本章题目是对于搜索和 DP 的灵活运用，同时引入了 SCC 算法

Translate:USACO/milk4

Milk Measuring 量取牛奶

Hal Burch

译 by Felicia Crazy

描述

农夫约翰要量取 Q（1 <= Q <= 20,000）夸脱（夸脱，quarts，容积单位——译者注） 他的最好的牛奶，并把它装入一个大瓶子中卖出。消费者要多少，他就给多少，从不有任何误差。

农夫约翰总是很节约。他现在在奶牛五金商店购买一些桶，用来从他的巨大的牛奶池中量出 Q 夸脱的牛奶。每个桶的价格一样。你的任务是计算出一个农夫约翰可以购买的最少的桶的集合，使得能够刚好用这些桶量出 Q 夸脱的牛奶。另外，由于农夫约翰必须把这些桶搬回家，对于给出的两个极小桶集合，他会选择“更小的”一个，即：把这两个集合按升序排序，比较第一个桶，选择第一个桶容积较小的一个。如果第一个桶相同，比较第二个桶，也按上面的方法选择。否则继续这样的工作，直到相比较的两个桶不一致为止。例如，集合 {3，5，7，100} 比集合 {3，6，7，8} 要好。

为了量出牛奶，农夫约翰可以从牛奶池把桶装满，然后倒进瓶子。他决不把瓶子里的牛奶倒出来或者把桶里的牛奶倒到别处。用一个容积为 1 夸脱的桶，农夫约翰可以只用这个桶量出所有可能的夸脱数。其它的桶的组合没有这么方便。

计算需要购买的最佳桶集，保证所有的测试数据都至少有一个解。

格式

PROGRAM NAME: milk4

INPUT FORMAT

Line 1: 一个整数 Q

Line 2: 一个整数 P（1 <= P <= 100），表示商店里桶的数量

Lines 3..P+2: 每行包括一个桶的容积（1 <= 桶的容积 <= 10000）

OUTPUT FORMAT

输出文件只有一行，由空格分开的整数组成：

为了量出想要的夸脱数，需要购买的最少的桶的数量，接着是：

一个排好序的列表（从小到大），表示需要购买的每个桶的容积

SAMPLE INPUT (file milk4.in)

```
16
3
3
5
7
```

SAMPLE OUTPUT (file milk4.out)

```
2 3 5
```

问题分析

刚开始看这道题目不是很好分析，感觉暴力搜索的复杂度会很高，这也是 USACO 的一贯风格。但是本题目就是使用 DFS-ID 加 DP 求解即可，首先要选择出要选用的桶的组合，这个可以使用 DFS-ID 很简单的实现。注意每个层次可以选择的桶的数目的上界，例如如果这次共选 Q-1个桶，那么第一个桶只能在1到2之间选择，选第3个会使得本程序的意义失效（think why...）,因此当前层次可以选的上界是 p-(edge-lev) 其中 p 代表桶的总数，edge 代表本次要选多少个，lev 代表当前已经选到第几个,这样我们就有了一个给定大小的集合的返回函数，定义为 S(lev)。

当找到目前要使用的桶的集合时，就要开始判断可否可行了，首先在 DFS-ID 的时候，我们的搜索顺序决定了当前集合是满足题目要求的。判断一个容量可否用给定集合组成可以使用完全背包判断。那么每个桶都要被使用至少一次，是否能被正确判断呢？我们假设函数 **Pa(Ps,S(lev))**(Ps 表示 S 的大小)的返回值集合是 **{true, false}**，如果返回值是 false 表明不可能组成，而返回值是 true 则相反，同时我们也可以得到附加的信息，首先搜索顺序决定了**如果 S(lev)可行那么 S(lev-1)一定不可行**，否则会产生矛盾，这样我们可以加上一个优化，一般完全背包的初始条件是 DP[0] = true,但是因为 Pa 函数的性质，我们可以不理会 DP[0]，毕竟每一个桶都至少要使用一次，因此初始条件可以把集合中的第一个桶都提前加上去。状态转移方程 **DP[i] = DP[i] | DP[i - v[j]];**

Translate:USACO/window

Window Area 窗体面积

IV Balkan Olympiad

译 by Felicia Crazy

描述

你刚刚接手一项窗体界面工程。窗体界面还算简单，而且幸运的是，你不必

显示实际的窗体。有 5 种基本操作：

创建一个新窗体
将窗体置顶

```
将窗体置底
删除一个窗体
输出窗体可见部分的百分比（就是，不被其它窗体覆盖的部分）。
```

在输入文件中，操作以如下的格式出现。

```
创建一个新窗体：w(I,x,y,X,Y)
将窗体置顶： t(I)
将窗体置底： b(I)
删除一个窗体：d(I)
输出窗体可见部分的百分比：s(I)
```

I 是每个窗体唯一的标识符，标识符可以是 'a'..'z', 'A'..'Z' 和 '0'..'9' 中的任何一个。输入文件中没有多余的空格。

（x,y）和（X,Y）是窗体的对角。当你创建一个窗体的时候，它自动被“置顶”。你不能用已经存在的标识符来创建窗体，但是你可以删除一个窗体后再用已删除窗体的标识符来创建窗体。坐标用正整数来表示，并且所有的窗体面积都不为 0（x <> X 且 y <> Y）。x 坐标和 y 坐标在 1 —— 32767 的范围内。

格式

PROGRAM NAME: window

INPUT FORMAT

输入文件包含给你的解释程序的一系列命令，每行一个。当输入文件结束时，停止程序。

OUTPUT FORMAT

只对于 s() 命令进行输出。当然，输入文件可能有许多 s() 命令(不超过 500 次)，所以输出文件应该是一个百分比的序列，每行一个，百分比是窗体可见部分的百分比。百分比应该四舍五入到三位小数。

SAMPLE INPUT (file window.in)

```
w(a,10,132,20,12)
w(b,8,76,124,15)
s(a)
```

SAMPLE OUTPUT (file window.out)

```
49.167
```

问题分析

和3.1.4运用到的核心思想一致，就是矩形切割，3.1.4我写的是非递归的，这回用递归写就很得心应手了。其他的部分就是模拟了，为了减少编程时间和避免编程错误，使用 STL 中 list 和 map 来辅助查询某一个矩形和模拟题目中的给定操作。判断两个矩形相交的时候，可以从反面思考，因为两个矩形不相交更容易判断(只需要判断4个条件)，除此之外就是相交的情况了。

Translate:USACO/schlnet

Network of Schools 校园网

IOI '96 Day 1 Problem 3

译 by Felicia Crazy

描述

一些学校连入一个电脑网络。那些学校已订立了协议：每个学校都会给其它的一些学校分发软件（称作“接受学校”）。注意如果 B 在 A 学校的分发列表中，那么 A 不一定也在 B 学校的列表中。

你要写一个程序计算，根据协议，为了让网络中所有的学校都用上新软件，必须接受新软件副本的最少学校数目（子任务 A）。更进一步，我们想要确定通过给任意一个学校发送新软件，这个软件就会分发到网络中的所有学校。为了完成这个任务，我们可能必须扩展接收学校列表，使其加入新成员。计算最少需要增加几个扩展，使得不论我们给哪个学校发送新软件，它都会到达其余所有的学校（子任务 B）。一个扩展就是在一个学校的接收学校列表中引入一个新成员。

格式

PROGRAM NAME: schlnet

INPUT FORMAT 输入文件的第一行包括一个整数 N：网络中的学校数目（2 <= N <= 100）。学校用前 N 个正整数标识。接下来 N 行中每行都表示一个接收学校列表（分发列表）。第 i+1 行包括学校 i 的接收学校的标识符。每个列表用 0 结束。空列表只用一个 0 表示。

OUTPUT FORMAT

你的程序应该在输出文件中输出两行。第一行应该包括一个正整数：子任务 A 的解。第二行应该包括子任务 B 的解。

SAMPLE INPUT (file schlnet.in)

```
5
2 4 3 0
```

```
4 5 0
0
0
1 0
```

SAMPLE OUTPUT (file schlnet.out)

```
1
2
```

问题分析

首先分析题目大意，简言之就是你要找出整张图的 SCC,第一问是给最少多少个学校发送软件可以使其他学校也得到软件。解决方案是求出 SCC 后，我们可以利用缩图的结果，找到所有入度为0的点，这些点肯定要的得到软件。其他的结点只要不是入度为0就都可以得到，因为 SCC 是拓扑的。这样第一问就解决了，用到的 SCC 算法是 Tarjan 算法。时间复杂度 O(M+N)。
第二问的要求是最少添加多少条有向边可以使得整张图任意一个学校有软件，其他的就能得到。也就是我们至少添加多少边可以令目前的缩图强连通。考虑 SCC 是拓扑的，那么每一对入度和出度为0的不同的点之间添加一条边使其成环即可让其融入缩图之中，那么当这些点对都处理完时，要么整张图都缩成一个点，要么剩余一些只剩出度或者入度的点，对于这些剩余点只要让其他强连通分量任意引出一个互补的边就可以使其也同样成为一个缩点。因此最小需要 **MAX(出度为0，入度为0)**个边。有一个特殊情况，如果 SCC 本身就是一个点，就不需要其他额外的边。

Translate:USACO/bigbrn

Big Barn 巨大的牛棚

A Special Treat 译 by Felicia Crazy

描述

农夫约翰想要在他的正方形农场上建造一座正方形大牛棚。他讨厌在他的农场中砍树，想找一个能够让他在空旷无树的地方修建牛棚的地方。我们假定，他的农场划分成 N x N 的方格。输入数据中包括有树的方格的列表。你的任务是计算并输出，在他的农场中，不需要砍树却能够修建的最大正方形牛棚。牛棚的边必须和水平轴或者垂直轴平行。

格式

EXAMPLE

考虑下面的方格，它表示农夫约翰的农场，'.'表示没有树的方格，'#'表示有树的方格

```
  1 2 3 4 5 6 7 8
1 . . . . . . . .
2 . # . . . # . .
3 . . . . . . . .
4 . . . . . . . .
5 . . . . . . . .
6 . . # . . . . .
7 . . . . . . . .
8 . . . . . . . .
```

最大的牛棚是 5 x 5 的，可以建造在方格右下角的两个位置其中一个。

PROGRAM NAME: bigbrn

INPUT FORMAT

Line 1: 两个整数： N （1 <= N <= 1000），农场的大小，和 T （1 <= T <= 10，000）有树的方格的数量

Lines 2..T+1: 两个整数（1 <= 整数 <= N), 有树格子的横纵坐标

OUTPUT FORMAT

输出文件只由一行组成，约翰的牛棚的最大边长。

SAMPLE INPUT (file bigbrn.in)

```
8 3
2 2
2 6
6 3
```

SAMPLE OUTPUT (file bigbrn.out)

```
5
```

问题分析

简单的 DP,状态转移方程：
DP[i][j] = MIN(DP[i-1][j],DP[i][j-1],DP[i-1][j-1])+1;
如果 MAP[i-1][j] || MAP[i][j-1] || MAP[i-1][j-1]是树，则 DP[i][j]=1.同时刷新最大值即可。