九章算法
www.jiuzhang.com

美团2020面试题解析
算法岗+开发岗+数据分析岗

九章算法 令狐冲

九章算法
www.jiuzhang.com

校招 vs 社招

哪个更难？为什么？

九章算法
www.jiuzhang.com

国内算法面试题特点

题目背景老TM长

动态规划老TM考

考智商题真TM多

算法岗 > 开发岗 > 数据岗

题目1：黑客入侵点定位

集团内部有一核心机密项目，共由150个代码模块按顺序串行执行组成（示例：模块1->模块2->...模块N...->模块149->模块150)。偶然一天，某一个模块突然被黑客入侵（当前模块也称入侵点）。因为内部已经有预防措施，现存两款从不同角度设计的反入侵检测程序，但同时也有一定检测限制：

1.输入：顺序代码段，必须以模块1开始（比如：模块1->模块2->...模块39）

2:输出：True-输入包含入侵点，False-输入不包含入侵点

3.每款检测程序可以运行多次：可多次返回False,但仅能返回一次True（由于入侵的对抗性存在，一旦输出True即报废，后续检测功能失效）

4.同一时刻只能有一款检测程序运行，每检测一次耗时10分钟

现在希望仅利用现有的两款反入侵检测程序，在最短的时间内确保可以快速定位入侵点。具体

如何设计检测流程，时间是多久。

黑客问题第一步：简化题型

有一个长度为 n=150 的 01 数组 [0,0,0...0] 里包含了一个 1

一个检测代码 containsOne(i) 可以检测前 i 个数里有没有 1

一旦 containsOne(i) return true，记错一次耗费

请在做多2次耗费的情况下，找到 1 所在的下标 i，并且使得 containsOne 的调用次数最少

举例，如 n=9 的 01数组 [0,0,0,0,0,1,0,0,0]

可以依次检测 containsOne(3) => false, containsOne(6) => true 一次耗费

再依次检测 containsOne(4) => false, containsOne(5) => false，此时可以确定 1 在 6

总共调用 4次 containsOne，是最少的调用次数。

九章算法
www.jiuzhang.com

黑客问题第二步：找相似题

这类题统称为 - 知道就知道，不知道就拉倒题

相似题 - 楼层扔鸡蛋问题

一个 n 层高的楼层，测试鸡蛋的耐摔程度，一共有两个鸡蛋，问最少仍几次能够测出鸡蛋的耐摔程度？

耐摔程度的定义为如果在第 k-1 层扔下去不破，但是第 k 层开始扔下去就破了，那么耐摔程度就是 k。如果在第 n 层扔下去也不破，那么耐摔程度定义为 n + 1。

此类问题的通用解法 - 归纳法

归纳法 = 不管题目给多大的数据，自己先从小数据开始一个个试，然后找规律

N = 1 => 1次

N = 2 => 2次 N = 3 => 2次 N = 4 => 3次

N = 5 => 3次 N = 6 => 3次 N = 7 => 4次

...

N = 10 => 4次

N = 11 => 5次

...

规律是什么？

进一步追问

如果有 n 层楼，m 个鸡蛋，最优的策略如何计算？

九章算法
www.jiuzhang.com

动态规划

——国内大厂必考算法

f[i][j] 表示一共 i 层，最多 j 个鸡蛋需要最少测多少次

f[i][j] = min{max(f[k-1][j-1], f[i - k][j]) + 1}

参考代码：http://www.jiuzhang.com/solutions/drop-eggs-ii

想拿A+：还有没有更快的办法?

刚才的时间复杂度是 O(n^2*m), n = 楼层高度, m = 鸡蛋个数

结合一下 2 个鸡蛋的思路有没有更快的办法?

还是动态规划

f[i][j] 代表一共 i 个鸡蛋，扔 j 次最多可以测出来的楼层高度

f[i][j] = f[i - 1][j - 1] + 1 + f[i][j - 1]

找到最小的 j 使得 f[m][j] >= n

时间复杂度 O(m * Answer) 完美！

领扣在线评测地址

https://www.lintcode.com/problem/drop-eggs/

https://www.lintcode.com/problem/drop-eggs-ii/

九章算法
www.jiuzhang.com

《九章动态规划专题班》

涵盖了所有DP面试的核心考点，"套路模板+解题秘籍"帮你轻松攻克大厂的出题套路。

解题秘籍：帮你解决80%以上的DP难题

题型总结：大厂面试必考DP题型大盘点

短期高效：轻松找到最优解法，突破面试难关

题目2：外卖小哥的保温箱

众所周知，美团外卖的口号是:''美团外卖,送啥都快''。身着黄色工作服的骑手作为外卖业务中商家和客户的重要纽带，在工作中，以快速送餐突出业务能力；工作之余，他们会通过玩智力游戏消遣闲暇时光，以反应速度彰显智慧，每位骑手拿出装有货物的保温箱，参赛选手需在最短的时间内用最少的保温箱将货物装好。

我们把问题简单描述一下:

1 每个货物占用空间都一模一样

2 外卖小哥保温箱的最大容量是不一样的,每个保温箱由两个值描述: 保温箱的最大容量 bi ,当前已有货物个数 ai ,(ai<=bi)

3 货物转移的时候,不必一次性全部转移,每转移一件货物需要花费 1秒 的时间

输入输出

输入描述:

第一行包含n个正整数(1<=n<=100)表示保温箱的数量

第二行有n个正整数a1,a2,...,an(1<=ai<=100)

ai表示第i个保温箱的已有货物个数

第三行有n个正整数b1,b2,...,bn(1<=bi<=100),bi表示第i个保温箱的最大容量

显然,每一个ai<=bi

输出描述:

输出为两个整数k和t, k表示能容纳所有货物的保温箱的最少个数,t表示将所有货物转移到这k个保温箱所花费的最少时间,单位为秒.

样例

输入例子1:

4

3 3 4 3

4 7 6 5

输出例子1:

2 6

尝试：贪心法

首先将所有的食物加起来，得到一个最少需要的容量
然后排序所有箱子，从大到小取箱子直到取到足够的容量
这样我们可以得到最少的箱子数

这个想法对么?

更进一步的，计算出所需的最少箱子数之后，就知道要“扔掉”多少箱子，设为 k

最少的耗费 = Min{Sigma(扔掉的箱子里的餐盒)}

于是按照箱子里的餐盒数正序排列取前 k 个

九章算法
www.jiuzhang.com

你所能想到的贪心法都是错的

面试中千万不要使用贪心法！

背包类动态规划

找最大最小的问题，一般来说，都可以试试看动态规划

动态规划解决三大类问题：找最值，求方案数，问可行性

背包类动态规划的特点是有“体积之和”的概念在题目中

状态描述

f[i][j] 表示前 i 个箱子，挑出若干个箱子后构成 j 的容量时的最优挑选方案 <x, y> 是什么

其中 x 代表最优方案里的箱子总数最少是多少

y 表示最优方案里在箱子总数最小的情况下这些箱子里的食物之和最大是多少

状态转移方程

考虑第 i 个箱子要不要被选中，产生两种决策

不选中，相当于要在前 $i-1$ 个箱子里选出 j 的体积，转移到 $<f[i-1][j].x, f[i][j].y>$

选中，相当于要在前 $i-1$ 个箱子里选出 $j-b[i]$ 的体积，转移到 $<f[i-1][j-b[i]].x+1, f[i-1][j-b[i]].y+a[i]>$

比较两者中得到最优的二元组，赋值给 $f[i][j]$ 即可

时间复杂度

设 n = 箱子个数(大概 100), m = 箱子体积之和(大概 10000)

O(n * m) = 大概 10^6 很高效

相关题

https://www.lintcode.com/problem/minimum-adjustment-cost/

题目3：最长公共前缀

输入n个字符串（1<=n<=300，字符串总长度L不超过1000，只包含小写字母）

后面多次查询，每次查询输入两个数字x，y，输出第x个字符串和第y个字符串的最长公共前缀长度。

（查询次数K不超过100）

输入描述:

第1行输入一个整数n，代表字符串数量；

第2~n+1行，每行一个字符串；

第n+2行开始，每行输入两个整数a和b，代表需要计算公共前缀的字符串编号。

输出描述:

每次查询输出一行一个整数，表示两个字符串的最长公共前缀的长度

最长公共前缀 - 例子

输入例子1:

2

abc

abe

1 2

输出例子1:

2

拿题先暴力

n=字符串个数，m=字符串长度，k=查询次数

两个字符串从头到尾比较 O(k*m)

这个算法面试的时候可以过，但是时间复杂度和 n 无关，是否可以利用上这一点进行优化？

公共前缀的特点

字符串数组排序之后，公共前缀相同的会被挤在一起

最长公共前缀 - 算法流程

第一步：排序，O(nlogn*L)

第二步：计算相邻字符串之间的最长公共前缀，O(nL)

第三步：将相邻字符串之间最长公共前缀的长度，放到线段树中

第四步：查询两个字符串的最长公共前缀 = 线段树中两个字符串之间的相邻公共前缀长度最小值

每次查询时间复杂度 O(logn)，k次查询 O(klogn)

总结：

预处理时间复杂度：O(nlogn*L)

每次查询时间复杂度：O(klogn)

九章算法
www.jiuzhang.com

万能数据结构——
线段树 SegmentTree

Java/Python/C++ 都没有自带，需要自己实现

利用了类似 Merge Sort 的分治思想

可以 logN 的时间内做到所有堆，红黑树，树状数组做的事情，还额外支持区间查询

相关领扣测试题

https://www.lintcode.com/problem/longest-common-prefix/description

https://www.lintcode.com/problem/the-longest-common-prefix-ii/description

题目4：字符串排序

生活中经常有需要将多个字符串进行排序的需要，比如将美团点评的部分业务名称（外卖、打车、旅游、丽人、美食、结婚、旅游景点、教培、门票、酒店），用拼音表示之后按字母逆序排序。字母逆序指从z到a排序，比如对两个字符串排序时，先比较第一个字母按字母逆序排z在a的前面，当第一个字母一样时再比较第二个字母按字母逆序排，以此类推。特殊情况1)空字符串需排在最前面；2)若一个短字符串是另一个长字符串的前缀则短字符串排在前面。请自行实现代码进行排序，直接调用sort等排序方法将不得分且视为作弊。

输入输出

输入描述:

输入为一行，由多个字符串以英文逗号拼接而成，最多不超过128个字符串且可能有重复。每个字符串由小写字母a-z组成，可以为空，最长不超过128个字符。

输出描述:

输出一行，为排序之后的字符串，用逗号隔开

输入例子1:

waimai,dache,lvyou,liren,meishi,jiehun,lvyoujingdian,jiaopei,menpiao,jiudian

输出例子1:

waimai,menpiao,meishi,lvyou,lvyoujingdian,liren,jiudian,jiehun,jiaopei,dache

九章算法
www.jiuzhang.com

面试必考—— nlogn 排序算法的实现

https://www.jiuzhang.com/solutions/quick-sort
https://www.jiuzhang.com/solutions/merge-sort

题目5：工作安排

小美是团队的负责人，需要为团队制定工作的计划，以帮助团队产出最大的价值。

每周团队都会有两项候选的任务，其中一项为简单任务，一项为复杂任务，两项任务都能在一周内完成。第i周，团队完成简单任务的价值为li，完成复杂任务的价值为hi。由于复杂任务本身的技术难度较高，团队如果在第i周选择执行复杂任务的话，需要在$i-1$周不做任何任务专心准备。如果团队在第i周选择执行简单任务的话，不需要提前做任何准备。

现在小美的团队收到了未来N周的候选任务列表，请帮助小美确定每周的工作安排使得团队的工作价值最大。

输入输出

输入描述:

第一行为N（0≤N≤1000）

接下来的N行表示第1到N周两项候选任务的价值，第i行的格式为：li hi，其中 0 < li < 10000，0< hi < 10000。

输出描述:

输出一个数字，表示小美团队在未来N周能产出的最大价值。

样例

输入例子1:

4

10 5

1 50

10 5

10 1

输出例子1:

70

还是动态规划

求最优值问题，基本都可以朝着动态规划去想

状态与转移方程

状态：f[i][0] 表示前 i 周，第 i 周 **不做任务** 的最大获益

f[i][1] 表示前 i 周，第 i 周 **做任务** 的最大获益

$$f[i][0] = \max(f[i-1][0], f[i-1][1])$$

$$f[i][1] = \max(\max(f[i-1][0], f[i-1][1]) + low[i], f[i-1][0] + high[i])$$

根据公式可以看出 $f[i][1] > f[i][0]$，简化后如下：

$$f[i][0] = f[i-1][1]$$

$$f[i][1] = \max(f[i-1][1] + low[i], f[i-1][0] + high[i])$$

进一步简化状态 f[i] = 前 i 周的最大收益

$$f[i] = \max(f[i-1] + low[i], f[i-2] + high[i])$$

数据分析岗校招题

特点：简单太多了

数据范围小，不一定要用最优做法

题目1：棋子翻转

在4*4的棋盘上摆满了黑白棋子，黑白两色的位置和数目随机其中左上角坐标为(1,1),右下角坐标为(4,4),现在依次有一些翻转操作，要对一些给定坐标为中心的上下左右四个棋子的颜色进行3次翻转，即将黑棋翻为白棋，将白棋翻为黑棋。请计算出翻转后的棋盘颜色。

输入：

- 第一行为初始棋盘，为4*4矩阵，期中0表示白色棋子，1表示黑色棋子
- 第二行为翻转位置。

输出：

- 返回翻转后的矩阵，为4*4的矩阵。

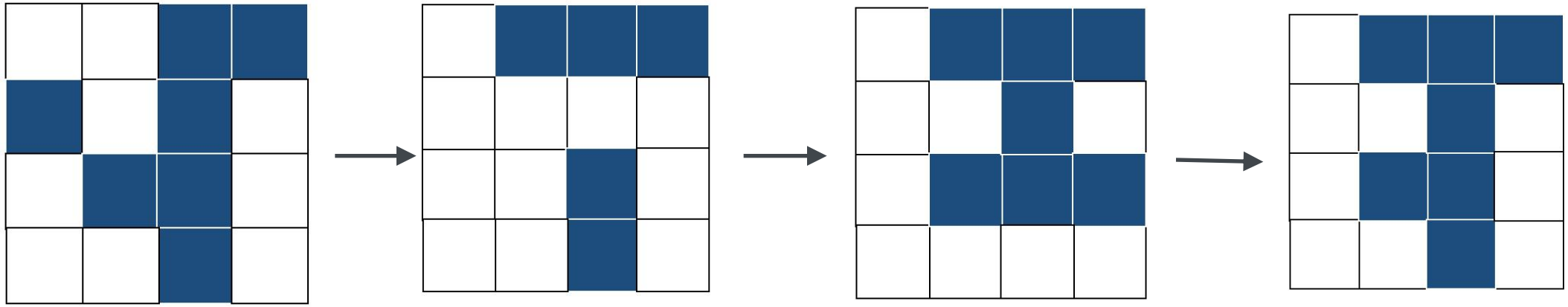
输入例子：

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

[[2,2],[3,3],[4,4]]

输出例子

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



直接模拟

使用异或操作能够更便捷地解决问题

异或符号：^

0^0=0

1^0=1

0^1=1

1^1=0

将每个相邻位置上的值异或1即可

题目2: 寻找最后的山峰

山峰元素是指其值大于或等于左右相邻值的元素。给定一个输入数组*nums*，任意两个相邻元素值不相等，数组可能包含多个山峰。找到索引最大的那个山峰元素并返回其索引。

假设 *nums[-1] = nums[n] = -∞*。

输入：

在命令行中输入一行数字，数字之间以空格分割，遇到换行符结束。输入的数字为整型，且总数量在*10*万以内。

输出：

输出索引最大的山峰的索引值（一个数字）

输入例子：

[2, 4, 1, 2, 7, 8, 4]

输出例子

5

暴力判断

枚举数组的每一个位置，若为山峰（大于左边和右边），则记录该位置

返回最后记录的位置

Follow up？返回任意一个山峰

二分搜索

查找可能区间的中间点

若该点比左边的大，比右边的小，则在右半部分查找

若该点比右边的大，比左边的小，则在左半部分查找

若比左右都小？往两边找都行

若比左右都大？直接返回答案！

相关练习

tvhttps://www.lintcode.com/problem/maximum-number-in-mountain-sequence

https://www.lintcode.com/problem/peak-index-in-a-mountain-array

https://www.lintcode.com/problem/find-peak-element/

题目3: 比大小

给定一个整数数组，返回一个数组。该返回数组中第i个数字为，原数组中第i个位置的数字至少往右走多少步才能遇到比它大的数字。如果遇不到或者已经处于最右的位置，则置为-1。

输入描述:

输入为多行，第一行为一个整数N，1≤N≤10^6

接下来一共有N个整数M由空格分割，0≤M≤2^32-1

输出描述:

输出一行共 N 个数字，每个数字表示转换之后的数组

输入例子:

5

91 10 3 22 40

输出例子:

-1 2 1 1 -1

先来暴力！

对于每个数字，不断往后查找是否有比它更大的数。

两重循环，时间复杂度O(N^2)

能否优化?

顺着不行，就倒着来！

数据结构：单调栈

记录所有元素之后，可能成为该元素的比它大的第一个元素的值和位置

a = [1,3,7,6,9]

对于a[1]来说

其后三个数7,6,9是否有一丝可能a[1]的下一个比它大

对于a[2]，如果a[1] < 7(a[2])，那么7会是a[1]的第一个比它大的元素

对于a[4]，如果a[1] >7(a[2]) 且a[1] < 9(a[4])，那么a[4]会是a[1]的第一个比它大的元素

但a[3]，永远不可能成为a[1]的下一个比它大的元素

连备胎都算不上的元素不配被记录。

数据结构：单调栈

记录所有元素之后，可能成为该元素的下一个元素的值和位置

新建一个栈，栈中元素为{index, value}形式

从后往前遍历数组，若当前值比栈顶大则栈顶出栈。

栈不断进行出栈操作，直到栈顶元素大于当前值或栈为空

多出栈操作结束，栈为空，则该元素往右走找不到比它大的数。

若栈不为空，该元素至少要往右走到栈顶元素所在的位置。

操作结束后，将该元素入栈。

相关练习

https://www.lintcode.com/problem/maximal-rectangle/

https://www.lintcode.com/problem/max-tree/description

https://www.lintcode.com/problem/largest-rectangle-in-histogram/

题目4：关联查询

数据对象data1List，员工表，存储员工ID，员工姓名

数据对象data2List， 员工工作时长表，存储员工ID，月份，工时

计算每个员工1-3月每月工时及总工时

输入描述:

员工ID 员工姓名

员工ID 月份 工时 月份 工时 月份 工时

输出描述:

员工姓名 空格 一月份工时 空格 二月份工时 空格 三月份工时 空格 总工时

输入输出

输入例子:

1 zhangwei01

1 01 200 02 150 03 196

输出例子:

zhangwei01 200 150 196 546

用map来记录信息

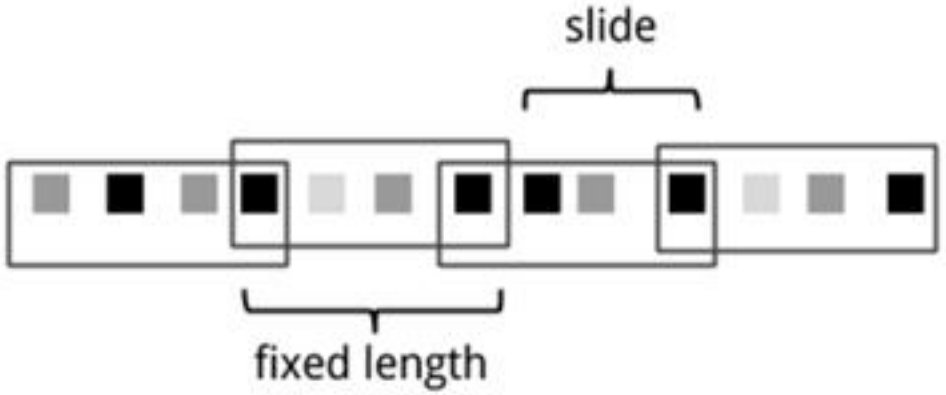
用一个map记录每个id的名字、1月工时、2月工时、3月工时

遍历map，输出结果即可

题目5：滑动窗口中位数

有一个长度为 k 的滑动窗口从数组的最左侧移动到数组的最右侧。注意你只可以看到在滑动窗口 k 内的数字，滑动位移大小=1，即滑动窗口每次只向右移动一位。

要求返回每一个滑动窗口内的中位数，解释中位数定义，例如：对于[2,3,4]，中位数是3；对于[2,3]，中位数是 (2 + 3) / 2 = 2.5



输入输出

输入描述:

输入两个数字n，k。n表示数组长度，k表示窗口大小

加下来n个整数用空格隔开，表示nums数组

(1<=k<=n)

(1<=n<=1000)

输出描述:

输出若干个数字，表示滑窗依次移动得到的结果，保留小数点后一位数字

例子

滑动窗口	中位数
[1 3 -1] -3 5 3 6 7	1
1 [3 -1 -3] 5 3 6 7	-1
1 3 [-1 -3 5] 3 6 7	-1
1 3 -1 [-3 5 3] 6 7	3
1 3 -1 -3 [5 3 6] 7	5
1 3 -1 -3 5 [3 6 7]	6

数据结构设计问题

需要一个这样的数据结构（可以理解为一个数的集合），支持

快速插入一个元素

快速删除一个元素

求集合中元素的中位数

暴力做法

数据量其实不大，即便每次都排序来找中位数，也只是 $O(n * klogk)$

稍微优化一点，始终保持一个大小k的排序数组，在排序数组上删除和插入一个元素（类似插入排序），都是 $O(k)$ 的，那可以降低到 $O(n * k)$ 通过面试是没问题的

九章算法
www.jiuzhang.com

A+的做法（或算法工程师的要求）

使用两个堆来实现

最大堆+最小堆

将滑动窗口中的元素，放在两个堆里

比较小的一半放在最大堆里，最多 n // 2 个元素

比较大的一半放在最小堆里，最多 (n + 1) // 2个元素

也就是当窗口内一共奇数个数时，比较大的这一半多放一个数

可以知道如果一共奇数个数，中位数就是右侧最小堆的堆顶（较大的一半的最小值）

如果一共偶数个数，中位数就是左右堆顶加起来除以二

每次删除一个元素时，从两个堆里找到并删除这个元素

每次插入一个元素时，根据和中位数比较判断放在较大的一半还是较大的一半

插入和删除之后，可能会破坏左右两边元素个数均衡的状态，此时做一次调整，如果左边多了

实现巨麻烦！

Sliding Window Median 比 Data Stream Median 要难很多

在于要支持堆的删除操作，那么必须记录下原数组中的每个位置的数，在哪个堆的哪个位置

这里要用到 Hash + Heap 的办法

并且在实现 Heap 的过程中，每次交换元素时，也要把 Hash 中的只对应修改

时间复杂度是 O(Nlogk)

领扣在线评测地址

https://www.lintcode.com/problem/sliding-window-median/

https://www.lintcode.com/problem/find-median-from-data-stream/description

题目6：月份天数

输入年份月份，请输出这个月的天数

输入例子1:

2018 2

2020 2

2019 1

输出例子1:

28

29

31

九章算法
www.jiuzhang.com

终于来了一道简单题

但是要把这个题写得优雅也不容易

优雅的写法 - 巧用常量

```
NORMAL_YEAR_DAYS = [-1, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
LEAP_YEAR_DAYS = [-1, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

def isLeapYear(year):
    return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

def getMonthDays(year, month):
    if month == 2 and isLeapYear(year):
        return LEAP_YEAR_DAYS[month]
    return NORMAL_YEAR_DAYS[month]
```

题目7：整数分解

一个正整数 N 可以分解为 M(M>1)个正整数的和，例如N=5、M=2时可以分解为(1+4,2+3)。

给定一个正整数 N(1<N<200) 及正整数 M(1<M<200)

求有多少种可能的分解组合 (注：K+L和L+K算一种)

输入输出与样例

输入描述:

输入两个数N和M

输出描述:

可以分解的组合数。

输入例子1:

5,2

输出例子1:

2

输入例子2:

6,3

输出例子2:

3

九章算法
www.jiuzhang.com

方案总数类依然是动态规划

不学DP就狗带

状态与转移方程

f[i][j][k] = i 拆成 j 个数最后一个数是k有多少种方法

f[i][j][k] = Sigma(f[i - k][j - 1][x], 1 <= x <= k)

时间复杂度 O(n^2 * m)

进一步优化

考虑第一个数是不是 1

$f[i][j] = f[i - 1][j - 1] + f[i - j][j]$

其中 $f[i - 1][j - 1]$ 代表第一个数是 1 的方案数

$f[i - j][j]$ 代表第一个数（最小的数）不是 1 的方案数，-j 代表既然都 > 1，那每个数都 -1

时间复杂度 $O(n * m)$