最罕见的昆虫（insects）

Pak Blangkon 的房子四周有 N 只昆虫，编号为 0 至 $N-1$。 每只昆虫有一个**类型**，以从 0 至 10^9（包含 0 和 10^9）的整数编号。 可能有多只昆虫类型相同。

假设将昆虫按照类型分组。 我们定义**最常见**昆虫类型的基数是昆虫最多的分组中的昆虫数。 类似地，**最罕见**昆虫类型的基数是昆虫最少的分组中的昆虫数。

例如，假设有 11 只昆虫，类型分别为 $[5, 7, 9, 11, 11, 5, 0, 11, 9, 100, 9]$。 在此情形中，**最常见**昆虫类型的基数是 3，是因为类型 9 和类型 11 的分组均有最多数目的昆虫，每个分组都有 3 只。 **最罕见**昆虫类型的基数是 1，是因为类型 7、类型 0 和类型 100 的分组均有最少数目的昆虫，每个分组都有 1 只。

Pak Blangkon 不知道这些昆虫的类型。 他有一台单按钮的机器，可以提供昆虫类型相关的信息。 刚开始时，机器是空的。 在使用机器时，可以做如下三种操作：

1. 将一只昆虫放进机器。
2. 将一只昆虫取出机器。
3. 按下机器的按钮。

每种操作最多可以做 40 000 次。

每当按下按钮时，机器会报告在机器内的**最常见**昆虫类型的基数。

你的任务是使用上述机器，确定 Pak Blangkon 的房子四周所有 N 只昆虫中**最罕见**昆虫类型的基数。 此外，在某些子任务里，你的得分取决于机器执行某种操作的最大次数（详见子任务一节）。

实现细节

你要实现以下函数：

```
int min_cardinality(int N)
```

- N：昆虫数量。
- 此函数应返回 Pak Blangkon 的房子四周所有 N 只昆虫中**最罕见**昆虫类型的基数。
- 此函数恰好被调用一次。

该函数可调用以下几个函数：

```
void move_inside(int i)
```

- i：将被放进机器的昆虫编号。编号 i 的取值范围是 0 至 $N-1$（包含 0 和 $N-1$）。
- 如果昆虫已在机器内，函数调用不会影响机器内昆虫的集合。但是，调用仍会被计入此类操作的次数。
- 此函数最多可以被调用 $40\,000$ 次。

```
void move_outside(int i)
```

- i：将被从机器中取出的昆虫编号。编号 i 的取值范围是 0 至 $N-1$（包含 0 和 $N-1$）。
- 如果昆虫已在机器外，函数调用不会影响机器内昆虫的集合。但是，调用仍会被计入此类操作的次数。
- 此函数最多可以被调用 $40\,000$ 次。

```
int press_button()
```

- 此函数返回机器内**最常见**昆虫类型的基数。
- 此函数最多可以被调用 $40\,000$ 次。
- 评测程序**不是适应性**的。也就是说，所有 N 只昆虫的类型在 `min_cardinality` 调用前已经确定。

例子

考虑在某个场景下，有 6 只类型分别为 $[5, 8, 9, 5, 9, 9]$ 的昆虫。 函数 `min_cardinality` 的调用方式如下：

```
min_cardinality(6)
```

此函数按以下次序调用了 `move_inside`、`move_outside` 和 `press_button`。

函数调用	返回值	机器内的昆虫	机器内的昆虫类型
		$\{\}$	$[\,]$
`move_inside(0)`		$\{0\}$	$[5]$
`press_button()`	1	$\{0\}$	$[5]$
`move_inside(1)`		$\{0,1\}$	$[5,8]$
`press_button()`	1	$\{0,1\}$	$[5,8]$
`move_inside(3)`		$\{0,1,3\}$	$[5,8,5]$
`press_button()`	2	$\{0,1,3\}$	$[5,8,5]$
`move_inside(2)`		$\{0,1,2,3\}$	$[5,8,9,5]$
`move_inside(4)`		$\{0,1,2,3,4\}$	$[5,8,9,5,9]$
`move_inside(5)`		$\{0,1,2,3,4,5\}$	$[5,8,9,5,9,9]$
`press_button()`	3	$\{0,1,2,3,4,5\}$	$[5,8,9,5,9,9]$
`move_inside(5)`		$\{0,1,2,3,4,5\}$	$[5,8,9,5,9,9]$
`press_button()`	3	$\{0,1,2,3,4,5\}$	$[5,8,9,5,9,9]$
`move_outside(5)`		$\{0,1,2,3,4\}$	$[5,8,9,5,9]$
`press_button()`	2	$\{0,1,2,3,4\}$	$[5,8,9,5,9]$

至此，已有充分信息表明，最罕见昆虫类型的基数是 1。因此，函数 `min_cardinality` 应返回 1。

在这个例子里，`move_inside` 被调用 7 次，`move_outside` 被调用 1 次，`press_button` 被调用 6 次。

约束条件

- $2 \leq N \leq 2000$

子任务

1. （10 分）$N \leq 200$
2. （15 分）$N \leq 1000$
3. （75 分）没有额外的约束条件。

如果在某个测试用例上，函数 `move_inside`、`move_outside` 或 `press_button` 的调用次数不符合“实现细节”中给出的约束条件，或者 `min_cardinality` 的返回值不正确，你的解答在此子任务上得分为 0。

令 q 为以下三个值的**最大值**：`move_inside` 的调用次数、`move_outside` 的调用次数、`press_button` 的调用次数。

在子任务 3 中，你可能会得部分分。令 m 为此子任务所有测试用例的 $\frac{q}{N}$ 的最大值。你在此子任务的得分将根据以下表格计算：

条件	得分
$20 < m$	0（CMS 报告“`Output isn't correct`”）
$6 < m \leq 20$	$\frac{225}{m-2}$
$3 < m \leq 6$	$81 - \frac{2}{3}m^2$
$m \leq 3$	75

评测程序示例

令 T 是长度为 N 的整数数组，其中 $T[i]$ 是编号为 i 的昆虫的类型。

评测程序示例按以下格式读取输入：

- 第 1 行：N
- 第 2 行：$T[0]\ T[1]\ \ldots\ T[N-1]$

如果评测程序示例检测到非法行为，评测程序示例将输出 `Protocol Violation: <MSG>`，其中 `<MSG>` 为如下某种类型：

- `invalid parameter`：在函数调用 `move_inside` 或 `move_outside` 时，参数 i 的值不在 0 至 $N-1$ 的范围内（包括 0 和 $N-1$）。
- `too many calls`：函数 `move_inside`、`move_outside` 或 `press_button` 中**某个**的调用次数超过 40 000 次。

否则，评测程序示例按以下格式输出：

- 第 1 行：`min_cardinality` 的返回值
- 第 2 行：q