主要介绍了图论的比较高等的内容和其他的分析题目

Translate:USACO/ditch

Drainage Ditches 草地排水

Hal Burch

译 by Felicia Crazy

描述

在农夫约翰的农场上，每逢下雨，Bessie 最喜欢的三叶草地就积聚了一潭水。这意味着草地被水淹没了，并且小草要继续生长还要花相当长一段时间。因此，农夫约翰修建了一套排水系统来使贝茜的草地免除被大水淹没的烦恼（不用担心，雨水会流向附近的一条小溪）。作为一名一流的技师，农夫约翰已经在每条排水沟的一端安上了控制器，这样他可以控制流入排水沟的水流量。

农夫约翰知道每一条排水沟每分钟可以流过的水量，和排水系统的准确布局（起点为水潭而终点为小溪的一张网）。需要注意的是，有些时候从一处到另一处不只有一条排水沟。

根据这些信息，计算从水潭排水到小溪的最大流量。对于给出的每条排水沟，雨水只能沿着一个方向流动，注意可能会出现雨水环形流动的情形。

格式

PROGRAM NAME:ditch

INPUT FORMAT:

(file ditch.in)

第 1 行: 两个用空格分开的整数 N (0 <= N <= 200) 和 M (2 <= M <= 200)。N 是农夫 John 已经挖好的排水沟的数量，M 是排水沟交叉点的数量。交点 1 是水潭，交点 M 是小溪。

第二行到第 N+1 行: 每行有三个整数，Si, Ei, 和 Ci。Si 和 Ei (1 <= Si, Ei <= M) 指明排水沟两端的交点，雨水从 Si 流向 Ei。Ci (0 <= Ci <= 10,000,000) 是这条排水沟的最大容量。

OUTPUT FORMAT:

(file ditch.out)

输出一个整数，即排水的最大流量。

SAMPLE INPUT

```
5 4
1 2 40
1 4 20
2 4 20
2 3 30
3 4 10
```

SAMPLE OUTPUT

```
50
```

分析:
直接网络流

Translate:USACO/stall4

The Perfect Stall 完美的牛栏

Hal Burch

译 by Felicia Crazy

描述

农夫约翰上个星期刚刚建好了他的新牛棚，他使用了最新的挤奶技术。不幸的是，由于工程问题，每个牛栏都不一样。第一个星期，农夫约翰随便地让奶牛们进入牛栏，但是问题很快地显露出来：每头奶牛都只愿意在她们喜欢的那些牛栏中产奶。上个星期，农夫约翰刚刚收集到了奶牛们的爱好的信息（每头奶牛喜欢在哪些牛栏产奶）。一个牛栏只能容纳一头奶牛，当然，一头奶牛只能在一个牛栏中产奶。

给出奶牛们的爱好的信息，计算最大分配方案。

格式

PROGRAM NAME: stall4

INPUT FORMAT:

(file stall4.in)

第一行 两个整数，N (0 <= N <= 200) 和 M (0 <= M <= 200) 。N 是农夫约翰的奶牛数量，M 是新牛棚的牛栏数量。

第二行到第 N+1 行 一共 N 行，每行对应一只奶牛。第一个数字 (Si) 是这头奶牛愿意在其中产奶的牛栏的数目 (0 <= Si <= M) 。后面的 Si 个数表示这些牛栏的编号。牛栏的编号限定在区间 (1..M) 中，在同一行，一个牛栏不会被列出两次。

OUTPUT FORMAT:

(file stall4.out)

只有一行。输出一个整数，表示最多能分配到的牛栏的数量.

SAMPLE INPUT

```
5 5
2 2 5
3 2 3 4
2 1 5
3 1 2 5
1 2
```

SAMPLE OUTPUT

```
4
```

分析:
直接二分图

Translate:USACO/job

Job Processing 工序安排

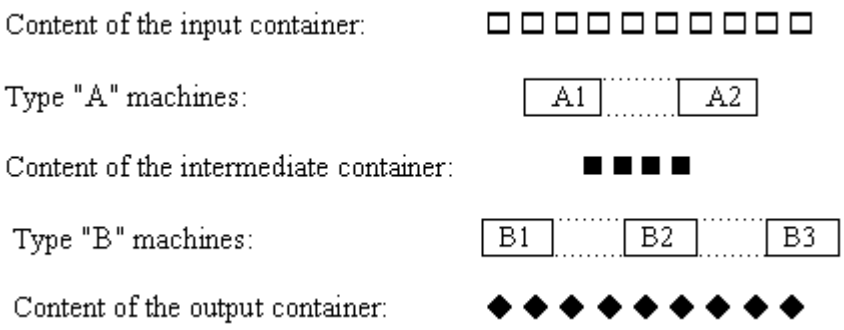
IOI'96

译 by Felicia Crazy

描述

一家工厂的流水线正在生产一种产品，这需要两种操作：操作 A 和操作 B。

每个操作只有一些机器能够完成。



上图显示了按照下述方式工作的流水线的组织形式。A 型机器从输入库接受工件，对其施加操作 A，得到的中间产品存放在缓冲库。B 型机器从缓冲库接受中间产品，对其施加操作 B，得到的最终产品存放在输出库。所有的机器平行并且独立地工作，每个库的容量没有限制。每台机器的工作效率可能不同，一台机器完成一次操作需要一定的时间。

给出每台机器完成一次操作的时间，计算完成 A 操作的时间总和的最小值，和完成 B 操作的时间总和的最小值。

格式

PROGRAM NAME: job

INPUT FORMAT:

(file job.in)

第一行 三个用空格分开的整数：N，工件数量 (1<=N<=1000);M1，A 型机器的数量 (1<=M1<=30);M2，B 型机器的数量 (1<=M2<=30)。

第二行…等 M1 个整数（表示 A 型机器完成一次操作的时间，1..20），接着是 M2 个整数（B 型机器完成一次操作的时间，1..20）

OUTPUT FORMAT:

(file job.out)

只有一行。输出两个整数：完成所有 A 操作的时间总和的最小值，和完成所有 B 操作的时间总和的最小值（A 操作必须在 B 操作之前完成）.

SAMPLE INPUT

```
5 2 3
1 1 3 1 4
```

SAMPLE OUTPUT

```
3 5
```

问题分析

我们令 delay[A][i]表示第 i 个 A 型机器的延时，显然，初始时所有 delay[A]为 0。然后对于所有的工件，我们选取 delay[A][i]+machine[A][i](machine[A][i]表示第 i 个 A 型机器的加工用时）最小，也就是能在最快时间内完工的机器加工。更新 delay[A][i]=delay[A][i]+machine[A][i]（这样这个机器就进入新的一轮加工中）。在此过程中，我们用 cost[A][j]记录 A 操作加工出第 j 个零件的用时。用同样的方法求出 delay[B][i]，cost[B][j]。

对于 A 操作的最短用时，显然就是 cost[A]中的最大值。

因为我们在求解过程中是一个工件一个工件地送去加工，每次把一个工件送给用时最短的机器加工，那么显然有 cost[A][1]<=cost[A][2]...<=cost[A][n]，同样适用于 cost[B]。为了使 B 操作的用时最少，**我们应该把 A 先加工出来的工件给 B 中加工最久的**，即：cost[A][1]->cost[B][n],cost[A][2]->cost[B][n-1].....cost[A][i]->cost[B][n-i+1] 那么 max(cost[A][i]+cost[B][n-i+1]) i=1,2,3,4...n 就是 B 操作的最短用时。

补充，要是 M1,M2 很大，可以用堆去维护。

问题证明

现在来证明为什么在第一步最优的前提下能得出第二步最优假设在 A 已经最优的前提下可以把一个零件拿出放到另一个 A 的机器上使得 B 最后的结果比当前好,那么假设这两台 A 机器为 A1,A2, 对应的两台 B 机器为 B1,B2. 对应的时间有分别是 a1,a2,b1,b2. 且 a2>=a1,b2>=b1,a2/2<=a1<=a2,b2/2<=b1<=b2. 那么原来 A 最优时的结果应该是 max1{a1+b2,a2+b1},而现在如果把 A 的一个机器(假设是 A2)的零件放到了 A1 上,那么现在的最优值就为 max2{a1+a2+b1,b2},现在只要证明 max1 恒<=max2 就可以了,分四种

情况证明：一：

假设：
a2+b1>=a1+b2,
b2>=a1+a2+b1
且 b2<a2+b1.
证明：
由 a2+b1>=a1+b2 => a2+b1>=b2 (1)
由 b2>=a1+a2+b1 => a2+b1<=b2 (2)
综合(1)(2)可知 a2+b1=b2 与假设矛盾.

二：

假设：
a2+b1>=a1+b2,
a1+a2+b1>=b2
且 a1+a2+b1<a2+b1 <<= 显然不成立

三：

假设：
a1+b2>=a2+b1, (1)
b2>=a1+a2+b1
且 b2<a1+b2 (4)
证明：
由 b2>=a1+a2+b1 => b2-a1>=a2+b1 (2)
（1）+（2） =〉 b2>=a2+b1 (3)
(3)(4)矛盾。

四：

假设：
a1+b2>=a2+b1, （1）
a1+a2+b1>=b2 （2）
且 a1+a2+b1<a1+b2 （3）
证明：
现在对不等式进行一种等价变换，在不影响(1)(2)不等式符号和 a1+b2,a1+a2+b1 的差的
前提下,来试图证明不等式（3）是不可能的
让 a1=a2,则(1)==>> a2+b2>=a2+b1
(2)==>> 2a2+b1>=b2
(3)==>> 2a2+b1<a2+b2

观察(2),(3),应为 a2>=0，所以显然是不可能同时成立的，所以问题得证。

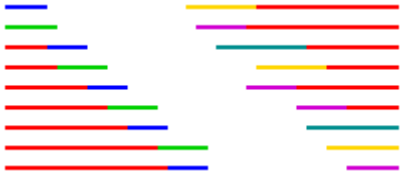
即：

在 A 为最优的前提下一定能够造出一个问题二也是最优的解。

USACO:You can greedily determine the earliest time the kth job can be finished by an "A" machine. For each machine, keep track when it will complete its current job. The time for it to complete the next job is this time plus the time is takes the machine to process a job. To determine the time for the kth job, pick the machine which would finish it first and assign that job to it, updating the time that the machine will complete its job.

Similarly, determine how early the kth job from the end can be put into a B machine, where earliness is measured from the end of all jobs. This is akin to flipping the process around, where each finish job must be `undone' by a B machine.

Once these two calculations have been done and the arrays have been sorted, you end up with a picture like this:



Each line represents the activity of one job. Green and blue are "A" machines, and yellow, cyan, and purple are type "B" machines. A red line means that the job is in a container instead of a machine. The left portion corresponds to "A" jobs, where the end of each line is the time at which the kth job is completed. The right portion corresponds to "B" jobs, where the beginning of the line is the earliest that the kth job can be started with respect to the ending time of the all the "B" jobs. The white space in the middle represents the `slack' time, the time that the job sits in an intermediate container.

The best option is to match up the earliest completed "A" job with the "B" job that starts earliest, the second earliest completed "A" job with the second earliest started "B" job, etc. Take the maximum of these times. This corresponds to moving the the two representations together until they touch (one job has no 'slack' time).

Translate:USACO/cowcycle

Cowcycles 奶牛自行车

Originally by Don Gillies

译 by Felicia Crazy

[读者请注意，原题中的一些词语已经被改成了有关奶牛的双关语。]

秀·谢夫（小奶牛）在花花公子杂志上中了大奖，于是她从农村搬到了城郊的一座别墅中。可是她还常常怀念乡村的生活，总想回到原来的农村逛逛。为了环保，秀决定骑上为她量身定做的奶牛自行车（特殊的自行车，专门为牛蹄设计）。

秀大约有一吨重。同样的，秀在普通的奶牛自行车上，要想骑得平平稳稳，也不是一件容易的事。因此，调节奶牛自行车的变速器让秀心力交瘁。

帮助秀选择她的奶牛自行车前面 F （1 <= F <= 5）个齿轮和后面 R （1 <= R <= 10）个齿轮，使她的 F*R 奶牛自行车符合下面的标准：

前面齿轮的型号（齿的数量）必须在给定的范围内。

后面齿轮的型号（齿的数量）必须在给定的范围内。

在每一种齿轮组合中，传动比率就是前面齿轮的齿数除以后面齿轮的齿数所得的商。

最大的传动比率至少是最小的三倍。

齿轮组合（已排好序）相邻两项的差的的方差（见下面的例子） 应该达到最小。

按照下面的公式计算平均数与方差（xi 代表数据）：

$$\text{平均数} = \frac{1}{n}\sum_{i=1}^{n} xi$$

$$\frac{1}{n}$$

方差 = --- SUM (xi - 平均数)2

n i=1

计算并确定最佳齿轮组合（其中 F 个前齿轮，R 个后齿轮），使方差最小（传动比率至少是 3x）。

格式

PROGRAM NAME: cowcycle

INPUT FORMAT:

(file cowcycle.in)

第一行是 F 和 R，表示前齿轮和后齿轮的数量。

第二行包括 4 个数字：F1，F2（25 <= F1 < F2 <= 80），R1，R2（5 <= R1 < R2 <= 40）。从 F1 到 F2 型号的前齿轮都是可用的；从 R1 到 R2 型号的后齿轮都是可用的。至少会有一组合法的解。

OUTPUT FORMAT:

(file cowcycle.out)

在第一行从小到大输出前齿轮的型号，用空格分开。在第二行从小到大输出后齿轮的型号，同样用空格分开。当然，齿轮的齿数一定是整数。

如果有多个解，输出前齿轮齿数最小的那一个（第一个齿轮齿数最小的，若第一个齿轮齿数相等，输出第二个齿轮齿数最小的……依此类推）。如果所有的前齿轮齿数都相等，照着上面的办法处理后齿轮（其实就是把第一个，第二个……齿轮分别设为第一，第二……个关键字来排序）。

SAMPLE INPUT

```
2 5
39 62 12 28
```

SAMPLE OUTPUT

```
39 53
12 13 15 23 27
```

Comment

注释

这个问题最大的挑战就是“读懂题目”。慢慢读，不要想一步登天。如果你读不懂题目，还是得一遍一遍的把它读进去。

问题要我们找出“最佳齿轮组合”，即传动比率最接近平均数的组合。考虑下面的测试数据：

```
2 5
39 62 12 28
```

这意味着有 2 个前齿轮，型号范围在 39..62 ；5 个后齿轮，型号范围在 12..28。程序必须检查所有前齿轮组成的有序对（共 62-39+1=24 种齿轮），和所有后齿轮组成的五元组（共 28-12+1=17 种齿轮）。根据组合数学原理，总共有 24!/22!/2! x 17!/5!/12! = 656,880 种可能（我是这么认为的）。

对于每一种可能，做下面的计算。举个例子来说，对于枚举到的第一种情况：前齿轮是 39 和 40，后齿轮是 12，13，14，15 和 16。

首先，计算所有可能的传动比率：

```
39/12 = 3.250000000000000000
```

```
39/13 = 3.00000000000000000000
39/14 = 2.78571428571428571428
39/15 = 2.60000000000000000000
39/16 = 2.43750000000000000000
40/12 = 3.33333333333333333333
40/13 = 3.07692307692307692307
40/14 = 2.85714285714285714285
40/15 = 2.66666666666666666666
40/16 = 2.50000000000000000000
```

然后，对它们进行排序：

```
39/16 = 2.43750000000000000000
40/16 = 2.50000000000000000000
39/15 = 2.60000000000000000000
40/15 = 2.66666666666666666666
39/14 = 2.78571428571428571428
40/14 = 2.85714285714285714285
39/13 = 3.00000000000000000000
40/13 = 3.07692307692307692307
39/12 = 3.25000000000000000000
40/12 = 3.33333333333333333333
```

然后，计算差的绝对值：

```
2.43750000000000000000 - 2.50000000000000000000 =
0.06250000000000000000
2.50000000000000000000 - 2.60000000000000000000 =
0.10000000000000000000
2.60000000000000000000 - 2.66666666666666666666 =
0.06666666666666666666
2.66666666666666666666 - 2.78571428571428571428 =
0.11904761904761904762
2.78571428571428571428 - 2.85714285714285714285 =
0.07142857142857142857
2.85714285714285714285 - 3.00000000000000000000 =
0.14285714285714285715
3.00000000000000000000 - 3.07692307692307692307 =
0.07692307692307692307
3.07692307692307692307 - 3.25000000000000000000 =
0.17307692307692307693
```

```
3.25000000000000000000 - 3.33333333333333333333 =
0.08333333333333333333
```

然后计算平均数和方差。

平均数是（我是这么认为的）0.09953703703703703703703666666.。方差大约是 0.00129798488416722。 找出使方差最小的齿轮组合。

问题分析

- 最大传动比率至少是最小的 3 倍。这个其实不用算出比率在判断，只要不满足 s1[F]/s2[1]-s1[1]/s2[R]>=3 的都剪掉 (s1 表示前齿轮型号，s2 表示后齿轮型号)。另外乘法计算要比除法快得多，上面判断式可以写成 s1[F]*s2[R]>=3*s1[1]*s2 [1]。这个剪枝很有效果。
- 改变求方差的方法。
- 当找到比当前更优秀的解的时候，不要用 For 循环一个一个复制当前解，用 memcpy 函数会更快。