2009 年暑假集训讲义

上海交通大学 马融

USACO 题目中常见的单词：

pasture	牧场
barn	牛棚
trail	小路
grazing location	放牧地点
intersection	十字路口
FJ	农夫约翰（Farmer John）的缩写
Bessie	贝西，一头非常聪明的母牛
hay	干草
hoof	牛蹄（复数 hooves）

第一讲　穷举与贪心

集市班车 (Fair Shuttle, USACO 2009 Feb)

逛逛集市，兑兑奖品，看看节目对农夫约翰来说不算什么，可是他的奶牛们非常缺乏锻炼——如果要逛完一整天的集市，他们一定会筋疲力尽的。所以为了让奶牛们也能愉快地逛集市，约翰准备让奶牛们在集市上以车代步。

但是，约翰木有钱，他租来的班车只能在集市上沿直线跑一次，而且只能停靠 N $(1 \le N \le 20000)$个地点（所有地点都以1到N之间的一个数字来表示）。现在奶牛们分成 K $(1 \le K \le 50000)$个小组，第 i 组有M_i $(1 \le M_i \le N)$头奶牛，他们希望从S_i跑到T_i $(1 \le S_i < T_i \le N)$。

由于班车容量有限，可能载不下所有想乘车的奶牛们，此时也允许小组里的一部分奶牛分开乘坐班车。约翰经过调查得知班车的容量是C $(1 \le C \le 100)$，请你帮助约翰计划一个尽可能满足更多奶牛愿望的方案。

输入格式

- 第一行：包括三个整数：K，N和C，彼此用空格隔开。
- 第二行到$K+1$行：在第$i+1$行，将会告诉你第i组奶牛的信息：S_i，E_i和M_i，彼此用空格隔开。

输出格式

- 第一行：可以坐班车的奶牛的最大头数

样例

shuttle.in	shuttle.out
8 15 3 1 5 2 13 14 1 5 8 3 8 14 2 14 15 1 9 12 1 12 15 2 4 6 1	10 （班车可以把2头奶牛从1送到5，3头奶牛从5送到8，2头奶牛从8送到14，1头奶牛从9送到12，1头奶牛从13送到14，1头奶牛从14送到15）

解题思路

贪心，在任何时刻总是带最早下车的奶牛，证明略。

翻转棋 (Fliptile, USACO 2007 Nov)

农夫约翰知道，聪明的奶牛可以产更多的牛奶。他为奶牛设计了一种智力游戏，名叫翻转棋。翻转棋可以分成$M \times N$ $(1 \leq M, N \leq 15)$个格子，每个格子有两种颜色，一面是黑的，一面是白的。

一旦翻转某个格子，这个格子的颜色就会颠倒。如果把所有的格子都翻成白的，就算奶牛赢了。然而，奶牛的蹄子很大，一旦它们打算翻转某个格子，这个格子附近（即和这个格子有公共边）的格子也会被翻转。一直翻来翻去也很无聊，奶牛们想最小化必须翻动的次数。

请帮助奶牛确定翻动的最少次数和具体的翻法。如果最小解有多个，则输出在字典序意义下最小的那个，如果不可能完成任务，则只要输出一行单词：IMPOSSOBLE。

输入格式

- 第一行：两个用空格分开的整数：M和N
- 第二行到$M+1$行：第$i+1$行从左至右依次描述了棋盘第i行的的颜色，共记N个用空格分开的整数，1 代表黑色，0 代表白色。

输出格式

- 第一行到第M行：每行输出N个用空格分开的整数，表示在这个格子上翻转的次数。

样例

fliptile.in	fliptile.out
4 4 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1	0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 （另外还有一个解是 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 但是这个解在字典序意义下没有样例来得小）

解题思路

如果第一行的翻法固定了，其余行的翻法是固定的，所以只要穷举第一行的翻法就可以了。

翻转奶牛 (Face The Right Way, USACO 2007 Mar)

农夫约翰把N $(1 \le N \le 5000)$头奶牛排成了一行，其中大部分的奶牛都很好，他们的脸都朝向正面，但就有一小撮的奶牛是朝向背面的，为了追求完美，约翰决定要让所有的牛都朝向正面。

幸运的是，约翰最近刚买了一部自动翻牛机。不过因为他买的是打折机型，所以事先必须要设定单次翻转的牛头数K $(1 \le K \le N)$，一旦固定，就再也不能修改了。而且这台机器只能同时翻转站在一起的奶牛，每次使用的时候，它会翻转相邻的K头奶牛（不能少于K头，就算靠在队列的两个端点上也一样），一头牛在被翻转之前是朝向正面的，使用之后就会被翻到背面去了，反之亦然。

由于约翰只能选一个不可以修改的K，所以请帮他确定一个K，使得使用机器的次数最少，同时，也请把这个最少的次数M求出来（如果有多个K满足条件，输出最小的那个）。

输入格式

- 第一行：单个整数：N
- 第二行到第$N+1$行：第$i+1$行只有一个字符 B 或 F，表示第i头奶牛是朝向正面（F）还是背面的（B）。

输出格式

- 第一行：两个用空格分开的整数：K和M

样例

cowturn.in	cowturn.out
7 B B F B F B B	3 3 （选择$K=3$，翻转三次：$(1,2,3)$，$(3,4,5)$，$(5,6,7)$即可）

解题思路

枚举K，从头扫描需要翻转的点，每个点的状态只和它前$K-1$个点翻转次数的奇偶性有关，所以可以在线性时间内求出对应的M，总的时间复杂度为$O(n^2)$。

第二讲 背包问题

参考资料：http://xxjs.ayyz.cn/pack/Index.html

基本理论

问题	给定n个物品的价值v_i和重量w_i，在总重量**不超过**（**恰好为**）C的情况下，求最优的背包方案	
子问题	f[i, j]：只使用前i个物品，在总重量**不超过**（**恰好为**）j的情况下，可以达到的最优的背包方案	
边界条件	“不超过”	$f[0,0] = f[0,1] = \cdots = f[0,C] = 0$
	“恰好为”	$f[0,0] = 0, f[0,1] = \cdots = f[0,C] = -\infty$
递推公式	$f[i,j] = \max\{f[i-1,j], f[i-1,j-w_i] + v_i\}$	
目标函数	f[n, C]	

背包问题的变化

- 01 物品：每个物品只能选择“取”（0）或“不取”（1）
- 无限物品：每个物品可以取任意多个或一个也不取
- 多重物品：每个物品可以取的次数有一个对应的上界
- 多费用物品：每个物品占用的资源有多个
- 分组物品：每个物品都在一个组里，同一个组里的物品只能选择一个（或若干多个）
- 附件物品：附件物品必须在主件物品被选择后才能选取
- 输出最优方案
- 输出第 k 优的解（vijos1412）

下面两个程序都是正确的（01 背包 vs 无限背包）

```
fillchar(f, sizeof(f), 0);

for i := 1 to n do
  for j := m downto v[i] do
    f[j] :=
     max(f[j], f[j-v[i]] + w[i]);

writeln(f[m]);
```

```
fillchar(f, sizeof(f), 0);

for i := 1 to n do
  for j := v[i] to m do
    f[j] :=
      max(f[j], f[j-v[i]] + w[i]);

writeln(f[m]);
```

分数膨胀 (Score Inflation, USACO 3.1)

选手在 USACO 竞赛中得分越高我们就越高兴。我们要让选手在竞赛中能够得到更多的分数，这需要你的帮忙。

竞赛题目可以按题型来分类，每个题型中的题目是无限多的。解决同一类问题的时间和得分都是一样的。你的任务是写一个程序来告诉 USACO 的工作人员，应该从每种题型中选取多少题目，才能让选手在规定的时间内获得最多的分数？

输入包括竞赛总时间M ($1 \le M \le 10000$)（不要担心，你如果不来集训营是不会有这么长的比赛时间的）和题型的总数N ($1 \le N \le 10000$)。接下来的每一行将包括两个整数来描述一种题型：第一个整数表示这类题目的得分（$1 \le$ 得分 ≤ 10000），第二个整数表示这类题目的耗时（$1 \le$ 耗时 ≤ 10000）。

你的程序要帮助我们确定从每类题目中选多少才能在竞赛时间内获得最大的总分。注意每种题型的题目可以不取、只取一道或很多道，请算出选手能够得到的最大分数。

输入格式

- 第一行：两个正整数：M和N，表示竞赛的总时间和题目类型的数目
- 第二行到第$N+1$行：每行两个整数，分别表示每个题型的得分和耗时

输出格式

- 第一行：能够得到的最大分数

样例

inflate.in	inflate.out
300 4 100 60 250 120 120 100 35 20	605 （从第二种类型中取出 4 道，再从第四种类型中取出 3 道）

解题思路

典型的无限背包。

原问题	给定N种题目的时间mt_i和分数pt_i，在总时间不超过M下的最大分数
子问题	$f[i,j]$：只使用前i种类型的题目，在总时间不超过j下的最大分数
边界条件	$f[0,0] = f[0,1] = \cdots = f[0,M] = 0$
递推公式	$f[i,j] = \max\{ f[i-1,j], f[i,j-mt_i] + pt_i \}$
目标函数	$f[N,M]$

货币系统 (Money Systems, USACO 2.3)

奶牛们不但创建了自己的政府，而且建立了自己的货币系统。这套货币是以反常的方式建立起来的，所以货币的面值相当奇快——习惯上，一套货币的面值应该是由 1，5，10，20 或 25，50 和 100 作为的单位组成才对，当然有时也会把 2 加入进来作为调剂。

奶牛想知道用一套货币表示一笔钱可以有多少种不同的方法。比如说，对于$\{1,2,5,10,\dots\}$这套货币系统，可以用来表示18元的方法有18×1，9×2，$8\times2+2\times1$，$3\times5+2+1$，等等。

给定一套货币的面值，写一个程序来计算有多少方法来表示某笔钱。我们保证答案不会超过带符号的 long long 类型（C/C++）或 Int64 类型（Free Pascal）。

输入格式

货币有V $(1\le V\le 25)$种，需要构造的钱的数目是N $(1\le N\le 10000)$

- 第一行：两个正整数：V, N
- 第二行至其余各行：V个表示货币单位面值的整数（每行出现的整数个数是不定的）

输出格式

- 第一行：用V种货币表示N元钱的方法总数

样例

money.in	money.out
3 10 1 2 5	10

解题思路

原问题	给定V种面值的钱币$c_1, c_2, \dots, c_V$，求有多少种方案可以恰好表示N元
子问题	$f[i,j]$：只使用前i种钱币，恰好可以表示出j元的方案总数
边界条件	$f[0,0]=1,\quad f[0,1]=\cdots=f[0,N]=0$
递推公式	$f[i,j]=f[i-1,j]+f[i,j-c_i]$
目标函数	$f[V,N]$

奶牛博览会 (Cow Exhibition, USACO 2003 Nov)

奶牛们想向社会证明其实他们是聪明而且风趣的。为此，贝西筹备了一个专门展览奶牛的博览会，她已经对N $(1 \le N \le 100)$头奶牛进行了面试，确定了每头奶牛的聪明指数S_i $(-1000 \le S_i \le 1000)$和风趣指数F_i $(-1000 \le F_i \le 1000)$。

贝西需要决定让哪些奶牛上博览会。设总聪明指数TS为各奶牛聪明指数S_i的和，总风趣指数TF为各奶牛风趣指数F_i的和。贝西想使TS与TF的和最大，同时她希望这两个值不要小于零（因为她要证明奶牛们是出色的，负的TS或TF会造成负面的效果）。请帮助贝西求出TS与TF在非负条件下的最大和。

输入格式

- 第一行：一个整数N，表示奶牛的数量
- 第二行到第$N+1$行：每行两个用空格分开的整数：S_i和F_i，分别代表每头奶牛的聪明指数和风趣指数

输出格式

- 第一行：单独一个整数，表示在TS和TF非负条件下的最大和。如果无解，则输出 0

样例

smrtfun.in	smrtfun.out
5 -5 7 8 -6 6 -3 2 1 -8 -5	8 （贝西可以选择 1，3，4 号奶牛，此时$TS=-5+6+2=3$，$TF=7-3+1=5$，总和为8。注意如果加入 2 号奶牛可以使总和提升到10，不过TF变负了，而这是不允许的。）

解题思路

01 背包，将S_i当成容量，将S_i+F_i当成价值

太空电梯 (Space Elevator, USACO 2005 Mar)

奶牛们准备升空了!为了进入轨道,他们计划建造一座由各种材料堆成的巨型太空电梯。她们拥有K ($1 \le K \le 400$)种型号不同的材料， i型材料的单位长度为h_i ($1 \le h_i \le 100$)，数量为c_i ($1 \le c_i \le 10$)，由于宇宙射线可能危及材料安全，因此规定堆建材料时，每块i型材料的高度都不能超过a_i ($1 \le a_i \le 40000$)。

请帮助奶牛们用这些材料堆出尽可能高的太空电梯。

输入格式

- 第一行：一个整数K
- 第二行到第$K+1$行：每行三个整数：h_i，a_i，c_i。第$i+1$行描述的是第i块材料的性质

输出格式

- 第一行：一个整数H，表示电梯的最高高度

样例

elevator.in	elevator.out
3 7 40 3 5 23 8 2 52 6	48 （自底向上：先取 3 块 2 型材料，再取 2 块 1 型材料，再取 6 块 3 型材料。但如果放 4 块 2 型材料，和 3 块 1 型材料是不合法的，因为 1 型材料的海拔超过了40）

解题思路

背包问题的变形。首先预处理数据，把所有材料按照a_i排序，并把每种材料拆成 01 物品，所以可以设$a_1 \le a_2 \le \cdots \le a_n$，$c_i \equiv 1$（$n = \sum_{i=1}^{K} c_i$）

原问题	给定K种材料，求所能建造的最高高度
子问题	f[i, j]：只使用前i个材料，是否可以达到高度j
边界条件	$f[0,0] = \text{True}, \quad f[0,1] = \cdots = f[0,\infty] = \text{False}$
递推公式	$f[i,j] = \begin{cases} f[i-1,j] \text{ or } f[i-1,j-h_i], & j \le a_i \\ \text{False}, & j > a_i \end{cases}$
目标函数	$\max_{0 \le j \le \infty} \{j \mid f[n,j] = \text{True}\}$

牛奶量取 (Milk Measuring, USACO 5.3)

农夫约翰要量取Q ($1 \le Q \le 20000$)夸脱的上等牛奶，把它装在瓶子里面送到一个客户的手上，瓶子里的牛奶一定要和客户订购的那样不多也不少。

约翰向来节约。为了从他的巨型的牛奶池中量出Q夸脱的牛奶，他必须去奶牛五金店里买一些称量用的提桶。由于每个桶的价格是一样的，你的任务就是帮约翰计算为了量出Q夸脱的牛奶，最少要买哪一些提桶。由于约翰必须把这些东西搬回家，所以对于任意两个最少数量的候选方案，他会偏向“更小”一组，即把两组解按升序排列，比较第一个桶的容积，选择容积小的一组。如果排在第一的两个桶容积相同，就比较第二个，一直继续这样的工作，直到找到两个桶的容积不一致为止，例如相比{3,5,7,100}，约翰更欢迎{3,6,7,8}。

量牛奶时，约翰会用池中的牛奶把提桶装满，然后倒进瓶子，他决不会把瓶子里的牛奶再倒出来或者把已经在一个桶里的牛奶倒到别的桶里去。如果约翰有一个容积为1的桶，他可以只用这个桶量出所有的份量，但如果他遇到的是其它的组合就不会这么方便了。

我们保证所有的测试数据至少有一个解，试确定购买的最优方案。

输入格式

- 第一行：单个整数Q
- 第二行：单个整数P ($1 \le P \le 100$)表示商店里提桶的数量
- 第三行到第$P+2$行：每行只有一个整数，表示某个提桶的容积 ($1 \le$ 容积 ≤ 10000)

输出格式

输出文件只有一行，由空格分开的整数组成，包括：

- 为了量出指定的夸脱，需要购买的提桶的最少数量，其次是
- 一个以升序排列的序列，表示需要购买的每个提桶的容积

样例

milk4.in	milk4.out
16 3 3 5 7	2 3 5

解题思路

无限背包，不但要求最优值，还要求最优解，编程难度较高。

第三讲 动态规划选讲

动态规划的英文为 Dynamic Programming（简称 DP），此处的 Programming，指的不是编程求解问题，而是指“use of a tabular solution method”，即用表格来解决问题。

问题：哪一类问题属于“动态规划问题”？

动态规划是一种**解题方法**，而不是一类特定问题的名称。在这一点动态规划与计算几何、图论不同。图论的研究对象是图，而动态规划没有明确的研究对象。有时为了说话方便，我们说“某问题是动态规划问题”一般就是指可以用动态规划解决这个问题。另外，我们一般使用动态规划解决**最优化问题**，有些书将某些经典的计数问题也称作动态规划问题，就没有什么必要了。

问题：动态规划和分治法的区别和联系各是什么？

动态规划和分治法的基本思想都源自于归纳（Induction）。所不同的是，分治法将原来的问题分成彼此独立的子问题，而动态规划面对的往往是子问题相互之间不是独立的、一些子问题是有重叠的。

问题：动态规划的解题步骤是怎么样的？

用动态规划解题可分为以下四步：

1) Characterize the structure of an optimal solution. 找出子问题
2) Recursively define the value of an optimal solution. 写出递推方程
3) Compute the value of an optimal solution in a bottom-up fashion. 自底向上地计算
4) Construct an optimal solution from computed information. 构造最优解（这步往往省略）

问题：动态规划等于递推吗？

动态规划不等于递推。诚如前面所言，动态规划是一种**解题方法**，而递推是相对与递归来说的，他们指的都是**实现算法思想的方法**。事实上，任何可以用递推实现的程序，都可以用递归+记忆来改写，反之也是如此。例如，计算斐波那契数列$f_n = f_{n-1} + f_{n-2}$就有两种方法：一是**自顶向下**（Top-Down）地利用递归函数计算，二是利用初值条件$f_0 = f_1 = 1$**自底向上**（Bottom-Up）地计算。这两种方法都能够在动态规划问题中得到应用，可以说是各有所长。如果在实现程序的时候使用递归，一般把这种实现方法称为**记忆化搜索**。

问题：动态规划有哪些要素？

- 状态转移方程（又称递推方程）
- 目标函数（又称目标状态）
- 边界条件（又称初始条件）

问题：适用动态规划的问题需要满足哪些条件？

1. 最优化原理——“最优策略的子策略也是最优策略”
也就是说一个问题的最优决策只取决于其子问题的最优决策，子问题的非最优决策对问题的求解没有影响。
例：最短路径。假设从 A 到 B 的最优解是先从 A 到 C 再到 B，则 A 到 B 的最优决策一定包含着从 A 到 C 的最优决策，从 A 到 C 的其他决策和 A 到 B 的最优决策没有关系。
2. 无后效性原则——“未来与过去无关，只和现在有关”
某**阶段**的**状态**一旦确定，则此后过程的演变不受此前各状态及**决策**的影响。

抓苹果 (Apple Catching, USACO 2004 Nov)

告诉你一个鲜为人知的事实：奶牛喜欢吃苹果。农夫约翰有编号为1和2的两棵苹果树，每棵树上都结满了苹果。贝西够不着树上的苹果，所以她必须等它们落到地上。然而，她必须在苹果落地之前接住它们（苹果掉在地上就被摔坏了，没有人爱吃坏苹果）。贝西吃东西的速度很快，可以在几秒钟内吃完一只苹果。

在每一分钟，两棵苹果树中的一棵会掉下一只苹果。贝西训练有素，只要她站在树下就能接到掉下来的苹果。尽管贝西可以快速地在两棵树之间行走（远不需要一分钟），但是她每一分钟只能站在一棵树下。此外，由于奶牛们的运动训练不足，所以她不高兴在两棵树之间无止尽地走来走去，就算这样就失去一些苹果也一样。

每分钟掉落一个苹果，一直会持续T $(1 \leq T \leq 1000)$分钟，贝西最多愿意来回奔波W $(1 \leq W \leq 30)$次。给出每分钟苹果掉落地情况，确定贝西可以抓到的最大苹果数量。贝西一开始在 1 号树下。

输入格式

- 第一行：两个用空格分开的整数：T和W
- 第二行到第$T+1$行：表示在这一分钟内哪棵树上的苹果将掉落

输出格式

- 第一行：贝西在移动不超过W次的条件下能够抓到的最大苹果数量

样例

bcatch.in	bcatch.out
7 2 2 1 1 2 2 1 1 （一共有七个苹果，第一个从 2 号树上掉落，其次是 1 号树上掉落两个，接下来是 2 号树上掉落两个，最后 1 号树上落下最后两个，而贝西只想移动两次）	6 （贝西可以抓住六个苹果：首先待在 1 号树下得到两个，再移动到 2 号树下抓住两个，再返回 1 号树，抓住最后两个）

解题思路

动态规划，第一维记录时间，第二维记录已经移动的次数（从已经移动的次数可以推出当前所在的位置）

最廉回文 (Cheapest Palindrome, USACO 2007 Open)

追踪每头奶牛的去向是一件棘手的任务，为此农夫约翰安装了一套自动系统。他在每头牛身上安装了一个电子身份标签，当奶牛通过扫描器的时候，系统可以读取奶牛的身份信息。目前，每个身份都是由一个字符串组成的，长度为M $(1 \leq M \leq 2000)$，所有的字符都取自小写的罗马字母。

奶牛们都是顽皮的动物，有时她们会在通过扫描器的时候倒着走，这样一个原来身份为 abcb 的奶牛就可能有两个不同的身份了（abcb 和 bcba），而如果身份是 abcba 的话就不会有这个问题了。

约翰想改变奶牛们的身份，使他们不管怎么走读起来都一样。比如说，abcb 可以在最后加个 a，变成回文 abcba；也可以在前面加上 bcb，变成回文 bcbabcb；或者去除字母 a，保留的 bcb 也是一条回文。总之，约翰可以在任意位置删除或插入一些字符使原字符串变成回文。

不巧的是，身份标签是电子做的，每增加或删除一个字母都要付出相应的费用（$0 \leq$ 代价 ≤ 10000）。给定一头奶牛的身份标签和增加或删除相关字母的费用，找出把原来字符串变成回文的最小费用。注意空字符串也是回文。

输入格式

- 第一行：两个用空格分开的整数：N和M
- 第二行：一个长度恰好为M的字符串，代表初始的身份标签
- 第三行到第$N+2$行：每行为一个用空格分开的三元组：其中包括一个字符和两个整数，分别表示增加或删除这个字符的费用

输出格式

- 第一行：只有一个整数，表示改造这个身份标签的最小费用

样例

cheappal.in	cheappal.out
3 4 abcb a 1000 1100 b 350 700 c 200 800	900 （如果在最后插入一个 a，得到 abcba，代价为1000；如果删除第一个 a，得到 bcb，代价为1100；如果在字符串的开头插入 bcb，代价为$350+200+350=900$，这才是最优的做法）

解题思路

原问题	给定字符串S，求变成回文的最小代价
子问题	$f[i,j]$：从i到j的子串$S_{\{i..j\}}$变成回文的最小代价
边界条件	$f[i,i]=0$
递推公式	$f[i,j]=\min\begin{cases} f[i+1,j]+add[S_i] \\ f[i+1,j]+del[S_i] \\ f[i,j-1]+add[S_j] \\ f[i,j-1]+del[S_j] \\ f[i+1,j-1]，若S_i=S_j \end{cases}$
目标函数	$f[1,N]$

麦香牛块 (Beef McNuggets, USACO 4.1)

农夫布朗的奶牛们正在进行斗争，因为它们听说麦当劳正在考虑引进一种新产品：麦香牛块。奶牛们正在想尽一切办法让这种可怕的设想泡汤。

奶牛们进行斗争的策略之一是“劣质包装”。“看!”，奶牛们说，“如果你用能装 3 块、6 块或者 10 块的三种盒子包装麦香牛块，你就不可能满足想买 1、2、4、5、7、8、11、14 或者 17 块麦香牛块的顾客了。劣质的包装意味着劣质的产品。”

你的任务是帮助这些奶牛。给出包装盒的种类数N ($1 \le N \le 10$)和N个代表不同包装盒容量的整数i ($1 \le i \le 256$)，输出不能用上述盒子（数量无限）包装的麦香牛块的最大块数。如果所有购买块数都能满足或者无法包装的块数没有上限，则输出 0。

无法包装的最大块数（倘若它存在）不会超过2×10^9。

输入格式

- 第一行：包装盒的种类N
- 第二行到第$N+1$行：每种包装盒容纳麦香牛块的个数

输出格式

- 第一行：单独一个整数，表示不能包装的最大块数，如果所有包装都能满足或者无法包装的块数有无限个，输出 0

样例

nuggets.in	nuggets.out
3 3 6 10	17

解题思路

题目本身的解法很简单，但是蕴含了一定的数论知识。

道路重建 (Rebuilding Roads, USACO Feb 2002)

在一场严重的地震之后，奶牛们重建了农夫约翰的牧场。牧场里有N个牛棚，$1 \le N \le 150$，用数字1到N表示。由于奶牛们的时间不够重建更多的道路，所以任意两个牛棚之间有且只有一条道路，也就是说，牧场的交通线路可以用一棵树来表示。

约翰想知道再来一场地震的话将造成多大的损失。他尤其想知道，至少几条道路被摧毁，才会让某个大小为P $(1 \le P \le N)$的牛棚所构成的子树和其他牛棚断开。

输入格式

- 第一行：两个正整数：N和P
- 第二行到第N行：共$N-1$行，每行有两个整数I和J，表示结点I是结点J的父亲。

输出格式

- 第一行：单独一个整数，表示至少要破坏几条道路，才会让一个大小为P的子树和其他结点断开

样例

roads.in	roads.out
11 6 1 2 1 3 1 4 1 5 2 6 2 7 2 8 4 9 4 10 4 11	2 （被断开的子树是(1,2,3,6,7,8)，而需要被破坏的道路是(1,4)和(1,5)。）

解题思路

树形动态规划。设f[u, k]为以u为根的树，分离出一个大小为k，并且包括u的子树所需要删除的最少道路。再设$f^{(i)}[u,k]$为以u为根，只看u的前i个儿子分支的树，分离出一个大小为k，并且包括u的子树所需要删除的最少道路。设u的儿子集合为S(u)，u的儿子个数为#S(u)。则有

$$f[u,k] = f^{(\#S(u))}[u,k]$$

$$f^{(0)}[u,1]=0$$

$$f^{(i)}[u,k]=\min_{1\le j\le k-1}\begin{cases}f^{(i-1)}[u,k]+1\\ f^{(i-1)}[u,j]+f[u_i,k-j]\end{cases}$$

上式中u_i是u的第i个儿子。

求解的目标是

$$obj=\min\begin{cases}f[u,p], & u=root\\ f[u,p]+1, & u\neq root\end{cases}$$

实现时，应用“左邻居右儿子”的数据结构来表示树。

第四讲 最小生成树问题

建造道路 (Building Roads, USACO 2007 Dec)

约翰刚得到了几片新农场！他打算把这些农场连成一片，而这些农场之间已经存在一些道理了。已知共有N $(1 \le N \le 1000)$片农场，方便起见以数字1到N来编号。每片农场在平面上有个坐标(X_i, Y_i) $(1 \le X_i, Y_i \le 1{,}000{,}000)$。设已存在的道路有M $(1 \le M \le 1000)$条，请帮助约翰计算最少修建多长的道路，才能把这些农场连成一片？

输入格式

- 第一行：两个用空格分开的整数：N和M
- 第二行到$N+1$行：两个用空格分开的整数：X_i和Y_i
- 第$N+2$行到$N+M+2$行：两个用空格分开的整数：i和j，代表已存在从i到j的道路

输出格式

- 第一行：需要修建的最短总长度，保留两位小数，保证结果不会超过 64 位的浮点数

样例

roads.in	roads.out
4 1 1 1 3 1 2 3 4 3 1 4	4.00 （在 1 和 2 之间，3 和 4 之间修建道路，每条道路的长度为 2.00，故总长为 4.00）

解题思路

典型的最小生成树问题，由于是稠密图，所以使用 Prim 算法为佳。事实上，还有更好的$O(n \log n + m)$算法：The idea is basically the edges that could potentially be in the minimum spanning tree must belong to what's known as the Delaunay triangulation, which has O(n) edges. We can find the Delaunary triangulation in O(nlogn) time and apply a fast version of Kruskal's algorithm for sparse graphs to get the desired runtime.

安慰奶牛 (Cheering up the Cows, USACO 2008 Nov)

农夫约翰变得非常懒，他不想再继续维护供奶牛通行的道路了。这些道路被用来连接 $N\ (5 \le N \le 10{,}000)$ 个牧场，牧场的编号为 1 到 N。每个牧场里都住着一头奶牛。一共有 $P\ (N-1 \le P \le 100{,}000)$ 条道路，约翰计划除去中其中尽可能多的道路，但是还保持牧场之间的连通性。

每条道路都是双向的，记第 j 条连接了牧场 S_j 和 E_j $(1 \le S_j, E_j \le N, S_j \ne E_j)$，走完它需要 $L_j\ (0 \le L_j \le 1{,}000)$ 的时间。任意两个牧场之间最多只有一条道路。

奶牛们非常伤心，因为她们的交通系统被削减了。你需要到每头奶牛的住处去安慰她们。每次到第 i 个牧场的时候（即使你已经去了不止一次），必须花去 $C_i\ (1 \le C_i \le 1{,}000)$ 的时间和奶牛交谈。

你可以选择在从一个牧场出发，一旦奶牛们都从悲伤中缓过神来之后，必须回到这个牧场过夜。注意，当你早上准备出发和晚上回去睡觉的时候，你都需要先和这个牧场里的奶牛交谈一次。

假设约翰听从你的建议来决定保留哪些道路，请算出安慰所有奶牛需要的最少时间。

输入格式

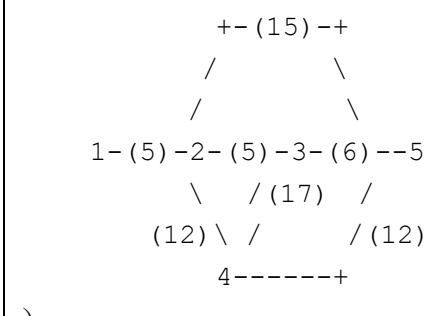
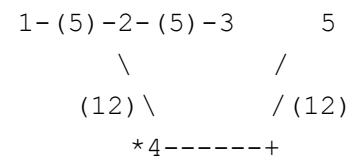
- 第一行：两个用空格分开的整数： N 和 P
- 第二行到 $N+1$ 行：第 $i+1$ 行包括一个整数 C_i
- 第 $N+2$ 行到 $N+P+1$ 行：第 $N+j+1$ 行包括三个用空格分开的整数 S_j，E_j 和 L_j

输出格式

- 第一行：一个整数，表示安慰所有奶牛需要的时间

样例

cheer.in	cheer.out
5 7 10 10 20 6 30 1 2 5 2 3 5 2 4 12 3 4 17 2 5 15	176

3 5 6 4 5 12 （如下图：  ） 	（保留以下路径：  选择在 4 号农场出发，依次访问 4，5，4，2，3，2，1，2，4 号农场，总时间为 176）

解题思路

Given a choice of the edges and starting point, which form a rooted tree, the optimal path is an Euler tour from the root. That is, we visit nodes in order:

```
visit(node v):
    go to v
    for c in children of v:
        visit(c)
        go to v
```

A node with K children appears in a tree's Euler tour K+1 times. For all nodes except the root, K+1 is also the degree of the node (whereas a root with K children has degree K). Thus, a node of degree D is visited D times, with the root being visited once more. Also, each edge is traversed twice: once going down and once going up.

Our total visiting time, for a tree rooted at vertex R with edges of length L_i and vertices of degree D_j and cost C_j is 2(L_1 + ... + L_N-1) + (D_1*C_1 + ... + D_N*C_N) + C_R.

We can now simplify the problem by computing the cheapest tree and picking the smallest C_R independently. We also note that counting a vertex j D_j times is counting it once for every edge connected to it. Equivalently, this is attributing to each edge i, the costs of the vertices at either end, S_i and E_i.

If we relabel an edge from S to E with length L with a weight of 2*L + C_E + C_S, the cost of the tree is the sum of the edges. This is exactly the minimum-cost spanning tree problem, which can be solved in O(P log P) time with Kruskal's or Prim's algorithm.

地震 (Earthquake, USACO 2001 Open)

一场地震把约翰家的农场全部摧毁了，而约翰决心重建家园。约翰已经重建了N ($1 \le N \le 400$)个农场，现在他希望能修建一些道路把它们连接起来。

在研究了地形之后，约翰发现可以修建的双向道路有M ($1 \le M \le 10000$)条，由于他的资金有限，所以他希望花的钱越少越好。

碰巧的是，奶牛们最近也成立一个工程队，专门从事修复被地震摧毁的道路。而然，奶牛们很有经济头脑，如果修路无钱可赚，它们是不会干的。

奶牛们关注的是挣钱的速度，即总利润和总施工时间的比值（又称为单位利润）。设约翰和奶牛达成的修复费用为F ($1 \le F \le 2{,}000{,}000{,}000$)。每条道路都有一个施工的时间t ($1 \le t \le 2{,}000{,}000{,}000$)，和一个建造的成本c ($1 \le c \le 2{,}000{,}000{,}000$)。连接两个相同的农场的道路可能有多条。保证所有的农场都有可以连通的道路，但是也有可能这些道路的建造成本会超过F。

请帮助奶牛们选择修复哪些道路，才能使单位利润最大？

输入格式

- 第一行：三个整数： N，M和F
- 第二行到$M+1$行：每行四个整数，分别为i，j，c，t，表示这条道路连接了i和j，施工时间为t，成本为c。

输出格式

- 第一行：一个保留四位小数的浮点数，表示奶牛们能挣到的最大单位利润，如果奶牛们无钱可赚，则输出 0.0000

样例

quake.in	quake.out
5 5 100 1 2 20 5 1 3 20 5 1 4 20 5 1 5 20 5 2 3 23 1	1.0625 （奶牛们可以选择连通最后四条道路，则总时间为 16，总成本为 83，所以单位利润为 $17/16 = 1.0625$）

解题思路

参数化搜索。每次二分地判断一个问题：“是否存在一个单位利润超过α的解？”这个问题等价于询问：“是否存在一棵生成树T，使得下面的不等式成立？”

$$T\text{ 的利润} - \alpha \cdot T\text{ 的时间} \geq 0$$

即

$$\left(F - \sum_{e \in T} c(e)\right) - \alpha \cdot \sum_{e \in T} t(e) \geq 0$$

等价于

$$\sum_{e \in T} (c(e) + \alpha \cdot t(e)) \leq F$$

令$w(e) = c(e) + \alpha \cdot t(e)$，原问题等价于求解权值为w的最小生成树。

第五讲 最短路径问题

我们所熟知的所有求最短路径的算法，基本上都是一种标号算法（Labeling Algorithm），可以把它们分成两个大类：

- 标号固定法（Label Setting，简称 LS）——如 Dijkstra。特点：“稳扎稳打”，每步可以确定到一个结点的最短距离
- 标号修定法（Label Correcting，简称 LC）——如 Bellman-Ford。特点：“逐步求精”，到每个点的距离会逐渐收敛。

Dijkstra+Heap 是首选的推荐算法。实现上可以使用一个小技巧，避免调整 Heap 里的数据，请参考《道路翻新》一题的程序。

注：在 OI 界盛传的 SPFA 算法，最早发表于 1994 年 4 月的西南交通大学学报:《关于最短路径的 SPFA 快速算法》（作者段凡丁），然而我认为这个算法的效率并不如一些网友宣称的那样高效，作为比较，可以实现一个 SPFA 版本的《道路翻新》，而且原文献中对算法复杂度的分析也是错误的。请读者自行甄别比较。

原文献中，声称 SPFA 算法的平均复杂度为$O(k|E|)$，其中$|E|$为边集个数，k为平均每条边的进队次数，有人声称$k \approx 2$，这是没有任何理论根据的，也是没有数据可以证实的。事实上，SPFA 算法的最坏情况的复杂度是$O(|V| \cdot |E|)$，远远高于 Dijkstra 算法。

http://dantvt.spaces.live.com/blog/cns!D87988A6CAC0A480!775.entry（这篇文章想说明 SPFA 比 Dijkstra 快，事实上，对结点个数$N \leq 1000$的数据是没有任何说服力的，要比较速度，N至少应取5000以上）
http://dantvt.spaces.live.com/blog/cns!D87988A6CAC0A480!790.entry （证明了在稠密图上 SPFA 不如 Dijkstra，其实 Dijkstra 在稀疏图上也应该很快，只是大部分的实现不正确而已）

http://hi.baidu.com/winterlegend/blog/item/8d9b16d801c861e038012f5e.html（此文以《道路翻新》的数据推翻了 SPFA 在稀疏图上有优势的论断）
注：《道路翻新》的数据规模在$100,000$左右，比较能够说明问题

道路翻新 (Revamping Trails, USACO 2009 Feb)

农夫约翰每天都要检查一下牛棚里的牛。他需要从编号为1牛棚出发，通过最近的道路走到编号为N的牛棚。现假设农场上一共有N个牛棚，为方便起见，用1到N的数字来编号，它们由M ($1 \le M \le 50000$)条双向道路连接，保证1号牛棚一定会与N号牛棚相连。每条道路i连接的牛棚用$P1_i$和$P2_i$ ($1 \le P1_i, P2_i \le N$)表示，通行消耗的时间用T_i ($1 \le T_i \le 1000000$)来表示。

现在农夫约翰想翻新一些道路来减少每天花在路上的时间。但他最多只能翻新K ($1 \le K \le 20$)条道路，翻新后的道路的通行时间将变成0。请帮助约翰选择最优的翻新方案使得从1号牛棚到N号牛棚的时间最短。

输入格式

- 第一行：包括三个数：N，M和K，彼此用空格分开。
- 第二行到$M+1$行：在第$i+1$行将会告诉你第i条道路的信息：$P1_i$，$P2_i$和T_i，彼此用空格分开。

输出格式

- 第一行：不超过K条道路被翻新后的最短路径长度。

样例

revamp.in	revamp.out
4 4 1 1 2 10 2 4 10 1 3 1 3 4 100	1 （选择翻新$3 \to 4$，时间从100减到0，最短路径为$1 \to 3 \to 4$，因而总用时为1）

解题思路

Define the K-modified length of a path to be the sum of the lengths of all but its K longest edges. For example, the 0-modified length of a path is just its length. One way of rephrasing the problem is that we need to calculate the shortest K-modified length among all paths from vertex 1 to vertex N. [NB: from the sample input and output, it is clear that this is NOT necessarily equal to the K-modified length of the regular shortest path from 1 to N].
From the given graph (V,E), construct a new graph as follows. The vertex set will be V x {0,1,2,...,K}. For each edge e=(v1,v2) of length L, add 2K+1 edges as follows:
* add an edge from (v1,k) to (v2,k) of length L for all k in {0,1,2,...,K}

* add an edge from (v1,k) to (v2,k+1) of length 0 for all k in {0,1,2,...,K-1}
Also add edges from (v,k) to (v,k+1) of length 0 for all v and for all k in {0,1,2,...,K-1}. It is left as an exercise to the reader to convince him/herself that the length of the shortest path from (1,0) to (N,K) in this modified graph is equal to the shortest K-modified length among paths from 1 to N in the original graph.
This new graph has O(VK) vertices and O(EK) edges, so since Dijkstra's algorithm is O(e log v), the above algorithm runs in O(E K log V), which should run in time..

道路障碍 (Roadblocks, USACO 2006 Nov)

贝西搬到了一个小农场，有时候，她会回来看看她的朋友们。由于路上风景不错，她不想走得太快，于是贝西决定不走最短路径，而是走次短（即第二短）的路径（假定次短路径一定是存在的）。

村里一共有R $(1 \le R \le 100000)$条双向道路，N $(1 \le N \le 5000)$个结点（从1到N编号）。贝西的起点是1号点，目的地（就是她朋友的家）在N号点。

次短路径的有些边可以和最短路径重复，而且也允许存在圈，即次短路径上允许重复多次通过一些点或边。我们要求次短路径要严格大于最短路径。比如说，如果有两条不同的路径都是最短路径，那么他们都不能算是次短路径，次短路径一定要比它们长。

输入格式

- 第一行：两个用空格分开的整数：N和R
- 第二行到$R+1$行：每一行有三个整数：A，B和D，表示一条道路连接了A点和B点，长度为D $(1 \le D \le 5000)$

输出格式

- 单独一行：从1号点到N号点的第二短的路径。

样例

block.in	block.out
4 4 1 2 100 2 4 200 2 3 250 3 4 100	450 （最优路线：1→2→4，长度为$100+200=300$；次优路线 1→2→3→4，长度为$100+250+100=450$）

解题思路

Define d(p,k) to be the length of the kth shortest path to vertex p. Then note that for this algorithm, the only relavent distances we have to consider are d(p,1) and d(p,2).

The sketch for the proof of correctness would be if d(i,3) was used for one of the paths, then the two paths with the path leading up to d(i,3) replaced by the paths leading up to d(i,1) and d(i,2) respectively would have lesser costs, contradicting with the claim the resulting path is one of the two shortest paths.

Notice that the state with the minimum distance to the source cannot have its distance reduced as all paths have positive lengths. Therefore, we could 'expand' this vertex by adding new states of the results of travelling to its neighbors from this vertex. By the above claim, it's not necessary to expand from the same vertex more than twice.

Therefore, we could store all the the distances in a priority queue, as long as there are elements in the queue, take out the minmum, if its vertex has not been 'expanded' twice, expand it and add all the new distances to the priority queue.

Since at most 2V expansions can be done, at most 2E nodes can be in the queue at a time, resulting in a runtime of O(ElogE) where E is the number of edges.

奶牛慢跑 (Cow Jogging, USACO 2008 Mar)

贝西尝到了懒惰的恶果——为了减肥，她不得不决定每周花几次时间在牛棚和池塘之间慢跑。但贝西并不想太累，所以她打算只跑从牛棚到池塘的下坡路，然后再慢慢地从池塘走回牛棚。

同时，贝西也不想跑得太远，所以她只想沿着通向池塘的最短路径跑步。在牧场里，每条道路连接了两个结点（这些结点的编号为1到N，$1 \le N \le 1000$）。另外，如果$X > Y$，说明结点X的地势要高于Y，所以下坡的道路是从X通向Y的，贝西所在牛棚的编号为N（最高点），池塘的编号为1（最低点）。

而然，一周之后，贝西对单调的路线厌倦了，她希望每天可以跑不同的路线，比如说，最好能有K $(1 \le K \le 100)$种不同的选择。为了不至于跑得太累，她希望这K条路径是从牛棚到池塘的最短的K条路径。

请帮助贝西算算她的运动量，即找出网络里最短的K条路径的长度。假设每条道路用(X_i, Y_i, D_i)表示，其中$1 \le Y_i < X_i \le N$，表示这条道路从X_i出发到Y_i，其长度为D_i $(1 \le D_i \le 1{,}000{,}000)$。

输入格式

- 第一行：三个用空格分开的整数：N，M和K
- 第二行到$M+1$行：第$i+1$行用三个整数描述了第i条道路的信息：X_i，Y_i和D_i。

输出格式

- 第一行到K行：在第i行输出第i短的路径长度，如果第i短的路径不存在，则输出-1。如果几条最短路径的长度相同，则应该重复输出相同的长度。

样例

block.in	block.out
5 8 7 5 4 1 5 3 1 5 2 1 5 1 1 4 3 4 3 1 1 3 2 1 2 1 1	1 2 2 3 6 7 -1 （六条路径按长度排序为：5→1，5→3→1，5→2→1，5→3→2→1，5→4→3→2→1）

解题思路

We are given an acyclic graph and we wish to find the K $\leqslant$ 100 shortest paths. Since the graph is acyclic with a natural ordering to the vertices, Dynamic Programming is a good first approach.

Suppose that we knew the K shortest paths starting at each vertex i for i $\geqslant$ l and going downhill to the barn at vertex N. Then, to compute the K shortest such paths beginning at vertex i-1, we consider all outgoing edges from vertex i-1. Suppose that vertex i-1 has d outgoing edges. The ends of these d edges are all larger than i-1 and we have computed the K shortest paths from each of these. We can consider prepending vertex i-1 to all of these d*K paths. The best K paths starting at vertex i-1 will come from these d*K paths. We can efficiently compute a sorted list of the K best paths starting at vertex i-1 by merging these d lists of K items. This requires d*K time at each vertex. Summing over all vertices yields O(MK) time. With M=10,000 and K=100, this will be around 1,000,000 operations, which works.

第六讲　广度优先遍历

青铜莲花池 (Bronze Lilypad Pond, USACO 2007 Feb)

为了让奶牛们娱乐和锻炼，农夫约翰建造了一个美丽的池塘。这个长方形的池子被分成了M行N列个方格（$1 \le M, N \le 30$）。一些格子是坚固得令人惊讶的莲花，还有一些格子是岩石，其余的只是美丽、纯净、湛蓝的水。

贝西正在练习芭蕾舞，她站在一朵莲花上，想跳到另一朵莲花上去，她只能从一朵莲花跳到另一朵莲花上，既不能跳到水里，也不能跳到岩石上。

贝西的舞步很像象棋中的马步：每次总是先横向移动M1 ($1 \le M1 \le 30$)格，再纵向移动M2 ($1 \le M2 \le 30, M1 \ne M2$)格，或先纵向移动M1格，再横向移动M2格。最多时，贝西会有八个移动方向可供选择。

给定池塘的布局和贝西的跳跃长度，请计算贝西从起点出发，到达目的地的最小步数，我们保证输入数据中的目的地一定是可达的。

输入格式

- 第一行：四个用空格分开的整数：M，N，M1和M2
- 第二行到$M+1$行：第$i+1$行有N个用空格分开的整数，描述了池塘第i行的状态：0 为水，1 为莲花，2 为岩石，3 为贝西所在的起点，4 为贝西想去的终点。

输出格式

- 第一行：从起点到终点的最少步数

样例

bronlily.in	bronlily.out
4 5 1 2 1 0 1 0 1 3 0 2 0 4 0 1 2 0 0 0 0 0 1 0 （贝西从第二行的最左边出发，目标是第二行的最右边）	2 （贝西先跳到第一行第三列的莲花上，再跳到终点，需要两步）

解题思路

跳马问题的简单变形，广度优先搜索或 Dijkstra 算法。

白银莲花池 (Silver Lilypad Pond, USACO 2007 Feb)

为了让奶牛们娱乐和锻炼，农夫约翰建造了一个美丽的池塘。这个长方形的池子被分成了M行N列个方格（$1 \le M, N \le 30$）。一些格子是坚固得令人惊讶的莲花，还有一些格子是岩石，其余的只是美丽、纯净、湛蓝的水。

贝西正在练习芭蕾舞，她站在一朵莲花上，想跳到另一朵莲花上去，她只能从一朵莲花跳到另一朵莲花上，既不能跳到水里，也不能跳到岩石上。

贝西的舞步很像象棋中的马步：每次总是先横向移动一格，再纵向移动两格，或先纵向移动两格，再横向移动一格。最多时，贝西会有八个移动方向可供选择。

约翰一直在观看贝西的芭蕾练习，发现她有时候不能跳到终点，因为中间缺了一些荷叶。于是他想要添加几朵莲花来帮助贝西完成任务。一贯节俭的约翰只想添加最少数量的莲花。当然，莲花不能放在石头上。

请帮助约翰确定必须要添加的莲花的最少数量。在添加莲花最少的基础上，确定贝西从起点跳到目标需要的最少步数。最后，确定满足添加的莲花数量最少时，步数最少的路径条数。

输入格式

- 第一行：两个用空格分开的整数：M和N
- 第二行到$M+1$行：第$i+1$行有N个用空格分开的整数，描述了池塘第i行的状态：0 为水，1 为莲花，2 为岩石，3 为贝西所在的起点，4 为贝西想去的终点。

输出格式

- 第一行：一个整数：需要添加的莲花的最少数目；如果无解，则输出-1
- 第二行：一个整数：在添加莲花最少的基础上，贝西从起点跳到终点需要的最少步数；如果第一行是-1，不输出这行
- 第三行：一个整数：在添加莲花最少的基础上，步数等于第二行输出的路径条数；如果第一行是-1，不输出这行

样例

silvlily.in	silvlily.out
4 8 0 0 0 1 0 0 0 0 0 0 0 0 0 2 0 1 0 0 0 0 0 4 0 0 3 0 0 0 0 0 1 0	2 6 2 （最少要加两朵莲花，位置如 x 所示： 0 0 0 1 0 0 0 0　　0 0 0 1 0 0 0 0 0 x 0 0 0 2 0 1　　0 0 0 0 0 2 0 1

（池塘分成四行八列，贝西的起点在第四行第一列，想去的终点在第三行第六列，池塘里一共有五朵莲花和一块石头）	0000x400 00x0x400 30000010 30000010 贝西至少要跳六步，两种不同的跳法如下： 000C0000 000C0000 0B00020F 0000020F 0000DG00 00B0DG00 A00000E0 A00000E0 ）

解题思路

We can treat this as a shortest path problem. At each square we keep track of how many lilypads we have to place to get there and how many jumps it took. We first want to minimize the lilypads and then minimize the jumps (moving onto a lilypad only takes one jump), while moving onto a blank square takes one lilypad + one jump.

We use BFS to update this information as we traverse the various routes. If a jump brings us to a square taking the same number of lilypads and jumps as it already says on the square, we can add the number of ways to that square. If we did better, we replace the number of ways to that square and add to queue. If we did worse, we don't update it.

黄金莲花池 (Lilypad Pond, USACO 2007 Feb)

为了让奶牛们娱乐和锻炼，农夫约翰建造了一个美丽的池塘。这个长方形的池子被分成了M行N列个方格（$1 \le M, N \le 30$）。一些格子是坚固得令人惊讶的莲花，还有一些格子是岩石，其余的只是美丽、纯净、湛蓝的水。

贝西正在练习芭蕾舞，她站在一朵莲花上，想跳到另一朵莲花上去，她只能从一朵莲花跳到另一朵莲花上，既不能跳到水里，也不能跳到岩石上。

贝西的舞步很像象棋中的马步：每次总是先横向移动一格，再纵向移动两格，或先纵向移动两格，再横向移动一格。最多时，贝西会有八个移动方向可供选择。

约翰一直在观看贝西的芭蕾练习，发现她有时候不能跳到终点，因为中间缺了一些荷叶。于是他想要添加几朵莲花来帮助贝西完成任务。一贯节俭的约翰只想添加最少数量的莲花。当然，莲花不能放在石头上。

请帮助约翰确定必须要添加的莲花的最少数量，以及有多少种放置这些莲花的方法。

输入格式

- 第一行：两个用空格分开的整数：M和N
- 第二行到$M+1$行：第$i+1$行有N个用空格分开的整数，描述了池塘第i行的状态：0 为水，1 为莲花，2 为岩石，3 为贝西所在的起点，4 为贝西想去的终点。

输出格式

- 第一行：一个整数，需要增加的最少莲花数；如果无解，输出-1
- 第二行：放置这些莲花的方案数量，保证这个数字不会超过一个 64 位的有符号整数，如果第一行是-1，不要输出第二行

样例

lilypad.in	lilypad.out
4 5 1 0 0 0 0 3 0 0 0 0 0 0 2 0 0 0 0 0 4 0 （池塘分成四行五列，贝西的起点在第二行第一列，想去的终点在第四行第四列，池塘里一共有三朵莲花和一块石头）	2 3 （最少需要两朵莲花，有三种方式可以放置，如下 X 所示： 1 0 0 0 0　1 0 X 0 0　1 0 X 0 0 3 0 X 0 0　3 0 0 0 0　3 0 0 0 X 0 0 2 0 0　0 X 2 0 0　0 0 2 0 0 0 X 0 4 0　0 0 0 4 0　0 0 0 4 0 ）

解题思路

It suffices to consider the grid as a graph and the possible jumps as edges. Then the problem becomes: Given a graph where some vertices are marked, find the least number of additional vertices to mark so there exist a path between s and t.

First notice that each connected component can be 'shrinked' into one vertex which combines all the edges of the vertices it contains as we could walk around the component free of cost.

The standard way to go here is to construct some kind of weight on the edges to change the problem into finding the shortest path and counting the number of shortest paths. It follows that the distance should be the least number of vertices that needs to be marked

Clearly, any edge from an unmarked vertex to an unmarked vertex has cost one as another vertex needs to be marked. Things gets tricky in situations where a marked vertex is traversed, namely the following situations:
v1, v2 are marked, a and b are not. The following edges exist:
a->v1, a->v2, v1->b, v2->b
Then any kind of shortest path which involves v1 and v2 would end up calculating the same set of unmarked vertices (a and b) twice since there are two paths. Therefore, it's necessary to 'skip' marked vertices by having a path of length 1 directly from a to b.

It's quite clear that the least number of vertices to be marked equals the shortest distance from s to t minus one (the last edge to t should be free).

To get the answers, do a BFS from s, then go through each vertex in order and count the number of paths by summing the paths that lead to of all its neighbors which are distance d-1 away (assuming this vetex has distance d from s).

第七讲 USACO 竞赛试题选讲

哞哞大学之奖学金 (Moo University - Financial Aid, USACO 2004 Mar)

贝西发现人类可以上很多大学，而奶牛们却没有大学可上。为了解决这个问题，她和她的伙伴们创立了一所奶牛大学，取名为哞哞大学。

为了选拔合适的学生入学，她们发明了一种奶牛学术能力测试（简称 CSAT），这种测试的分数异常精确，每头奶牛的成绩可以用1到2,000,000,000之间的一个整数表示。

哞哞大学的学费很贵，很多奶牛都负担不起，他们需要申请一些奖学金（$1 \leq$ 奖学金 ≤ 10000）。可是政府并没有为奶牛准备奖学金，于是所有的预算都必须要从学校自身有限的基金中间扣除（设基金总额为F，$0 \leq F \leq 2{,}000{,}000{,}000$）。

更糟的事，哞哞大学的只有N ($1 \leq N \leq 19999$)间教室，N是一个奇数，而一共有C ($N \leq C \leq 100{,}000$)头奶牛申请入学，为了让最多的奶牛接受教育，贝西打算接受N头奶牛的申请，而且她还想让这些奶牛 CSAT 成绩的中位数尽可能地高。

所谓中位数，就是在一堆数字在排序后处在最中间的那个数字，比如{3,8,9,7,5}的中位数就是7。

给定每头奶牛的 CSAT 成绩和打算申请的奖学金数目，以及可以资助的基金总数，确定贝西接受哪些奶牛的申请才可以使成绩的中位数达到最大。

输入格式

- 第一行：三个用空格分开的整数：N，C和F。
- 第二行到$C+1$行：每行有两个用空格分开的整数。第一个数是这头奶牛的 CSAT 成绩，第二个数是这头奶牛想申请的奖学金。

输出格式

- 第一行：一个整数，表示贝西可以得到的最大中位数，如果现有基金不够资助N头奶牛，则输出-1。

样例

finance.in	finance.out
3 5 70 30 25 50 21 20 20 5 18 35 30	35 （贝西接受 CSAT 分数为 5，35，50 的奶牛的申请，中位数为 35，需支付的奖学金总额为 $18+30+21=69 \leq 70$）

解题思路

对中位数进行穷举，在中位数的两端分别选择最便宜的$(N-1)/2$头奶牛，寻找的过程可以用堆来实现。

This is a nice mix of algorithms. It is greedy in that, once you
have chosen the median cow, one should fund the cheapest (N-1)/2 cows
on the left, and the same on the right. Finding the cost of funding the
cheapest (N-1)/2 cows of the left-hand K cows can be done dynamically,
using a heap. The heap stores the actual costs of funding the cheapest
(N-1)/2 cows, with the most _expensive_ at the root. To add another cow
to the range, compare it to the root. If it is more expensive, ignore
it, otherwise remove the root and add the new cow to the heap.

The efficiency is O(C.log N).

哞哞大学之校队选拔 (Moo University - Team Tryouts, USACO 2004 Mar)

N头奶牛报名参加哞哞大学体操队的选拔，每头奶牛都有一个高度和宽度（不大于10000的正整数）。你的任务是尽可能让更多的奶牛加入队伍，而唯一要求就是，在这只队伍里面的每一头奶牛，都需满足以下不等式：

$$A(H-h)+B(W-w)\leq C$$

其中H和W为这头奶牛的高度和宽度，h和w为队伍里奶牛的最小高度和宽度，而A、B、C为三个不大于10000的正的整型常数。现在请计算队伍里最多可以有多少头奶牛。

输入格式

- 第一行：一个整数：N
- 第二行：三个分开的整数：A，B和C
- 第三行到第$N+2$行：每行有两个用空格分开的整数，分别表示一头奶牛的高度和宽度

输出格式

- 第一行：在队伍里的最大奶牛头数

样例

tryout.in	tryout.out
8 1 2 4 5 1 3 2 2 3 2 1 7 2 6 4 5 1 4 3	5 （第 1，2，3，4，7 号奶牛可以组成一支符合要求的队伍，没有更大的队伍了）

解题思路

The following simple algorithm will work, but will be too slow on
the big cases:

Loop over all weights and all heights in the set, and consider the loop variables to be the minimum weight and height. In this, loop over all cows and count the number that meet the minimum weight and height and fit the constraint.

This is O(n^3), which will be too slow for N = 1000.

We can refine this algorithm by building things up. Suppose we fix minh and loop through the possibilities for minw in increasing order. As minw increases, the set of points satisfying the linear constraint gets strictly bigger. If we sort the points by Ah + Bw, then they will become valid points in the order they appear (valid in the sense of the linear constraint, not relative to minw or minh). Thus we can use the following algorithm:

- Loop over minh
  - Loop over minw in increasing order
    - add new points to the set of valid points, keeping track of the last one added (for next time round the loop)
    - compare the count to the best so far
    - remove the point corresponding to minw from the set

This is now an O(n^2) algorithm, easily fast enough.

http://blog.doglog.net/2009/03/2008-moo-university-team-tryouts/

哞哞大学之匹萨预定 (Moo University - Emergency Pizza Order, USACO 2004 Mar)

哞哞大学咖啡厅的干草用完了，必须马上为C (1 ≤ C ≤ 1000)头奶牛预定匹萨。当地著名的披萨店——匹萨农庄出产的最大号匹萨正好可以喂饱一头奶牛。

匹萨农庄乐意为每头奶牛制作匹萨，但是由于订单规模太大，必须做出三个限制：

- 匹萨农庄有T (1 ≤ T ≤ 30)种蔬菜馅料，每份匹萨必须恰好使用K (1 ≤ K ≤ T)种馅料；
- 一份匹萨里面的相同馅料不能重复出现（比如说，不能用两份洋葱做一份匹萨）
- 任何两份匹萨的馅料不能完全相同。比如说如果 1 号匹萨使用了洋葱、青椒、凤梨和小麦草，那么 2 号匹萨的馅料就不能和它完全相同，但可以使用洋葱、青椒、凤梨和橄榄。

所有的馅料都用1到T之间的整数来表示。哞哞大学的奶牛们对馅料是很挑剔的，一些奶牛可能不会喜欢所有的馅料，每头奶牛吃的匹萨上一定要有一种她喜欢的馅料，请帮忙确定匹萨农庄最多能满足多少头奶牛的订单。

输入格式

- 第一行：三个整数：C，T和K。
- 第二行到C + 1行：每行描述了一头奶牛喜欢的馅料。第一个整数表示这头奶牛喜欢的馅料数目，其余的整数为这头奶牛喜欢的馅料种类，彼此用空格分开。

输出格式

- 第一行：可以吃到匹萨的奶牛的最大头数

样例

pizza.in	pizza.out
3 2 1 2 2 1 1 1 1 2 （总共有三头奶牛，匹萨农庄有两种馅料，每份匹萨只能用一种馅料，第一头奶牛两种馅料都喜欢，第二头奶牛只喜欢第一种馅料，第三头奶牛只喜欢第二种馅料）	2 （只能做出两种不同的披萨，可以把只加第一种馅料的披萨送给第一头奶牛，再把只加第二种馅料的披萨送给第二头奶牛，但是没有方法可以同时满足三头奶牛的要求。）

解题思路

This is essentially a matching problem: match as many cows to legal pizzas as possible. There may be an awful lot of pizzas, so they should be generated on the fly rather than building a giant list of pizzas and sets of legal pizzas for each cow. Optimal matching can be done by network flow [http://people.cs.uct.ac.za/~bmerry/manual/algorithms/netflow.html].

The algorithm can be made roughly 30 times faster in some places by representing pizzas (and topping sets in general) is to use the bits in a 32-bit integer, rather than an array of up to 30 booleans. This means that one can, for example, test whether a given pizza will satisfy a given cow in only a few ops: ~cow_likes & pizza == 0 (or for Pascal people, not cow_likes and pizza = 0).

Efficiency is at worst O(n^3) (n steps of BFS in a dense n-node graph), although in practice I think it will be much better in general, more like O(n^2) even when all the cows end up getting pizza.

第八讲 USACO 竞赛试题选讲（续）

黄金平衡 (USACO 2007 Mar)

输入格式

- 第一行：两个用空格分开的整数：N和K
- 第二行到第$N+1$行：第$i+1$行有一个K比特长的十进制数，表示第i头奶牛的K种特性，如果末位比特是 1，表示这头牛具有 1 号特性 1，如果首位比特是 1，则表示具有K号特性。

输出格式

- 第一行：只有一个整数，表示最大平衡区间的长度

样例

lineup.in	lineup.out
7 3 7 6 7 2 1 4 2	4 （选择 3 号牛到 6 号牛构成长度为 4 的区间，每种特性恰好出现了两次）

解题思路

可以把输入数据看成是一个$N \times K$的 01 矩阵A，构造一个可以快速求出某个区间和的矩阵S：S是一个$(N+1) \times K$的矩阵，对任意列c，$S[0,c] \equiv 0$，$S[r,c] = \sum_{i \le r} A[i,c]$。原问题就是要寻找一对间隔最大的行$0 \le r_1 < r_2 \le N$，使得对任意列c，

$$S[r_2,c] - S[r_1,c] = S[r_2,1] - S[r_1,1]$$

即

$$S[r_2,c] - S[r_2,1] = S[r_1,c] - A[r_1,1]$$

令$S'[r,c] = S[r,c] - S[r,1]$，将$S'$看做是$N+1$个K维行向量的序列，原问题就是要在$S'$中找一对全等的间隔最远的向量。

- 方法一：对向量进行排序
- 方法二：哈希表

奶牛排名 (Ranking the Cows, USACO 2007 Mar)

农夫约翰的每头奶牛的产奶率都不同。约翰打算按照产奶率从高到底为他的奶牛排名。

约翰一次只能拿两头奶牛经行 PK。约翰已经 PK 了M ($1 \leq M \leq 10000$)对奶牛的产奶率，为了完成所有名次的排序工作，约翰打算列一张还要继续 PK 的奶牛名单，这张名单的长度当然越短越好，设它的长度为C。请帮助约翰找出最小的C。

输入格式

- 第一行：两个用空格分开的整数：N和M
- 第二行到第$M+1$行：每行分别为两个用空格分开的整数：X和Y。X和Y都在1和N之间，表示奶牛X的产奶率要高于奶牛Y

输出格式

- 第一行：单独一个整数表示C的最小值

样例

ranking.in	ranking.out
5 5 2 1 1 5 2 3 1 4 3 4	3 （已知 2 高于 1 高于 5，2 高于 3 高于 4，所以 2 排在第一。还需要比较 1 和 3 分出第二，同理也要比较 4 和 5，若 1 高于 3，还需要比较 5 和 3。所以约翰至少要问三个问题：“奶牛 1 是否高于奶牛 3？”、“奶牛 4 是否高于奶牛 5？”、“奶牛 5 是否高于奶牛 3？”）

解题思路

如果从已知的比较结果中不能判断X和Y的高低，就称X, Y为一组无序对，记C'为无序对的数目，则$C' = C$。

奶牛交通 (Cow Traffic, USACO 2007 Mar)

由于奶牛数量的大膨胀，牧场里通往牛棚的道路不堪重负，为了缓解在挤奶高峰时间的交通堵塞，农夫约翰决定找出最拥挤的道路来整治。

牧场里有M ($1 \le M \le 50000$)条单行的道路，N ($1 \le N \le 5000$)个路口（从1到N标号），每条道路连接了两个不同的路口。牛棚就在N号路口上。.每一条道路都由编号较小的路口通向编号较大的路口，所以在道路网络里将不会出现环，而且犹如老话所讲的那样：“条条道路通牛棚”。注意同一对路口上可能出现一条以上的道路。

在挤奶的高峰期，一些奶牛会从放牧点走向牛棚。那些只出不进的路口都是放牧点。一个放牧点连向牛棚的道路序列构成一条路径，每条路径上都会有一头不同的奶牛通过。

帮助约翰找到最拥挤的那条道路上通过的奶牛数量。我们保证结果不会超过一个有符号的 32 位整数。

输入格式

- 第一行：两个用空格分开的整数：N和M
- 第二行到第$M+1$行：两个用空格分开的整数表示一条道路连接的两个路口的编号

输出格式

- 第一行：最拥挤的道路上通过的路径数量

样例

traffic.in	traffic.out
7 7 1 3 3 4 3 5 4 6 2 3 5 6 6 7	4 （最拥挤的道路是(6,7)，一共有四条路径： 1→3→4→6→7 1→3→5→6→7 2→3→4→6→7 2→3→5→6→7）

解题思路

$$\text{经过}(u,v)\text{的路径数目} = \text{从各起点到 u 的路径数目} \times \text{从 v 到 N 的路径数目}$$

校庆聚会 (Ural 1039)

Ural 大学的校长正在筹备学校的 80 周年纪念聚会。由于学校的职员有不同的职务级别，可以构成一棵以校长为根的人事关系树。每个职员都有一个唯一的整数编号（范围在1到N之间），并且每个人在参加聚会之后有一个欢乐气氛度。为了使每个参加聚会的人都能感到身心愉快，校长规定任何一个职员和他的直接上司不能同时参加聚会。

你的任务是草拟一份客人的邀请名单，使所有客人的欢乐气氛度总和最高。

输入格式

输入的第一行是一个整数N，$1 \leq N \leq 6000$。其后N行，每行表示一个职员的欢乐气氛度。欢乐度是一个在-128到127之间的整数。接着是学校的人事关系树，每一行有两个数L, K组成，表示第K个职员是第L个职员的直接上司，关系树将以0 0表示结束。

输出格式

输出客人欢乐气氛度总和的最大值。

样例

输入	输出
7 1 1 1 1 1 1 1 1 3 2 3 6 4 7 4 4 5 3 5 0 0	5

解题思路

第九讲 线段树