Automata

Time limit: 2 seconds
Memory limit: 1024 MiB

There is a field consisting of N cells in a row, numbered 0 to $N-1$ from left to right. Each cell has a distinct height: the height of cell i is p_i, and the sequence $p_0, p_1, \ldots, p_{N-1}$ is a permutation of the numbers $0, 1, \ldots, N-1$.

Two cells with **indices** i and j are called *close* if and only if $|i-j| \leq 1$. In particular, every cell is close to itself.

A robot stands on the field, initially placed at some cell. The robot accepts commands, each one of the following two types:

- `MAX`: among all cells close to the robot's current cell, the robot's next position will be at the unique cell with the **maximum** height;
- `MIN`: among all cells close to the robot's current cell, the robot's next position will be at the unique cell with the **minimum** height.

Since a cell is close to itself, a command may leave the robot at the same cell.

A *program* is a finite sequence of commands, where each command is either `MAX` or `MIN`. If the robot starts at cell X and follows program S, it ends at some uniquely determined cell, denoted by $\text{result}(S, X)$.

You will be asked Q queries, each giving you some set of starting cells of size K_i ($0 \leq i < Q$). More specifically the i-th query will be for some cells $x_0, x_1, \ldots, x_{K_i-1}$; the robot will be placed at one of them, but which one is not known in advance. For each query you must determine whether there exists a program that moves the robot to the same final cell, regardless of which starting position from x will be used.

Formally, you must determine if there exists a program S such that:

$$\text{result}(S, x_0) = \text{result}(S, x_1) = \ldots = \text{result}(S, x_{K_i-1}).$$

The permutation p is **the same** for all queries.

Note that you do not need to construct such a program – only to report whether one exists.

Implementation details

The first function you must implement is:

```
void initialize(std::vector<int> p)
```

- p: a permutation of the numbers from 0 to $N-1$.

The second function you must implement is:

```
bool exists_program(std::vector<int> x)
```

- x: an array specifying the cells in strictly increasing order, given for a query.

Function `initialize` is called exactly once, before any calls to `exists_program` function are made.

Function `exists_program` is called Q times, once for each query. It must return `true` if there exists a program after which the robot ends up at the same cell no matter which of the given cells it started from, and `false` otherwise.

Constraints

- $3 \leq N \leq 2 \cdot 10^5$
- $1 \leq Q \leq 5 \cdot 10^5$
- $p_0, p_1, \ldots, p_{N-1}$ is a permutation of the numbers $0, 1, \ldots, N-1$
- $2 \leq K_i \leq N$
- $0 \leq x_0 < x_1 < \ldots < x_{K_i-1} < N$ for every query
- The sum of K_i over all Q queries does not exceed 10^6.

Examples

Sample grader input 1	Sample grader output 1
3 3 0 2 1 2 0 2 3 0 1 2 2 1 2	111

Sample grader input 2	Sample grader output 2
7 3 0 4 2 1 3 5 6 3 0 1 3 3 1 3 4 2 3 6	001

Explanation of example 1

For this example, $p = [0, 2, 1]$. For the second query, the robot may start at cell 0, cell 1 or cell 2. Consider the program `[MAX]`.

- When the robot starts at cell 0: the cells close to 0 are cells with indices 0 and 1. When the robot executes the command `MAX`, since $p_0 < p_1$, it moves to cell 1.
- When the robot starts at cell 1: the cells close to 1 are cells with indices 0, 1 and 2. When the robot executes the command `MAX`, since $p_0 < p_1$ and $p_1 > p_2$, it stays at cell 1.
- When the robot starts at cell 2: the cells close to 2 are cells with indices 1 and 2. When the robot executes the command `MAX`, since $p_1 > p_2$, it moves to cell 1.

Therefore there exists a program after which the robot ends up at cell 1 from all three starting cells, and the answer for this query is `true`. Other programs also work, such as `[MIN, MAX]`, `[MIN, MIN, MAX, MIN, MAX, MIN]` and `[MAX, MIN, MAX]`.

Explanation of example 2

For this example, $p = [0, 4, 2, 1, 3, 5, 6]$.

For the first two queries, it can be shown that there is no program for which the robot ends up at the same cell from every given starting cell, so the answer to both is `false`.

Consider the last query, for which the robot may start at cell 3 or cell 6, and consider the program `[MAX, MAX, MAX]`.

- When the robot starts at cell 3: the cells close to 3 are cells with indices 2, 3, and 4. Since $p_3 < p_4$ and $p_2 < p_4$, the robot moves to cell 4. Following the next two commands in the same way, it ends up on cell 6.
- When the robot starts at cell 6, it stays at cell 6 throughout and ends up there.

Therefore the answer for this query is `true`. The program `[MIN, MAX, MAX, MAX]` also works. However, other programs may not work. For example, if the program is `[MAX, MAX]`, the robot ends up at cell 5 when it starts at cell 3, but it ends up at cell 6 when it starts at cell 6.

Subtasks

Subtask	Points	N	Q	K_i	Additional constraints
0	0	–	–	–	The examples.
1	3	$\leq 2 \cdot 10^5$	$\leq 5 \cdot 10^5$	$= 2$	If the answer for a query is positive, then there exists a valid program using exactly 1 command.
2	7				If the answer for a query is positive, then there exists a valid program using at most 5 commands.
3	9	≤ 100	≤ 500	$= 2$	–
4	17	$\leq 5\,000$	$\leq 5 \cdot 10^5$	$= 2$	–
5	7	$\leq 2 \cdot 10^5$			$x_1 = x_0 + 1$ for each query.
6	8	$\leq 5\,000$	$\leq 5 \cdot 10^5$	$\leq N$	–
7	13	$\leq 2 \cdot 10^5$	$\leq 5 \cdot 10^5$	$\leq N$	The permutation is at first increasing, then decreasing, then increasing again. Formally, there exist two integers a, b such that $0 < a < b < N - 1$ and $p_0 < p_1 < \ldots < p_a > p_{a+1} > \ldots > p_b < p_{b+1} < \ldots < p_{N-1}$.
8	13				$p_0 < p_1 > p_2 < p_3 > \ldots < p_{N-1}$ if N is even, or $p_0 < p_1 > p_2 < p_3 > \ldots > p_{N-1}$ if N is odd.
9	23				–

Sample grader

The input format is the following:

- line 1: two integers – the values of N and Q;
- line 2: N integers $p_0, p_1, \ldots, p_{N-1}$, where p_i is the height of the i-th cell;
- line $3 + i$: the cells given for the i-th query – an integer K_i, followed on the same line by K_i integers $x_0, x_1, \ldots, x_{K_i-1}$, the possible starting cells of the robot.

The output format is the following:

- line 1: a binary string of length Q, whose i-th character is `1` if the answer for the i-th query is `true`, and `0` otherwise.