Препятствия за лама

Една лама иска да се придвижва през Алтиплано (платовидната част на Андите). Тя има карта на платото във формата на таблица от $N \times M$ квадратни клетки. Редовете на картата са номерирани с числата от 0 до $N-1$ от горе надолу, а колоните са номерирани с числата от 0 до $M-1$ от ляво надясно. Клетката на ред i и колона j ($0 \leq i < N, 0 \leq j < M$) означаваме с (i, j).

Ламата е изучила климата на платото и е открила, че всички клетки в даден ред на картата имат една и съща **температура**, а всички клетки в дадена колона на картата имат една и съща **влажност**. Тя Ви дава два целочислени масива T и H с големини N и M, съответно. Тук $T[i]$ ($0 \leq i < N$) задава температурата на клетките на ред i, а $H[j]$ ($0 \leq j < M$) задава влажността на клетките на колона j.

Ламата също е изучила флората на платото и е забелязала, че клетка (i, j) е **без растителност** тогава и само тогава, когато температурата е по-висока от влажността, формално $T[i] > H[j]$.

Ламата може да се предвижва през платото само по **валидни пътища**. Валиден път е поредица от различни клетки, които удовлетворяват следните условия:

- Всяка двойка от последователни клетки в пътя имат обща страна в таблицата.
- Всички клетки на пътя са без растителност.

Вашата задача е да отговорите на Q въпроса. Всеки въпрос се характеризира с четири цели числа: L, R, S и D. Вие трябва да определите дали съществува валиден път, за който:

- Пътят започва от клетка $(0, S)$ и завършва в клетка $(0, D)$.
- Всички клетки на пътя са между колони L и R, включително.

Гарантирано е, че и $(0, S)$, и $(0, D)$ са без растителност.

Детайли по имплементацията

Първата функция, която трябва да имплементирате, е:

```
void initialize(std::vector<int> T, std::vector<int> H)
```

- T: масив с големина N, задаващ температурата на всеки ред.

- H: масив с големина M, задаващ влажността на всяка колона.
- Тази функция се извиква точно веднъж за даден тест, преди извикванията на функцията `can_reach`.

Втората функция, която трябва да имплементирате, е:

```
bool can_reach(int L, int R, int S, int D)
```

- L, R, S, D: цели числа, описващи въпрос.
- Тази функция се извиква Q пъти за даден тест.

Тази функция трябва да върне `true` тогава и само тогава, когато съществува валиден път от клетка $(0, S)$ до клетка $(0, D)$, така че всички клетки в него са между колони L и R, включително.

Ограничения

- $1 \leq N, M, Q \leq 200\,000$
- $0 \leq T[i] \leq 10^9$ за всяко i, спазващо $0 \leq i < N$.
- $0 \leq H[j] \leq 10^9$ за всяко j, спазващо $0 \leq j < M$.
- $0 \leq L \leq R < M$
- $L \leq S \leq R$
- $L \leq D \leq R$
- Клетките $(0, S)$ и $(0, D)$ са без растителност.

Подзадачи

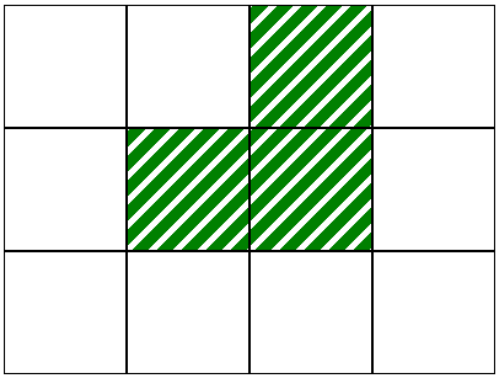
Подзадача	Точки	Допълнителни ограничения
1	10	$L = 0$, $R = M - 1$ за всеки въпрос. $N = 1$.
2	14	$L = 0$, $R = M - 1$ за всеки въпрос. $T[i-1] \leq T[i]$ за всяко i, спазващо $1 \leq i < N$.
3	13	$L = 0$, $R = M - 1$ за всеки въпрос. $N = 3$ и $T = [2, 1, 3]$.
4	21	$L = 0$, $R = M - 1$ за всеки въпрос. $Q \leq 10$.
5	25	$L = 0$, $R = M - 1$ за всеки въпрос.
6	17	Няма допълнителни ограничения.

Пример

Нека разгледаме следното извикване:

```
initialize([2, 1, 3], [0, 1, 2, 0])
```

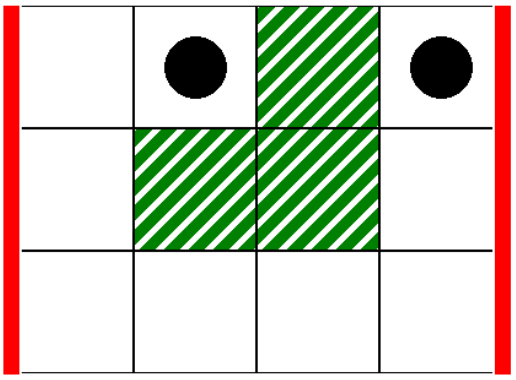
То съответства на картата в следващото изображение, като белите клетки са без растителност:



Нека за първия въпрос имаме следното извикване:

```
can_reach(0, 3, 1, 3)
```

То съответства на сценария от следващото изображение, като плътните вертикални линии задават обхвата на колоните от $L = 0$ до $R = 3$, а черните кръгчета задават началната и крайната клетка:



В този случай, ламата може да достигне от клетка $(0, 1)$ до клетка $(0, 3)$ по следния валиден път:

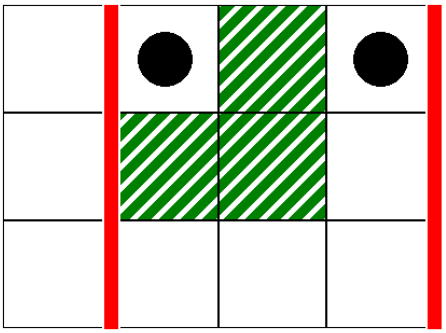
$$(0, 1), (0, 0), (1, 0), (2, 0), (2, 1), (2, 2), (2, 3), (1, 3), (0, 3)$$

Затова извикването трябва да върне `true`.

Нека за втория въпрос имаме следното извикване:

```
can_reach(1, 3, 1, 3)
```

Това съответства на сценария от следното изображение:



В този случай, няма валиден път от клетка $(0, 1)$ до клетка $(0, 3)$, така че всички клетки на пътя да са между колони 1 и 3, включително. Затова извикването трябва да върне `false`.

Локален грейдър

Формат на входа:

```
N M
T[0] T[1] ... T[N-1]
H[0] H[1] ... H[M-1]
Q
L[0] R[0] S[0] D[0]
L[1] R[1] S[1] D[1]
...
L[Q-1] R[Q-1] S[Q-1] D[Q-1]
```

Тук $L[k], R[k], S[k]$ и $D[k]$ $(0 \leq k < Q)$ задават параметрите за всяко извикване на функцията `can_reach`.

Формат на изхода:

```
A[0]
A[1]
...
A[Q-1]
```

Тук $A[k]$ $(0 \leq k < Q)$ е 1, ако извикването `can_reach(L[k], R[k], S[k], D[k])` връща `true` и 0 в противен случай.