# A 礼物

时空限制：2s 256MB

### 文件名

gift.in/gift.out/gift.cpp

## 题意

Farmer John 为他的 $N$ 头编号为 $1 \ldots N$ 的奶牛准备了 $N$ 个同样编号为 $1 \ldots N$ 的礼物 $(1 \leq N \leq 500)$ 。每头奶牛都有一个愿望 单, 是所有 $N$ 个礼物的一个排列, 奶牛相比序列中较晚出现的礼物更喜欢序列中较早出现的礼物。 FJ 很懒, 只是对于所有 $i$ 将礼物 $i$ 分配给奶牛 $i$ 。现在, 奶牛们自行聚集在一起并决定重新分配礼物, 使得在重新分配后, 每头奶 牛最终得到的是她最初得到的礼物, 或者比她最初得到的礼物更喜欢的礼物。 对于 1 到 $N$ 中的每一个 $i$, 计算奶牛 $i$ 在重新分配后有希望收到的最喜欢的礼物。

## 输入格式

输入的第一行包含 $N$ 。以下 $N$ 行每行包含一头奶牛的愿望单。输入保证每行均为 $1 \ldots N$ 的一个排列。

## 输出格式

输出 $N$ 行, 其中第 $i$ 行包含奶牛 $i$ 在重新分配后有希望收到的最喜欢的礼物。

## 输入样例

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

## 输出样例

```
1
3
2
4
```

## 样例解释

在这个例子中, 有两种可能的重新分配方式:

- 初始的分配方式：奶牛 1 收到礼物 1, 奶牛 2 收到礼物 2, 奶牛 3 收到礼物 3, 奶牛 4 收到礼物 4。
- 奶牛 1 收到礼物 1, 奶牛 2 收到礼物 3, 奶牛 3 收到礼物 2, 奶牛 4 收到礼物 4。 观察到奶牛 1 和 4 均不可能收到比她们最初得到的礼物更好的礼物。然而, 奶牛 2 和 3 可以。

## 数据范围限制

$20\%$的测试数据满足 $N \leq 8$

# B 机器人

时空限制：4s 512MB

### 文件名

robot.in/robot.out/robot.cpp

## 题意

Bessie 正在学习如何控制她最近作为礼物收到的机器人。

机器人从坐标平面上的点 $(0,0)$ 开始移动, 而 Bessie 希望机器人移动到点 $(x_g, y_g)$ 。Bessie 初始时有一个包含 $N(1 \leq N \leq 40)$ 条 指令的指令列表可以用于控制机器人, 其中第 $i$ 条指令可以令机器人向右移动 $x_i$ 个单位, 向上移动 $y_i$ 个单位 (或向左、向下, 当 $x_i$ 和 $y_i$ 分别为负数时) 。

对于从 1 到 $N$ 中的每一个 $K$, 帮助 Bessie 计算她可以从原来的 $N$ 条指令中选择 $K$ 条的方法数, 使得在执行这 $K$ 条指令后, 机器 人将移动到点 $(x_g, y_g)$ 。

## 输入格式

输入的第一行包含 $N$。。第二行包含 $x_g$ 和 $y_g$, 均在 $-10^9 \ldots 10^9$ 范围内。最后 $N$ 行表示指令。每行包含两个整数 $x_i$ 和 $y_i$, 同样在 $-10^9 \ldots 10^9$ 范围内。

输入保证 $(x_g, y_g) \neq (0,0)$ 以及对于所有的 $i$ 有 $(x_i, y_i) \neq (0,0)$ 。

## 输出格式

输出 $N$ 行, 对 $1$ 到 $N$ 中的每一个 $K$ 输出 Bessie 可以从原来的 $N$ 条指令中选择 $K$ 条的方法数。

## 输入样例

```
7
5 10
-2 0
3 0
4 0
5 0
0 10
0 -10
0 10
```

## 输出样例

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

## 样例解释

在这个例子中, Bessie 有六种选择指令的方法:

```
(-2,0) (3,0) (4,0) (0,10) (0,-10) (0,10) (1 2 3 5 6 7)
(-2,0) (3,0) (4,0) (0,10) (1 2 3 5)
(-2,0) (3,0) (4,0) (0,10) (1 2 3 7)
(5,0) (0,10) (0,-10) (0,10) (4 5 6 7)
(5,0) (0,10) (4 5)
(5,0) (0,10) (4 7)
```

对于第一种方法, 机器人的移动路径如下:

$$(0,0) \to (-2,0) \to (1,0) \to (5,0) \to (5,10) \to (5,0) \to (5,10)$$

## 数据范围限制

$20\%$的测试数据满足 $N \leq 20$

# C 邮件

时空限制：2s 256MB

### 文件名

mail.in/mail.out/mail.cpp

## 题意

Farmer John 在整理收件箱方面已经落后了。他的屏幕排列方式是, 屏幕左侧有一个垂直的文件夹列表, 屏幕右侧有一个垂直的电子邮件列表。共有 $M$ 个文件夹, 编号为 $1 \ldots M$ $\left(1 \leq M \leq 10^4\right)$ 。他的收件箱当前包含 $N$ 封编号为 $1 \ldots N$ $\left(1 \leq N \leq 10^5\right)$ 的电子 邮件; 第 $i$ 封电子邮件需要归档到文件夹 $f_i$ $\left(1 \leq f_i \leq M\right)$ 。

FJ 的屏幕有些小, 所以他同时只能查看 $K(1 \leq K \leq \min(N, M))$ 个文件夹和 $K$ 封电子邮件。最初, 他的屏幕左侧显示文件夹 $1 \ldots K$, 右侧显示电子邮件 $1 \ldots K$ 。要访问其他文件夹和电子邮件, 他需要滚动相应的列表。例如, 如果他将文件夹列表向下滚动 一个位置, 他的屏幕将显示文件夹 $2 \ldots K+1$, 然后再向下滚动一个位置将显示文件夹 $3 \ldots K+2$ 。当 FJ 将一封电子邮件拖曳至 一个文件夹时, 该电子邮件将从电子邮件列表中消失, 并且消失的电

子邮件之后的电子邮件会向上移动一个位置。例如, 如果当前 显示的是电子邮件 $1, 2, 3, 4, 5$, 然后 FJ 将电子邮件 3 拖曳至其对应的文件夹中, 则电子邮件列表现在将显示电子邮件 $1, 2, 4, 5, 6$ 。 FJ 只可以将电子邮件拖曳至其需要归档的文件夹中。

不幸的是, FJ 的鼠标滚轮坏了, 所以他只能向下滚动, 不能向上滚动。他可以实现类似向上滚动的唯一情况是, 他正在查看他的 电子邮件列表中的最后 $K$ 封电子邮件, 并且他归档了其中的一封。在这种情况下, 电子邮件列表将再继续显示尚未归档的最后 $K$ 封电子邮件, 实际上使得最上方的电子邮件向上滚动了一个位置。如果剩余少于 $K$ 封电子邮件, 则将显示所有电子邮件。

请帮助 FJ 判断是否可能归档他的所有电子邮件。

## 输入格式

输入的第一行包含 $T(1 \leq T \leq 10)$, 为当前测试用例中的子测试用例数量, 所有子测试用例必须全部正确才能通过当前测试用例。以下是 $T$ 个子测试用例。每一个子测试用例的第一行包含 $M, N$ 和 $K$ 。第二行包含 $f_1 \ldots f_N$ 。

输入保证所有子测试用例的 $M$ 之和不超过 $10^4$, 所有子测试用例的 $N$ 之和不超过 $10^5$ 。

## 输出格式

输出 $T$ 行, 每行包含 YES 或 NO, 表示对于 $T$ 个子测试用例中的每一个, FJ 是否可以成功归档他的所有电子邮件。

## 输入样例

```
6
5 5 1
1 2 3 4 5
5 5 1
1 2 3 5 4
5 5 1
1 2 4 5 3
5 5 2
1 2 4 5 3
3 10 2
1 3 2 1 3 2 1 3 2 1
3 10 1
1 3 2 1 3 2 1 3 2 1
```

## 输出样例

```
YES
YES
NO
YES
YES
NO
```

## 数据范围限制

$80\%$的测试数据满足 $M$ 之和不超过 $10^3$ 。