个数字都对应了多个字母，如 2 对应了"abc"，现给出一个数字串，要求把其中的每个数字都转化为对应的字母中的一个，列出所有的组合情况。
注意点：
对结果的排列顺序没有要求
例子：
输入: digits="23"
输出: ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"]
LintCode: http://www.lintcode.com/zh-cn/problem/letter-combinations-of-a-phone-number/
LeetCode: https://oj.leetcode.com/problems/letter-combinations-of-a-phone-number/

http://blog.csdn.net/nerv3x3/article/details/37335129
http://blog.csdn.net/u013291394/article/details/50363733

第十一章 分治法

难度：Easy

11.1 x 的平方根 Sqrt(x)

实现 `int sqrt(int x)` 函数，计算并返回 *x* 的平方根。
样例
sqrt(3) = 1

sqrt(4) = 2

sqrt(5) = 2

sqrt(10) = 3
LintCode: http://www.lintcode.com/zh-cn/problem/sqrtx/
LeetCode: https://leetcode.com/problems/sqrtx/#/description

http://blog.csdn.net/u013291394/article/details/50544536

难度：Medium

11.2 x 的 n 次幂 Pow(x,n)

实现 pow(x,n)

注意事项

不用担心精度，当答案和标准输出差绝对值小于 `1e-3` 时都算正确

样例

```
Pow(2.1, 3) = 9.261

Pow(0, 1) = 0

Pow(1, 0) = 1
```

LintCode: http://www.lintcode.com/zh-cn/problem/powx-n/
LeetCode: https://oj.leetcode.com/problems/powx-n/

http://blog.csdn.net/nerv3x3/article/details/3465663
http://blog.csdn.net/u013291394/article/details/50487773

第十二章 贪心法

难度：Easy

12.1 买卖股票的最佳时机 Best Time to Buy and Sell Stock

假设有一个数组，它的第 i 个元素是一支给定的股票在第 i 天的价格。如果你最多只允许完成一次交易(例如,一次买卖股票),设计一个算法来找出最大利润。

样例

给出一个数组样例 [3,2,3,1,2], 返回 1

LintCode: http://www.lintcode.com/zh-cn/problem/best-time-to-buy-and-sell-stock/
LeetCode: https://leetcode.com/problems/best-time-to-buy-and-sell-stock/#/description

http://blog.csdn.net/nerv3x3/article/details/2920118
http://blog.csdn.net/u013291394/article/details/51085858

12.2 买卖股票的最佳时机 Best Time to Buy and Sell Stock II

假设有一个数组，它的第 i 个元素是一个给定的股票在第 i 天的价格。设计一个算法来找到最大的利润。你可以完成尽可能多的交易(多次买卖股票)。然而,你不能同时参与多个交易(你必须在再次购买前出售股票)。

样例

给出一个数组样例[2,1,2,0,1], 返回 2

LintCode: http://www.lintcode.com/zh-cn/problem/best-time-to-buy-and-sell-stock-ii/
LeetCode: https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/

http://blog.csdn.net/nerv3x3/article/details/3465750
http://blog.csdn.net/u013291394/article/details/51107328
http://blog.csdn.net/monkeyduck/article/details/38079623

难度：Medium

12.3 跳跃游戏 Jump Game

给出一个非负整数数组，你最初定位在数组的第一个位置。

数组中的每个元素代表你在那个位置可以跳跃的最大长度。

判断你是否能到达数组的最后一个位置。

注意事项

这个问题有两个方法，一个是贪心和 动态规划。

贪心方法时间复杂度为 O（N） 。

动态规划方法的时间复杂度为为 O（n^2） 。

我们手动设置小型数据集，使大家阔以通过测试的两种方式。这仅仅是为了让大家学会如何使用动态规划的方式解决此问题。如果您用动态规划的方式完成它，你可以尝试贪心法，以使其再次通过一次。

样例

A = [2,3,1,1,4]，返回 true.

A = [3,2,1,0,4]，返回 false.

LintCode: http://www.lintcode.com/zh-cn/problem/jump-game/
LeetCode: https://leetcode.com/problems/jump-game/#/description

http://blog.csdn.net/u013291394/article/details/50499216

12.4 最长无重复字符的子串 Longest Substring Without Repeating Characters

给定一个字符串，请找出其中无重复字符的最长子字符串。

样例

例如，在"abcabcbb"中，其无重复字符的最长子字符串是"abc"，其长度为 3。

对于，"bbbbb"，其无重复字符的最长子字符串为"b"，长度为 1。

LintCode:http://www.lintcode.com/zh-cn/problem/longest-substring-without-repeating-characters/
LeetCode:https://leetcode.com/problems/longest-substring-without-repeating-characters/#/description

http://blog.csdn.net/u013291394/article/details/50285579

12.5 装最多水的容器 Container With Most Water

给定 *n* 个非负整数 $a_1, a_2, ..., a_n$，每个数代表了坐标中的一个点 `(i, ai)`。画 *n* 条垂直线，使得 *i* 垂直线的两个端点分别为`(i, ai)`和`(i, 0)`。找到两条线，使得其与 *x* 轴共同构成一个容器，以容纳最多水。

注意事项

容器不可倾斜。

样例

给出`[1,3,2]`, 最大的储水面积是 `2`.

LintCode: http://www.lintcode.com/zh-cn/problem/container-with-most-water/
LeetCode: https://leetcode.com/problems/container-with-most-water/#/description

http://blog.csdn.net/nerv3x3/article/details/37331325
http://blog.csdn.net/u013291394/article/details/50341073

难度：Hard

12.6 跳跃游戏 Jump Game II

给出一个非负整数数组，你最初定位在数组的第一个位置。

数组中的每个元素代表你在那个位置可以跳跃的最大长度。

你的目标是使用最少的跳跃次数到达数组的最后一个位置。

样例

给出数组 A = [2,3,1,1,4]，最少到达数组最后一个位置的跳跃次数是 2(从数组下标 0 跳一步到数组下标 1，然后跳 3 步到数组的最后一个位置，一共跳跃 2 次)

LintCode: http://www.lintcode.com/zh-cn/problem/jump-game-ii/
LeetCode: https://leetcode.com/problems/Jump-Game-II/

http://blog.csdn.net/u013291394/article/details/50501753
https://github.com/williamwhe/leetcode/blob/master/jump-game2.py

第十三章 动态规划

难度：Easy

12.7 最大子数组 Maximum Subarray

给定一个整数数组，找到一个具有最大和的子数组，返回其最大和。

注意事项

子数组最少包含一个数

样例

给出数组`[-2,2,-3,4,-1,2,1,-5,3]`，符合要求的子数组为`[4,-1,2,1]`，其最大和为`6`

LintCode: http://www.lintcode.com/zh-cn/problem/maximum-subarray/
LeetCode: https://leetcode.com/problems/maximum-subarray/

http://blog.csdn.net/nerv3x3/article/details/3465696
http://blog.csdn.net/u013291394/article/details/50491468

难度：Medium

12.8 数字三角形 Triangle

给定一个数字三角形，找到从顶部到底部的最小路径和。每一步可以移动到下面一行的相邻数字上。

注意事项

如果你只用额外空间复杂度 O(n)的条件下完成可以获得加分，其中 n 是数字三角形的总行数。

样例

比如，给出下列数字三角形：

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

从顶到底部的最小路径和为 11 (2 + 3 + 5 + 1 = 11)。

LintCode: http://www.lintcode.com/zh-cn/problem/triangle/
LeetCode: https://leetcode.com/problems/triangle/

http://blog.csdn.net/u013291394/article/details/50937097

12.9 最小路径和 Minimum Path Sum

给定一个只含非负整数的 m*n 网格，找到一条从左上角到右下角的可以使数字和最小的路径。

注意事项

你在同一时间只能向下或者向右移动一步

LintCode: http://www.lintcode.com/zh-cn/problem/minimum-path-sum/
LeetCode: https://leetcode.com/problems/minimum-path-sum/#/description

http://blog.csdn.net/u013291394/article/details/50532528

12.10 解码方法 Decode Ways

有一个消息包含 `A-Z` 通过以下规则编码

```
'A' -> 1
'B' -> 2
...
'Z' -> 26
```

现在给你一个加密过后的消息，问有几种解码的方式

样例

给你的消息为 `12`，有两种方式解码 AB(12) 或者 L(12). 所以返回 `2`

LintCode: http://www.lintcode.com/zh-cn/problem/decode-ways/
LeetCode: https://leetcode.com/problems/decode-ways/#/description

http://blog.csdn.net/nerv3x3/article/details/2921931
http://blog.csdn.net/u013291394/article/details/50596724
https://github.com/williamwhe/leetcode/blob/master/decodeways.py

12.11 单词切分 Word Break

给出一个字符串 s 和一个词典，判断字符串 s 是否可以被空格切分成一个或多个出现在字典中的单词。
样例
给出

s = "lintcode"

dict = ["lint","code"]

返回 true 因为"lintcode"可以被空格切分成"lint code"
LintCode: http://www.lintcode.com/zh-cn/problem/word-break/
LeetCode: http://oj.leetcode.com/problems/word-break/

http://blog.csdn.net/u013291394/article/details/51318411

难度：Hard

12.12 分割回文串 Palindrome Partitioning II

给定一个字符串 s，将 s 分割成一些子串，使每个子串都是回文。

返回 s 符合要求的的最少分割次数。
样例
比如，给出字符串 s = "aab"，

返回 1， 因为进行一次分割可以将字符串 s 分割成["aa","b"]这样两个回文子串
LintCode: http://www.lintcode.com/zh-cn/problem/palindrome-partitioning-ii/
LeetCode: https://leetcode.com/problems/palindrome-partitioning-ii/#/description

http://blog.csdn.net/u013291394/article/details/51245538

12.13 最大矩形 Maximal Rectangle

给你一个二维矩阵，权值为 `False` 和 `True`，找到一个最大的矩形，使得里面的值全部为 `True`，输出它的面积
样例
给你一个矩阵如下

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

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

输出 6

LintCode: http://www.lintcode.com/zh-cn/problem/maximal-rectangle/

LeetCode: https://leetcode.com/problems/maximal-rectangle/

http://blog.csdn.net/u013291394/article/details/50865281

https://github.com/williamwhe/leetcode/blob/master/max-rect.py

12.14 买卖股票的最佳时机 Best Time to Buy and Sell Stock III

假设你有一个数组，它的第 i 个元素是一支给定的股票在第 i 天的价格。设计一个算法来找到最大的利润。你最多可以完成两笔交易。

注意事项

你不可以同时参与多笔交易(你必须在再次购买前出售掉之前的股票)

样例

给出一个样例数组 [4,4,6,1,1,4,2,5], 返回 6

LintCode: http://www.lintcode.com/zh-cn/problem/best-time-to-buy-and-sell-stock-iii/

LeetCode: https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/#/description

http://blog.csdn.net/u013291394/article/details/51125454

12.15 交叉字符串 Interleaving String

给出三个字符串:s1、s2、s3，判断 s3 是否由 s1 和 s2 交叉构成。

样例

比如 s1 = "aabcc" s2 = "dbbca"

- 当 s3 = "aadbbcbcac"，返回 true.
- 当 s3 = "aadbbbaccc"， 返回 false.

LintCode: http://www.lintcode.com/zh-cn/problem/interleaving-string/

LeetCode: https://leetcode.com/problems/Interleaving-String/

http://blog.csdn.net/nerv3x3/article/details/4258745

http://blog.csdn.net/u013291394/article/details/50620484

12.16 攀爬字符串 Scramble String

给定一个字符串 `S1`，将其递归地分割成两个非空子字符串,从而将其表示为二叉树。

下面是 `s1 = "great"`的一个可能表达：

```
    great
   /    \
  gr    eat
 / \    /  \
g   r  e   at
           / \
          a   t
```

在攀爬字符串的过程中，我们可以选择其中任意一个非叶节点，然后交换该节点的两个儿子。

例如，我们选择了 `"gr"` 节点，并将该节点的两个儿子进行交换，从而产生了攀爬字符串 `"rgeat"`。

```
    rgeat
   /    \
  rg    eat
 / \    /  \
r   g  e   at
           / \
          a   t
```

我们认为， `"rgeat"` 是 `"great"` 的一个攀爬字符串.

类似地，如果我们继续将其节点 `"eat"` 和 `"at"` 进行交换，就会产生新的攀爬字符串 `"rgtae"`。

```
    rgtae
   /    \
  rg    tae
 / \    /  \
r   g  ta  e
```

```
    / \
   t   a
```

同样地，`"rgtae"` 也是 `"great"`的一个攀爬字符串。

给定两个相同长度的字符串 `s1` 和 `s2`，判定 `s2` 是否为 `s1` 的攀爬字符串。

LintCode: http://www.lintcode.com/zh-cn/problem/scramble-string/
LeetCode: https://leetcode.com/problems/scramble-string/#/description

http://blog.csdn.net/u013291394/article/details/50603159
https://github.com/williamwhe/leetcode/blob/master/scramble.py

12.17 编辑距离 Edit Distance

给出两个单词 word1 和 word2，计算出将 word1 转换为 word2 的最少操作次数。

你总共三种操作方法：

- 插入一个字符
- 删除一个字符
- 替换一个字符

样例
给出 work1="mart" 和 work2="karma"

返回 3
LintCode: http://www.lintcode.com/zh-cn/problem/edit-distance/
LeetCode: https://leetcode.com/problems/edit-distance/#/

http://blog.csdn.net/nerv3x3/article/details/37334113
http://blog.csdn.net/u013291394/article/details/50550512

12.18 不同子序列 Distinct Subsequences

给出字符串 S 和字符串 T，计算 S 的不同的子序列中 T 出现的个数。

子序列字符串是原始字符串通过删除一些(或零个)产生的一个新的字符串，并且对剩下的字符的相对位置没有影响。(比如，“ACE”是“ABCDE”的子序列字符串,而“AEC”不是)。
样例
给出 S = "rabbbit", T = "rabbit"

返回 3
LintCode: http://www.lintcode.com/zh-cn/problem/distinct-subsequences/
LeetCode: https://leetcode.com/problems/distinct-subsequences/

http://blog.csdn.net/u013291394/article/details/50843558
https://github.com/williamwhe/leetcode/blob/master/distinct-subsequences.py

12.19 单词切分 Word Break II

Given a string s and a dictionary of words dict, add spaces in s to construct a sentence where each word is a valid dictionary word.

Return all such possible sentences.
样例
Gieve s = `lintcode`,
dict = `["de", "ding", "co", "code", "lint"]`.

A solution is `["lint code", "lint co de"]`.
LintCode: http://www.lintcode.com/zh-cn/problem/word-break-ii/
LeetCode: https://leetcode.com/problems/word-break-ii/#/description

http://blog.csdn.net/u013291394/article/details/51457343

第十四章 图

难度：Medium

14.1 克隆图 Clone Graph

克隆一张无向图，图中的每个节点包含一个 `label` 和一个列表 `neighbors`。

数据中如何表示一个无向图？http://www.lintcode.com/help/graph/

你的程序需要返回一个经过深度拷贝的新图。这个新图和原图具有同样的结构，并且对新图的任何改动不会对原图造成任何影响。

样例
比如，序列化图 `{0,1,2#1,2#2,2}` 共有三个节点，因此包含两个个分隔符#。

1. 第一个节点 label 为 0，存在边从节点 0 链接到节点 1 和节点 2
2. 第二个节点 label 为 1，存在边从节点 1 连接到节点 2
3. 第三个节点 label 为 2，存在边从节点 2 连接到节点 2(本身),从而形成自环。

我们能看到如下的图：

```
1
```

```
  / \
 /   \
0 --- 2
     / \
     \_/
```

LintCode: http://www.lintcode.com/zh-cn/problem/clone-graph/
LeetCode: https://leetcode.com/problems/clone-graph/#/description

http://blog.csdn.net/u013291394/article/details/51254404