# 2017 省队选拔模拟测试—网络流

**（请选手务必仔细阅读本页内容 时间 8:00 ~12:30）**

一、题目概况

| 中文名称 | 奶牛议会 | 基因串 | 最优标号 | 美食节 |
|---|---|---|---|---|
| 英文名称 | cowngress | gen | optimalmarks | delicacy |
| 输入文件 | cowngress.in | gen.in | ~.in | delicacy.in |
| 输出文件 | cowngress.out | gen.out | ~.out | delicacy.out |
| 测试点时限 | 1000 毫秒 | 1000 毫秒 | 1000 毫秒 | 3000 毫秒 |
| 测试点数目 | 10 | 10 | 10 | 10 |
| 测试点分值 | 10 | 10 | 10 | 10 |
| 比较方式 | 全文比较 | | | |
| 题目类型 | 传统 | 传统 | 传统 | 传统 |
| 内存上限 | 256 兆字节 | 256 兆字节 | 256 兆字节 | 512 兆字节 |

二、提交源程序文件名

| C | cowngress.c | gen.c | optimalmarks.c | delicacy.c |
|---|---|---|---|---|
| C++ | cowngress.cpp | gen.cpp | ~.cpp | delicacy.cpp |
| Pascal | cowngress.pas | gen.pas | ~.pas | delicacy.pas |

三、编译命令

| C | gcc -Wall -std=c99 -DCONTEST -o foo src.c -lm |
|---|---|
| C++ | g++ -Wall -std=c++11 -DCONTEST -o foo src.cpp -lm |
| Pascal | fpc -Mtp -v0 -dCONTEST -Sgic -Tlinux -ofoo src.pas -lm |

**注意事项：**

1、 文件名（程序名和输入输出文件名）必须使用英文小写。

2、 C/C++中函数 main 的返回值类型必须是 int，程序正常结束时返回值必须是 0。

3、 评测时采用的机器配置为：Intel Pentium G2020 2.90 GHz × 2 处理器，4GB 内存。
上述时限以此配置为准。

4、 特别提醒：评测在 Ubuntu 14.04 x86_64 操作系统上进行，各语言的编译器版本如下：
GCC 4.8.4，FPC 2.6.4。

5、评测在官方 Linux 环境下进行，不会开启 -O2 优化。

6、**题目不难，要当作正式比赛认真对待，争取把能得到的分都拿到**。

7、**严禁上网，严禁互相讨论交流，严禁大声喧哗影响他人做题，违者驱逐出考场**。

# 1. 奶牛议会

【问题描述】

由于对 Farmer John 的领导感到极其不悦，奶牛们退出了农场，组建了奶牛议会。议会以“每头牛 都可以获得自己想要的”为原则，建立了下面的投票系统： M 只到场的奶牛 (1 <= M <= 4000) 会给 N 个议案投票(1 <= N <= 1,000) 。每只奶牛会对恰好两个议案 B_i and C_i (1 <= B_i <= N; 1 <= C_i <= N)投 出“是”或“否”（输入文件中的'Y'和'N'）。他们的投票结果分别为 VB_i (VB_i in {'Y', 'N'} ) and VC_i (VC_i in {'Y', 'N'} )。

最后，议案会以如下的方式决定：每只奶牛投出的两票中至少有一票和最终结果相符合。

例如 Bessie 给议案 1 投了赞成'Y'，给议案 2 投了反对'N'，那么在任何合法的议案通过 方案中，必须满足议案 1 必须是'Y'或者议案 2 必须是'N'（或者同时满足）。

给出每只奶牛的投票，你的工作是确定哪些议案可以通过，哪些不能。如果不存在这样一个方案， 输出"IMPOSSIBLE"。如果至少有一个解，输出：

Y 如果在每个解中，这个议案都必须通过

N 如果在每个解中，这个议案都必须驳回

? 如果有的解这个议案可以通过，有的解中这个议案会被驳回

考虑如下的投票集合：

```
----- 议案 -----
     1       2       3
奶牛 1       YES     NO
奶牛 2       NO      NO
奶牛 3       YES     YES
奶牛 4       YES     YES
```

下面是两个可能的解：

* 议案 1 通过（满足奶牛 1，3，4）
* 议案 2 驳回（满足奶牛 2）
* 议案 3 可以通过也可以驳回（这就是有两个解的原因）

事实上，上面的问题也只有两个解。所以，输出的答案如下： YN?

【输入】

* 第 1 行：两个空格隔开的整数：N 和 M
* 第 2 到 M+1 行：第 i+1 行描述第 i 只奶牛的投票方案：B_i, VB_i, C_i, VC_i

【输出】

* 第 1 行：一个含有 N 个字符的串，第 i 个字符要么是'Y'（第 i 个议案必须通过），或者是'N' （第 i 个议案必须驳回），或者是'?'。 如果无解，输出"IMPOSSIBLE"。

【输入输出样例】

| Input | Output |
|---|---|
| 3 4<br>1 Y 2 N<br>1 N 2 N<br>1 Y 3 Y<br>1 Y 2 Y | YN? |

# 2. 基因串

【问题描述】

基因串是由一串有限长度的基因所组成的，其中每一个基因都可以用 26 个英文大写字母中的一个来表示，不同的字母表示不同的基因类型。一个单独的基因 可以生长成为一对新的基因，而可能成长的规则是通过一个有限的成长规则集所决定的。每一个成长的规则可以用三个大写英文字母 A1A2A3 来描述，这个规则 的意思是基因 A1 可以成长为一对基因 A2A3。

我们用大写字母 S 来表示一类被称作超级基因的基因。因为每一个基因串都是由一串超级基因根据给出的规则所成长出来的。

请写一个程序：

从文件中读入有限条成长的规则和一些我们想要得到的基因串；

对于每个基因串，试判断它是否可以由一个有限长度的超级基因串成长得出。如果可以，那么请给出可成长为该基因串的最短超级基因串的长度；

【输入】

第一行包括一个整数 n，1 <= n <= 1000。以下的 n 行中每行都包括一个成长的规则，每个规则由三个大写英文字母组成。

第 n+1 行包括一个整数 k，1 <= k <=1000。以下的 k 行中每行都有一个基因串。每个基因串都是一个长度不超过 100 的大写字符串。

【输出】

输出在第 i（应共 k 行）行中你应该输出入下内容：

一个正整数，表示成长为该基因串所需的最短的超级基因串的长度；或一个单词 NIE（波兰语的“否”），如果说无法由超级基因串成长成为该基因串。

【输入输出样例 1】

| Input | Output |
|---|---|
| 6<br>SAB<br>SBC<br>SAA<br>ACA<br>BCC<br>CBC<br>3<br>ABBCAAABCA<br>CCC<br>BA | 3<br>1<br>NIE |

# 3. 最优标号

【问题描述】

给你一张无向图 G(V,E)。每个顶点都有一个标号，它是一个[0,2^32-1]内的整数。不同的顶点可能会有相同的标号。

对每条边(u,v)，我们定义其费用 cost(u,v)为 u 的标号与 v 的标号的异或值。现在我们知道一些顶点的标号。你需要确定余下顶点的标号使得所有边的费用和尽可能小。

【输入】

第一行有两个整数 N,M(1<=N<=500,0<=M<=3000)，N 是图的点数，M 是图的边数。

接下来有 M 行，每行有两个整数 u,v，代表一条连接 u,v 的边。

接下来有一个整数 K，代表已知标号的顶点个数。接下来的 K 行每行有两个整数 u,p，代表点 u 的标号是 p。假定这些 u 不会重复。

【输出】

输出一行一个整数，即最小的费用和。

【输入输出样例 1】

| Input | Output |
| --- | --- |
| 3 2<br>1 2<br>2 3<br>2<br>1 5<br>3 100 | 97 |

【样例解释】

一个可能的标号方案是：点 1 标 5，点 2 标 4，点 3 标 100.

# 4. 美食节

【问题描述】

CZ 市为了欢迎全国各地的同学，特地举办了一场盛大的美食节。

作为一个喜欢尝鲜的美食客，小 M 自然不愿意错过这场盛宴。他很快就尝遍了美食节所有的美食。然而，尝鲜的欲望是难以满足的。尽管所有的菜品都很可口，厨师做菜的速度也很快，小 M 仍然觉得自己桌上没有已经摆在别人餐桌上的美食是一件无法忍受的事情。于是小 M 开始研究起了做菜顺序的问题，即安排一个做菜的顺序使得同学们的等待时间最短。

小 M 发现，美食节共有 n 种不同的菜品。每次点餐，每个同学可以选择其中的一个菜品。总共有 m 个厨师来制作这些菜品。当所有的同学点餐结束后，菜品的制作任务就会分配给每个厨师。然后每个厨师就会同时开始做菜。**厨师们会按照要求的顺序进行制作，并且每次只能制作一人份**。

此外，小 M 还发现了另一件有意思的事情: 虽然这 m 个厨师都会制作全部的 n 种菜品，**但对于同一菜品，不同厨师的制作时间未必相同**。 他将菜品用 1, 2, ..., n 依次编号，厨师用 1, 2, ..., m 依次编号，将第 j 个厨师制作第 i 种菜品的时间记为 $t_{i,j}$ 。

小 M 认为：每个同学的等待时间为所有厨师开始做菜起，到自己那份菜品完成为止的时间总长度。换句话说，如果一个同学点的菜是某个厨师做的第 k 道菜，则他的等待时间就是这个厨师制作前 k 道菜的时间之和。 而**总等待时间为所有同学的等待时间之和**。

现在，小 M 找到了所有同学的点菜信息: 有 pi 个同学点了第 i 种菜品（i=1, 2, ..., n）。他想知道的是最小的总等待时间是多少。

【输入】

第 1 行包含两个正整数 n 和 m，表示菜品的种数和厨师的数量。

第 2 行包含 n 个正整数，其中第 i 个数为 pi，表示点第 i 种菜品的人数。

接下来有 n 行，每行包含 m 个非负整数，这 n 行中的第 i 行的第 j 个数为 $t_{i,j}$，表示第 j 个厨师制作第 i 种菜品所需的时间。

输入文件中每行相邻的两个数之间均由一个空格隔开，行末均没有多余空格。

【输出】

输出仅一行包含一个整数，为总等待时间的最小值。

【输入输出样例】

| Input | Output |
|---|---|
| 3 2<br>3 1 1<br>5 7<br>3 6<br>8 9 | 47 |

【样例说明】

厨师 1 先制作 1 份菜品 2，再制作 2 份菜品 1。点这 3 道菜的 3 个同学的等待时间分别为 3，3+5=8，3+5+5=13。

厨师 2 先制作 1 份菜品 1，再制作 1 份菜品 3。点这 2 道菜的 2 个同学的等待时间分别为 7，7+9=16。
总等待时间为 3+8+13+7+16=47。
虽然菜品 1 和菜品 3 由厨师 1 制作更快，如果这些菜品都由厨师 1 制作，总等待时间反而更长。如果按上述的做法，将 1 份菜品 1 和 1 份菜品 3 调整到厨师 2 制作，这样厨师 2 不会闲着，总等待时间更短。
可以证明，没有更优的点餐方案。

数据约束：

对于 100%的数据，$n \le 40, m \le 100, p \le 800, t_{i,j} \le 1000$（其中$p = \sum p_i$，即点菜同学的总人数）。

每组数据的 $n$、$m$ 和 $p$ 值如下：

| 测试点编号 | $n$ | $m$ | $p$ |
|---|---|---|---|
| 1 | $n=5$ | $m=5$ | $p=10$ |
| 2 | $n=40$ | $m=1$ | $p=400$ |
| 3 | $n=40$ | $m=2$ | $p=300$ |
| 4 | $n=40$ | $m=40$ | $p=40$ |
| 5 | $n=5$ | $m=40$ | $p=100$ |
| 6 | $n=10$ | $m=50$ | $p=200$ |
| 7 | $n=20$ | $m=60$ | $p=400$ |
| 8 | $n=40$ | $m=80$ | $p=600$ |
| 9 | $n=40$ | $m=100$ | $p=800$ |
| 10 | $n=40$ | $m=100$ | $p=800$ |