**题目1：管道系统**

时间限制: 5000ms 内存限制: 256MB

**描述**

你正在参与一项输油管道的项目，此项目的目标是架设一组输油管道系统，使得从源头到目的地输送的石油流量最大。如果直接从源头架设管道到目的地，一旦管道有损坏，将无法输送石油。为了增强管道系统抵抗破坏的能力，你决定设置 N - 2 个中间节点，从而把管道分散开来。

源头和目的地可以视为特殊的节点，他们的编号分别为 1 和 N，而其他的中间节点的标号为 2 到 N - 1。每条管道有特定的流量限制 c，石油可以以不超过 c 的速率在管道内流动。两个节点之间至多由一条管道连接，也就是说这两个节点之间单位时间内流动的石油不超过 c。

现在你们已经选定了 N 个节点，并且制定好了建设计划。你想知道，每当一条管道建造完成，从节点 1 到节点 N 最大流量能够增加多少。

**输入**

输入包括多组数据。第一行是整数 T，表示有多少组测试数据。每组测试数据的第一行包括两个整数 N 和 M，表示节点的数量和将要建造的管道数量。接下来 M 行按建造顺序描述每条管道，每行包括三个整数 i, j, c，表示要在节点 i 和 j 之间建造一条容量为 c 的管道。

**输出**

对于每组测试数据，先输出一行“Case #c:”，其中 c 为测试数据编号（从 1 开始）。紧接着按建造顺序输出使得总流量增加的管道编号和相应的增加量，以一个空格隔开。

**数据范围**

1 ≤ T ≤ 100

0 ≤ M ≤ N(N - 1)/2

1 ≤ v ≠ u ≤ N

1 ≤ c ≤ 100

小数据：2 ≤ N ≤ 20

大数据：2 ≤ N ≤ 200

**样例输入**

3

2 1

1 2 3

4 5

1 2 9

2 3 7

3 4 6

2 4 4

1 3 1

5 7

1 2 1

1 3 2

1 4 3

2 5 4

3 2 3

4 3 3

3 5 2

**样例输出**

Case #1:

1 3

Case #2:

3 6

4 3

5 1

Case #3:

4 1

5 2

6 1

7 2

**解题思路**

此题要求每当向网络增加一条边时，计算网络总流量的增加量。一种容易想到的方法是：每当增加一条边时，就更新一次最大流。由于更新最大流的时间可达O(M)，所以这种方法的时间复杂度为O(M^2)，很有可能超时。事实上，并不是每次增加新边时都需要更新最大流：设当前网络的割集为S-T，只有当新边“跨越”S和T时，才有可能增大流量。所以只需要一点额外开销维护割集就可以把更新最大流的此时降低到O(F)（F为最大流量），总的时间就降为O(MF)。

**题目2：仙剑5前传之软星包子**

时间限制: 1000ms 内存限制: 256MB

**描述**

仙剑是一款经典的RPG游戏，最近又推出了仙剑5前传。Alice身为忠实的仙剑粉丝，当然是在第一时间就开始玩了。迷宫以及各类小游戏是仙剑系列的一大传统，这次也不例外。而且还增加了称号系统，玩家如果在满足一定条件下通过迷宫或是完成小游戏，都可以获得相应的称号奖励。Alice虽然智商也不算太低，顺利的完成游戏还是没什么问题的，但是某些称号对于她来说好像比较困难，所以她来找你帮忙。

仙剑5前传中增加了软星包子这种很萌的NPC，它散落在世界各地，你每找到一个，都可以挑战他，挑战成功可以拿到一些卡片。挑战具体说来就是一个记忆力游戏，有 N 种卡片，每种两张，总共 2N 张卡片面朝下放置。玩家可以点击面朝下的卡片并将其翻面。当有两张卡片正面朝上时，如果这两张卡片是同一种，则将这两张卡片收走，否则的话，再将这两张卡片面朝下放置在原位。

Alice的记忆力很差，只能记住 K 张卡片（包括刚刚翻开的）。她的策略如下：

\* 当没有卡片正面朝上时：

   \* 如果她记住的卡片中有两张是同一种的，那就翻开那两张卡片；

   \* 否则，她随机的翻开一张她不知道的卡片。

\* 当有一张卡片正面朝上时：

   \* 如果她记得另一张同样的卡片在哪里，就直接翻开那张；

   \* 否则，她随机的翻开一张她不知道的卡片。

Alice想知道，她期望要点多少次才可以完成这个小游戏。

**输入**

输入数据的第一行包含一个整数 T，表示数据组数。

接下来有 T 组数据，每组数据中有一行，包含两个整数 N, K。

**输出**

对于每组数据，输出一行"Case #X: Y"，其中 X 为数据组数编号，从 1 开始，Y 为期望的步数。

绝对误差在1e-6以内或相对误差在1e-8以内即可。

**数据范围**

1 ≤ T ≤ 10

1 ≤ K ≤ 10

小数据：1 ≤ N ≤ 100

大数据：1 ≤ N ≤ 100000

**提示**

下面是一种可能的游戏过程，用来解释Alice可以记住 K 张卡片这一点：

假设N = 5, K = 3

|  |  |  |  |
| --- | --- | --- | --- |
| **已经翻开的卡片** | **刚刚翻开的卡片** | **Alice记住的卡片** | **解释** |
| - | A0 | A0 |  |
| A0 | B0 | A0, B0 |  |
| - | C0 | A0, B0, C0 |  |
| C0 | B1 | B0, C0, B1 |  |
| - | B0, B1 | C0, B0, B1 | B0和B1都在记忆中，直接翻开移除 |
| - | A1 | C0, A1 |  |
| A1 | D0 | C0, A1, D0 |  |
| - | E0 | A1, D0, E0 |  |
| E0 | E1 | D0, E0, E1 | 碰巧翻到两张一样的，E0和E1移除 |
| - | D1 | D0, D1 |  |
| D1 | D0 | D1, D0 | Alice记得D0，D0和D1移除 |
| - | A1 | A1 |  |

继续……

**样例输入**

2

2 1

3 3

**样例输出**

Case #1: 8.000000

Case #2: 9.000000

**解题思路**

这道题很容易可以发现，状态应为f[i][j],其中i表示还剩几对牌没有翻开，j表示当前记住了几张牌。其中记住的几张牌中一定没有相同的，不然就直接翻开即可。状态总数为N\*K。讨论转移时的情况：

第一张一定是从不记得的牌中翻开，翻开后又两种可能：

1.和记忆中的一张一样，那就直接翻开记忆中的那张，消除一对

2.和记忆中的都不一样，那就再随便翻开一张不记得的，可能性有：

a.恰巧两张一样，直接消除

b.和记忆中的一张一样，那么这次未消除，但是接着一定是连续翻开这两张，消除一对

c.和之前的都不一样

转移的时候，只有在2c的情况中，可能出现到自己状态的转移，需要解方程得出，其它都可以直接递推得出。

**题目3：集会**

时间限制: 2000ms 内存限制: 256MB

**描述**

在一条河的一侧，分布着 N 个村庄。这些村庄平日里需要一些贸易往来，然而商人们来回走遍每一座村庄是非常辛苦的，于是他们决定每个月都在河边举行一次集会，大家都来集会上购买需要的物品。然而在集会地点的选择上，大家却有分歧，因为谁都不愿意集会的地点离自己村庄非常远。经过一番激烈的讨论之后，大家决定要将集会地点挑选在这样一个位置：它离最远的村庄的距离要尽可能的近。

我们把河看做一条足够长的直线，河岸就是平面坐标系上 y = 0 的这条线，y < 0 的区域是河水，而所有村庄都在 y ≥ 0 的区域里。现在给出所有村庄的平面坐标，你要在河岸上找到这样一个位置，使得它到所有村庄的最远距离最小。

**输入**

输入文件包含多组测试数据。

第一行，给出一个整数 T，为数据组数。接下来依次给出每组测试数据。

每组数据的第一行是一个整数 N，表示村庄的数量。接下来 N 行，每行有两个实数 xi 和 yi，表示每一个村庄的坐标。

**输出**

对于每组测试数据，输出一行"Case #X: Y"，其中 X 表示测试数据编号，Y 表示集会地点的 x 坐标值，要求与正确答案的绝对误差在10-6以内。所有数据按读入顺序从 1 开始编号。

**数据范围**

小数据：T ≤ 100, 0 < N ≤ 50, 0 ≤ |xi|, yi ≤ 10000

大数据：T ≤ 10, 0 < N ≤ 50000, 0 ≤ |xi|, yi ≤ 10000

**样例输入**

1

5

0 8

1 6

4 4

-5 7

-6 1

**样例输出**

Case #1: -1.000000

**解题思路**

这道题目有很多种解法，这里列出两种：

首先，如果我们容易设计出一个算法来判断，是否存在一个x轴上的点，它到左右村庄的距离不超过r。方法是，对每个村庄都计算出x轴上的一个区间，使得区间中的点到该村庄距离不超过r，然后对所有村庄的坐标区间求交集，交集中任意一点都是满足条件的集会地点。有了这个算法，我们便可以用二分法确定出这个r值，并且此时所有村庄的区间交集必定为一个点，输出这个点的坐标即可。

第二种解法，我们先将所有村庄关于x轴作对称，然后我们求出所有这些点的最小圆覆盖。由于所有点关于x轴对称，因此解出来的圆的圆心一定会在x轴上，此时圆心的位置即为该问题的解，圆的半径r为圆心到最远村庄的距离。为什么它就是最优解呢？我们可以假设在x轴上存在另外一点更优，即它到所有村庄的最远距离r'<r，那么它到所有对称点的距离也同样为r'，那么我们能得到一个更小的圆覆盖所有点，这显然与最小圆覆盖的解的定义矛盾。