NOIP 模拟题

第 14 套

题目名称	消除游戏	绝地求生	贪心算法
题目类型	传统型	传统型	传统型
目录	elim	battleground	greedy
可执行文件名	elim	battleground	greedy
输入文件名	elim.in	battleground.in	greedy.in
输出文件名	elim.out	battleground.out	greedy.out
每个测试点时限	1.0 秒	4.0 秒	2.0 秒
内存限制	512 MB	512 MB	512 MB
测试点/包数目	10	20	10
测试点是否等分	是	是	是

提交源程序文件名

对于 C++ 语言	elim.cpp	battleground.cpp	greedy.cpp
对于 C 语言	elim.c	battleground.c	greedy.c
对于 Pascal 语言	elim.pas	battleground.pas	greedy.pas

编译选项

对于 C++ 语言	-O2 -std=c++14
对于 C 语言	-O2 -std=c14
对于 Pascal 语言	-O2

消除游戏（elim）

【题目描述】

消除类游戏是深受大众欢迎的一种游戏，游戏在一个包含有 n 行 m 列的游戏棋盘上进行，棋盘的每一行每一列的方格上放着一个有颜色的棋子，当一行或一列上有连续三个或更多的相同颜色的棋子时，这些棋子都被消除。当有多处可以被消除时，这些地方的棋子将同时被消除。现在给你一个 n 行 m 列的棋盘，棋盘中的每一个方格上有一个棋子，请给出经过一次消除后的棋盘。请注意：一个棋子可能在某一行和某一列同时被消除。

【输入格式】

从文件 ***elim.in*** 中读入数据。

输入的第一行包含两个整数 n,m，用空格分隔，分别表示棋盘的行数和列数。满足：$1 \le n,m \le 30$。接下来 n 行，每行 m 个整数，用空格分隔，分别表示每一个方格中的棋子的颜色。颜色使用 1 至 9 编号。

【输出格式】

输出到文件 ***elim.out*** 中。

输出 n 行，每行 m 个整数，相邻的整数之间使用一个空格分隔，表示经过一次消除后的棋盘。如果一个方格中的棋子被消除，则对应的方格输出 0，否则输出棋子的颜色编号。

【样例 1 输入】

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

【样例 1 输出】

```
2 2 3 0 2
3 4 5 0 4
2 3 2 0 3
0 0 0 4 4
```

【样例 1 解释】

棋盘中第 4 列的 1 和第 4 行的 2 可以被消除，其他的方格中的棋子均保留。

【样例 2 输入】

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

【样例 2 输出】

```
2 2 3 0 2
3 0 0 0 0
2 3 2 0 3
2 2 0 0 0
```

【样例 2 解释】

棋盘中所有的 1 以及最后一行的 3 可以被同时消除，其他的方格中的棋子均保留。

绝地求生（battleground）

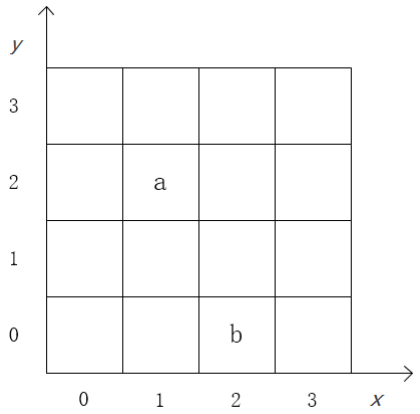
【题目描述】

《绝地求生》是一款战术竞技型射击类沙盒游戏，玩家需要在游戏地图上收集各种资源，并在不断缩小的安全区域内对抗其他玩家，让自己生存到最后。

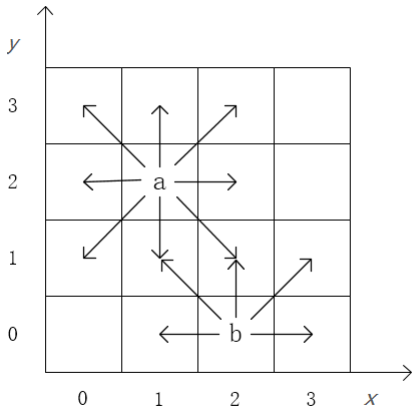
本题简化了游戏规则，需要你计算出最终的游戏结果，简化版规则如下。

【游戏规则】

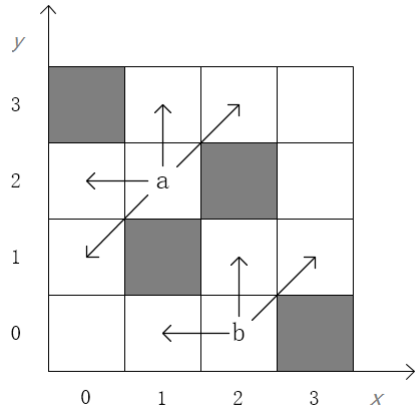
游戏地图是 $n \times n$ 的正方形棋盘，由 1×1 的方格组成，每个玩家用一个 1×1 的方格表示。



在不超出棋盘边界的情况下，玩家可以向八个方向（上、下、左、右、左上、左下，右上、右下）移动，进入周围的格子，一次移动称为一步。下图示意性地给出了玩家 a 和玩家 b 可能的移动方向，由于玩家 b 位于棋盘的边缘，因此可能的移动方向仅有 5 种。



棋盘中可能有**障碍物**，障碍物也是 1×1 的方格，玩家不能进入障碍物的方格，也不能穿越两个斜向相邻障碍物方格的间隙。



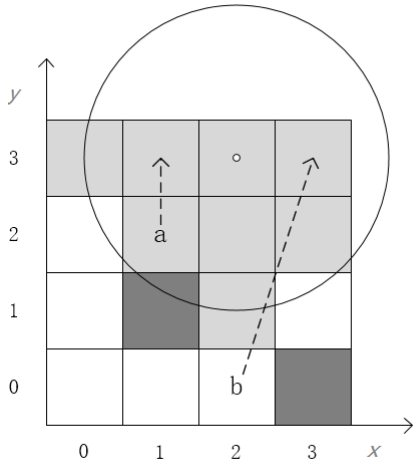
不同玩家之间**互不影响**，他们可以出现在一个方格里面。

【游戏流程】

游戏开始时，会给出棋盘的大小、玩家数量、障碍物数量、每个障碍物的位置、每个玩家的初始位置。所有的玩家在游戏开始时，都会被赋予相等的“生命值”。一次游戏分为多个回合，在游戏开始时，会给出本次游戏的回合数目。

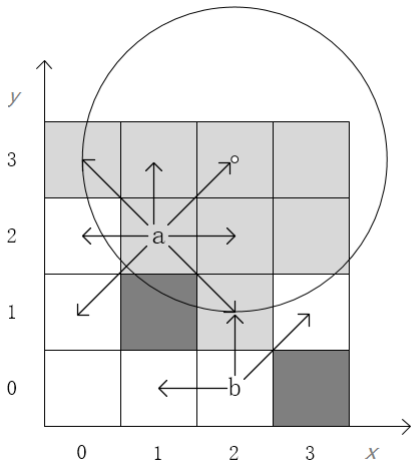
每回合开始时，都会给出每个玩家的目标位置。在这一回合内，玩家需要从上一回合结束时的位置（对于第一回合则为初始位置）移动到这一回合的目标位置，移动的步数不限。如果玩家在这一回合的起始位置和某一障碍物重合，那么假定在这一回合内，该障碍物对于**该玩家**是失效的。

下图中给出了某一回合开始时，玩家 a 的位置 $(1,2)$ 和目标位置 $(1,3)$，以及玩家 b 的位置 $(2,0)$ 和目标位置 $(3,3)$。



每一回合内，都会出现大小、位置都固定不变的一个圆形的安全区域，直到本回合结束。安全区域的圆心位于方格中心，如果某个方格的中心到圆心的**直线距离**小于或等于安全区域的半径，那么这个方格就是**安全的**。从不安全的方格移动一步到其他位置会被**扣除 1 点生命值**。安全的方格内的障碍物将会**在本回合失效**，允许玩家通过。所有玩家的目标位置保证是安全的。

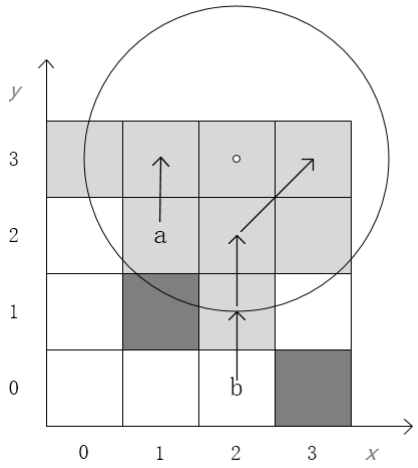
图中圆心的方格坐标是 $(2,3)$，半径为 2，浅灰色的方格是安全的方格，安全的方格内的障碍物会在本回合失效。



你的任务是，为每一位玩家找到**生命值扣除最少**的移动路线。若对于某位玩家，任何到达本回合目标位置的移动路线都会导致生命值扣除至 0，则称该玩家**死亡**。死亡的玩家不参与之后回合的游戏。

下图展示生命值扣除最少的移动路径，在此过程中，玩家 `a` 不被扣除生命值，玩家 `b` 被扣除 1 点生命值。

所有回合结束后，你需要输出所有玩家剩余的生命值，已经死亡的玩家输出 `0`。



【输入格式】

从文件 ***battleground.in*** 中读入数据。

输入第一行包括五个整数：n、m、e、f、h，表示棋盘为 $n\times n$ 大小，一共有 m 个玩家，棋盘上有 e 个障碍物，游戏一共有 f 个回合，玩家的初始生命值是 h。$1\le n\le 400$，$1\le m\le 10^5$，$0\le e\le n\times n$，$1\le f\le 10$，$0<h\le 2.5n$。

接下来有 e 行，每行包含两个整数 p、q，(p,q) 即为该障碍物所在方格的坐标，$0\le p,q<n$。

随后有 m 行，每行包含两个整数 i、j，(i,j) 即为该玩家初始所在方格的坐标，$0\le i,j<n$。

随后有 f 个回合的数据，每个回合的数据有 $m+1$ 行。其中，包含一行安全区信息以及 m 行玩家移动目标。安全区信息包括三个整数，a、b 和 r，(a,b) 表示安全区圆心所在方格的坐标，r 表示安全区半径。玩家移动目标包含两个整数，u 和 v，(u,v) 表示玩家移动目标的方格坐标。即使玩家已经死亡，也会提供移动目标，但是并不需要进行计算。$0\le a,b,u,v<n$，$0<r\le 200$。保证每个玩家给出的目标坐标一定在安全区域以内。保证在任意回合，对于任意玩家，都存在一条到达本回合目标位置的移动路线。

所有输入都是整数。

【输出格式】

输出到文件 ***battleground.out*** 中。

输出 m 行，每行包含一个整数：z，表示该玩家的最终生命值。

m 个玩家的输出顺序与输入顺序相同。

【样例输入】

```
4 2 4 1 1
0 3
1 1
2 2
3 0
1 2
2 0
2 3 2
1 3
3 3
```

【样例输出】

```
1
0
```

【子任务】

- 子任务一（4 组数据）：输入数据保证无障碍物，每回合开始玩家都在安全区中；
- 子任务二（4 组数据）：输入数据保证无障碍物，每回合开始玩家不一定在安全区中，$1 \le n \le 10$，$1 \le m \le 20$；
- 子任务三（4 组数据）：输入数据保证无障碍物，每回合开始玩家不一定在安全区中，$10 < n \le 400$，$20 < m \le 10^5$；
- 子任务四（4 组数据）：输入数据保证有障碍物，$1 \le n \le 10$，$1 \le m \le 20$；
- 子任务五（4 组数据）：输入数据保证有障碍物，$10 < n \le 400$，$20 < m \le 10^5$。

贪心算法（greedy）

【题目描述】

点独立集是图论中的概念。一个点独立集是一个图中一些两两不相邻的顶点的集合，亦即一个由顶点组成的集合 S，使得 S 中任两个顶点之间没有边。顿顿设计了一个在无向图上求解点独立集的算法，希望你可以帮他实现一下。

算法框架

1. 对于给定的无向图，不断地使用“归约规则”缩减图的规模，直至无法继续使用为止。
2. 当无法使用归约规则时，每次“贪心”地选取一个顶点直接从图中删去，直至能继续使用归约规则或图为空。
3. 反复迭代上述步骤，直至图为空。

归约规则

每成功地执行一次规约规则，会将一个顶点选入答案中，选入的顶点按下面的规则唯一确定：

1. 若图中有顶点度为 0，则将其中编号最小的选入答案中，并把它从图中删去；
2. 否则若图中有顶点度为 1，则将其中编号最小的选入答案中，并把它和它唯一的邻接顶点从图中删去；
3. 否则不能成功执行规约规则。

贪心策略

当图中不存在度小于 2 的顶点时，需要从图中贪心地删去一个顶点，被删去的顶点按下面的策略唯一确定：

1. 若图中度最大的顶点唯一，则把它从图中删去；
2. 否则，在上述顶点中，选择这样的顶点，使得删去它之后，图中剩余的度为 1 的顶点最多。若这样的顶点唯一，则把它从图中删去；
3. 否则，在这样的顶点中，选择编号最大的那个从图中删去。

【输入格式】

从文件 ***greedy.in*** 中读入数据。

输入第一行包含用空格分隔的两个正整数 n 和 m，表示图中有 n 个顶点、m 条无向边，顶点编号从 1 到 n。

接下来 m 行，每行包含用空格分隔的两个正整数 u 和 v，表示编号为 u 和 v 的两个顶点间有一条无向边。输入数据保证所有的顶点编号（u、v 和 w）均为 $[1,n]$ 范围内的正整数，保证 $u \neq v$ 且同一条边不会出现多次。

【输出格式】

输出到文件 ***greedy.out*** 中。

按求解顺序输出该点独立集（即每成功地执行归约规则一次就输出一个被选入的顶点），每行输出一个顶点编号。

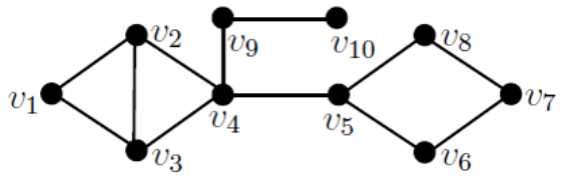
【样例输入】

```
10 12
1 2
2 3
3 1
2 4
3 4
4 9
4 5
9 10
5 8
8 7
7 6
6 5
```

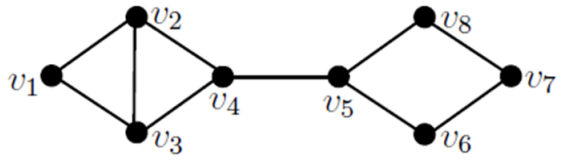
【样例输出】

```
10
6
8
1
4
```

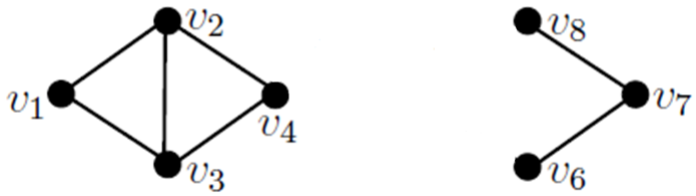
【样例解释】



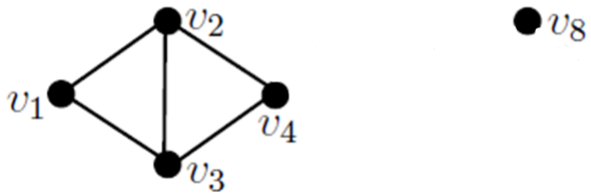
输出 v_{10}，删去 v_{10}、v_9。



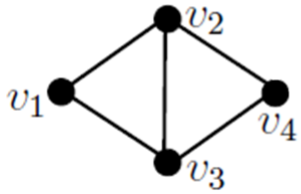
删去 v_5。



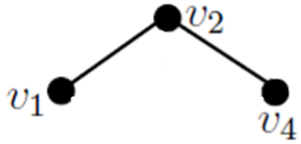
输出 v_6，删去 v_6、v_7。



输出 v_8，删去 v_8。



删去 v_3。



输出 v_1，删除 v_1、v_2。

输出 v_4，删除 v_4。

【子任务】

子任务一（2 组数据）：输入数据保证 $n \le 2,000$ 且 $m \le 10^5$，且图中没有环。

子任务二（2 组数据）：输入数据保证 $n \le 2,000$ 且 $m \le 10^5$，且对于任意三个不同顶点 u、v 和 w，如果 u 和 v 之间有边且 v 和 w 之间有边，则 u 和 w 之间有边。

子任务三（2 组数据）：输入数据保证 $n \le 2,000$ 且 $m \le 10^5$。

子任务四（4 组数据）：输入数据保证 $n \le 10^5$ 且 $m \le 5 \times 10^5$。

【提示】

本题输入输出数据量较大，请选择合理方式进行读写。