最も珍しい昆虫

Pak Blangkon の家の周りでは，0 から $N-1$ までの番号が付けられた N 匹の昆虫が走り回っている．それぞれの昆虫には，0 以上 10^9 以下の整数で表される **種類** が決まっている．2 匹以上の昆虫が同じ種類であるかもしれない．

昆虫たちを種類でグループ分けすることを考える．**最も頻出の**種類の出現頻度を，最も昆虫の数が多いグループにおける昆虫の数と定義する．同様に，**最も珍しい**種類の出現頻度を，最も昆虫の数が少ないグループにおける昆虫の数と定義する．

例えば，種類の値が $[5, 7, 9, 11, 11, 5, 0, 11, 9, 100, 9]$ である 11 匹の昆虫がいる場合を考える．この場合，**最も頻出の**種類の出現頻度は 3 である．**最も頻出の**種類は，共に 3 匹の昆虫が含まれる種類 9 と種類 11 である． **最も珍しい**種類の出現頻度は 1 である．**最も珍しい**種類は，それぞれ 1 匹の昆虫のみが含まれる種類 7，種類 0，および種類 100 である．

Pak Blangkon はどの昆虫の種類も知らない．彼は，昆虫の種類についての情報を与えてくれる機械を持っていて，その機械には1 つのボタンがついている．最初機械は空である．機械を使うために 3 種類の操作をすることができる:

1. 昆虫を 1 匹機械の中に入れる．
2. 昆虫を 1 匹機械から取り出す．
3. 機械についているボタンを押す．

3 種類の操作はそれぞれ 40 000 回まで行うことができる．

ボタンを押すと機械は，機械の中にいる昆虫のみを考えた時の**最も頻出の**種類の出現頻度を報告する．

あなたの課題は，機械を使うことで，Pak Blangkon の家にいる N 匹全ての昆虫の中での**最も珍しい**種類の出現頻度を求めることである．いくつかの小課題では，あなたの得点は各種類の操作を行った回数の最大値に依存する．(詳細は小課題の項を参照．)

実装上の注意

あなたは次の関数を実装する必要がある:

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

- N: 昆虫の数

- この関数は，Pak Blangkon の家にいる N 匹全ての昆虫の中での**最も珍しい**種類の出現頻度を返さなければならない．
- この関数はちょうど 1 回呼び出される．

上の関数は以下の関数を呼び出すことができる．

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

- i: 機械の中に入れる昆虫の番号．i は 0 以上 $N-1$ 以下でなければならない．
- もし指定した昆虫が既に機械の中にいる場合，この呼び出しは効果を持たないが，依然として 1 回の呼び出しとみなされる．
- この関数は $40\ 000$ 回まで呼び出すことができる．

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

- i: 機械から取り出す昆虫の番号．i は 0 以上 $N-1$ 以下でなければならない．
- もし指定した昆虫が既に機械の外にいる場合，この呼び出しは効果を持たないが，依然として 1 回の呼び出しとみなされる．
- この関数は $40\ 000$ 回まで呼び出すことができる．

```
int press_button()
```

- この関数は，機械の中にいる昆虫のみを考えた時の**最も頻出の**種類の出現頻度を報告する．
- この関数は $40\ 000$ 回まで呼び出すことができる．
- 採点プログラムは**適応的ではない**．すなわち，`min_cardinality` が呼び出される前にN 匹すべての種類は決定されている．

入出力例

種類の値がそれぞれ $[5, 8, 9, 5, 9, 9]$ であるような 6 匹の昆虫がいる状況を考える． 関数 `min_cardinality` は次のように呼び出される．

```
min_cardinality(6)
```

この関数は `move_inside`，`move_outside`，`press_button` を次のように呼び出すことができる．

呼び出し	戻り値	機械の中の昆虫	機械の中の昆虫の種類
		$\{\}$	$[]$
`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	$\{0,1,2,3,4\}$	$[5,8,9,5,9]$

この時点で，**最も珍しい**種類の出現頻度は 1 であると結論づけるのに十分な情報が得られている．従って，関数 `min_cardinality` は 1 を返さなければならない．

この例では，`move_inside` は 7 回，`move_outside` は 1 回，そして `press_button` は 6 回呼び出されている．

制約

- $2 \leqq N \leqq 2000$

小課題

1. (10 点) $N \leqq 200$
2. (15 点) $N \leqq 1000$
3. (75 点) 追加の制約はない．

いずれかのテストケースで，`move_inside`，`move_outside`，もしくは `press_button` への呼び出しが実装上の注意の項にある制約を満たさなかった場合，または `min_cardinality` の戻り値が正しくない場合，その小課題におけるあなたの得点は 0 となる．

q を，次の 3 つの値の最大値とする: `move_inside` の呼び出し回数，`move_outside` の呼び出し回数，`press_button` の呼び出し回数

小課題 3 では部分点を得ることができる． m を，小課題中のすべてのテストケースについての $\frac{q}{N}$ の最大値とする． この小課題におけるあなたの得点は次の表に従って計算される．

条件	得点
$20 < m$	0 (CMS 上では "`Output isn't correct`" と表示される)
$6 < m \leqq 20$	$\frac{225}{m-2}$
$3 < m \leqq 6$	$81 - \frac{2}{3}m^2$
$m \leqq 3$	75

採点プログラムのサンプル

T を，$T[i]$ が i 番目の昆虫の種類であるような配列とする．

採点プログラムのサンプルは，以下の形式で入力を読み込む．

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

採点プログラムのサンプルがプロトコル違反を検出した場合，採点プログラムのサンプルの出力は `Protocol Violation: <MSG>` となる．`<MSG>` は次のいずれかである．

- `invalid parameter`: `move_inside` または `move_outside` の呼び出しにおいて i が 0 以上 $N-1$ 以下でない．
- `too many calls`: `move_inside`，`move_outside`，`press_button` の**いずれか**への呼び出し回数が 40 000 を超えている．

その他の場合，採点プログラムのサンプルの出力は以下の形式である．

- 1 行目: `min_cardinality` の戻り値
- 2 行目: q