Автомат

Време: 2 секунди
Памет: 1024 MiB

Има поле, състоящо се от N клетки, подредени в редица и номерирани от 0 до $N-1$ отляво надясно. Всяка клетка има различна височина: височината на клетка i е p_i, а редицата $p_0, p_1, \ldots, p_{N-1}$ е пермутация на числата $0, 1, \ldots, N-1$.

Две клетки с индекси i и j се наричат *близки* тогава и само тогава, когато $|i-j| \leq 1$. В частност, всяка клетка е близка до самата себе си.

Върху полето се намира робот, който първоначално е поставен в някоя клетка. Роботът приема команди от следните два вида:

- `MAX`: измежду всички клетки, близки до текущата клетка на робота, следващата му позиция ще бъде единствената клетка с **максимална** височина;
- `MIN`: измежду всички клетки, близки до текущата клетка на робота, следващата му позиция ще бъде единствената клетка с **минимална** височина.

Тъй като една клетка е близка до самата себе си, дадена команда може да остави робота в същата клетка.

Програма е крайна последователност от команди, като всяка команда е `MAX` или `MIN`. Ако роботът започне от клетка X и изпълни програмата S, той завършва в еднозначно определена клетка, означена с $\text{result}(S, X)$.

Ще ви бъдат зададени Q заявки, като всяка от тях задава множество от начални клетки с размер K_i ($0 \leq i < Q$). По-точно, i-тата заявка съдържа клетките $x_0, x_1, \ldots, x_{K_i-1}$. Роботът ще бъде поставен в една от тях, но предварително не е известно в коя. За всяка заявка трябва да определите дали съществува програма, която премества робота в една и съща крайна клетка, независимо коя от зададените начални позиции от x ще бъде използвана.

Формално трябва да определите дали съществува програма S, за която:

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

Пермутацията p е **една и съща** за всички заявки.

Обърнете внимание, че не е необходимо да конструирате такава програма — трябва само да съобщите дали тя съществува.

Детайли по имплементацията

Първата функция, която трябва да имплементирате, е:

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

- p: пермутация на числата от 0 до $N-1$.

Втората функция, която трябва да имплементирате, е:

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

- x: вектор, задаващ клетките от дадена заявка в строго нарастващ ред.

Функцията `initialize` се извиква точно веднъж, преди което и да е извикване на функцията `exists_program`.

Функцията `exists_program` се извиква Q пъти — по веднъж за всяка заявка. Тя трябва да върне `true`, ако съществува програма, след изпълнението на която роботът завършва в една и съща клетка, независимо от коя от зададените клетки е започнал, и `false` в противен случай.

Ограничения

- $3 \leq N \leq 2 \cdot 10^5$
- $1 \leq Q \leq 5 \cdot 10^5$
- $p_0, p_1, \ldots, p_{N-1}$ е пермутация на числата $0, 1, \ldots, N-1$
- $2 \leq K_i \leq N$
- $0 \leq x_0 < x_1 < \ldots < x_{K_i-1} < N$ за всяка заявка
- Сумата на K_i по всички Q заявки не надвишава 10^6.

Примери

Вход 1	Изход 1
3 3 0 2 1 2 0 2 3 0 1 2 2 1 2	111

Вход 2	Изход 2
7 3 0 4 2 1 3 5 6 3 0 1 3 3 1 3 4 2 3 6	001

Обяснение на пример 1

В този пример $p = [0, 2, 1]$. За втората заявка роботът може да започне от клетка 0, клетка 1 или клетка 2. Нека разгледаме програмата `[MAX]`.

- Когато роботът започне от клетка 0: клетките, близки до 0, са клетките с индекси 0 и 1. Когато роботът изпълни командата `MAX`, тъй като $p_0 < p_1$, той се премества в клетка 1.
- Когато роботът започне от клетка 1: клетките, близки до 1, са клетките с индекси 0, 1 и 2. Когато роботът изпълни командата `MAX`, тъй като $p_0 < p_1$ и $p_1 > p_2$, той остава в клетка 1.
- Когато роботът започне от клетка 2: клетките, близки до 2, са клетките с индекси 1 и 2. Когато роботът изпълни командата `MAX`, тъй като $p_1 > p_2$, той се премества в клетка 1.

Следователно съществува програма, след изпълнението на която роботът завършва в клетка 1 и от трите начални клетки, затова отговорът за тази заявка е `true`.

Работят и други програми, например `[MIN, MAX]`, `[MIN, MIN, MAX, MIN, MAX, MIN]` и `[MAX, MIN, MAX]`.

Обяснение на пример 2

За първите две заявки може да се покаже, че не съществува програма, при която роботът завършва в една и съща клетка от всяка от зададените начални клетки, следователно отговорът и за двете е `false`.

Нека разгледаме последната заявка, при която роботът може да започне от клетка 3 или клетка 6, както и програмата `[MAX, MAX, MAX]`.

- Когато роботът започне от клетка 3: клетките, близки до 3, са клетките с индекси 2, 3 и 4. Тъй като $p_3 < p_4$ и $p_2 < p_4$, роботът се премества в клетка 4. След като изпълни следващите две команди по същия начин, той завършва в клетка 6.
- Когато роботът започне от клетка 6, той остава в клетка 6 през цялото време и завършва там.

Следователно отговорът за тази заявка е `true`.

Програмата `[MIN, MAX, MAX, MAX]` също работи.

Други програми обаче може да не работят. Например, ако програмата е `[MAX, MAX]`, роботът завършва в клетка 5, когато започне от клетка 3, но завършва в клетка 6, когато започне от клетка 6.

Подзадачи

Подзадача	Точки	N	Q	K_i	Допълнителни ограничения
0	0	–	–	–	Примерите.
1	3	$\leq 2 \cdot 10^5$	$\leq 5 \cdot 10^5$	$= 2$	Ако отговорът за дадена заявка е положителен, тогава съществува валидна програма, състояща се само от 1 команда.
2	7				Ако отговорът за дадена заявка е положителен, тогава съществува валидна програма, използваща най-много 5 команди.
3	9	≤ 100	≤ 500		–
4	17	$\leq 5\,000$	$\leq 5 \cdot 10^5$		–
5	7	$\leq 2 \cdot 10^5$			$x_1 = x_0 + 1$ за всяка заявка.
6	8	$\leq 5\,000$	$\leq 5 \cdot 10^5$	$\leq N$	–
7	13	$\leq 2 \cdot 10^5$	$\leq 5 \cdot 10^5$		Пермутацията първоначално е нарастваща, след това намаляваща и накрая отново нарастваща. Формално съществуват две цели числа a и b, такива че $0 < a < b < N - 1$ и $p_0 < p_1 < \ldots < p_a > p_{a+1} > \ldots > p_b < p_{b+1} < \ldots < p_{N-1}$.
8	13				$p_0 < p_1 > p_2 < p_3 > \ldots < p_{N-1}$ ако N е четно или $p_0 < p_1 > p_2 < p_3 > \ldots > p_{N-1}$ ако N е нечетно.
9	23				–

Примерен грейдър

Входният формат е следният:

- ред 1: две цели числа — стойностите на N и Q;
- ред 2: N цели числа $p_0, p_1, \ldots, p_{N-1}$, където p_i е височината на i-тата клетка;

- ред $3 + i$: клетките, зададени в i-тата заявка — цяло число K_i, последвано на същия ред от K_i цели числа $x_0, x_1, \ldots, x_{K_i-1}$, представляващи възможните начални клетки на робота.

Изходният формат е следният:

- ред 1: двоичен низ с дължина Q, чийто i-ти символ е `1`, ако отговорът за i-тата заявка е `true`, и `0` в противен случай.