장애물

라마가 안데스 고원을 여행하려고 한다. 라마는 이 고원의 지도를 가지고 있는데 그 지도는 $N \times M$의 격자 형태로 되어 있다. 지도의 행은 위에서 아래로 0부터 $N-1$까지 번호가 붙어 있으며 열들은 왼쪽에서 오른쪽으로 0부터 $M-1$까지 번호가 붙어있다. 이 지도에서 i행과 j열이 교차하는 칸은 (i, j)로 표시한다 $(0 \leq i < N, 0 \leq j < M)$.

라마는 이 지도에서 각 행의 모든 칸의 **온도**가 동일하고 각 열의 모든 칸의 **습도**가 동일한 것을 발견했다. 라마는 당신에게 길이가 N인 정수 배열 T와 길이가 M인 정수 배열 H를 주었다. 이 배열에서 $T[i]$ $(0 \leq i < N)$는 i행의 온도이고 $H[j]$ $(0 \leq j < M)$는 j열의 습도이다.

라마는 (i, j) 칸의 온도가 습도보다 큰 경우에 그리고 이 경우에만 (즉, $T[i] > H[j]$) (i, j) 칸이 **건조**하다는 사실도 발견했다.

라마는 이 지역을 **유효 경로**들을 통해서만 이동할 수 있다. 유효 경로는 다음의 조건을 만족하는 다른 칸들의 연속으로 구성된다.

- 연속된 칸들은 변을 공유해야 한다.
- 경로의 모든 칸들이 건조해야 한다.

응시자는 Q개의 질문에 답해야 한다. 각 질문에는 4개의 정수 L, R, S, D가 주어진다. 응시자는 다음을 만족하는 유효 경로가 있는지 결정해야 한다:

- 경로는 $(0, S)$에서 출발해서 $(0, D)$에 도착해야 한다.
- 경로의 모든 칸은 L열과 R열 사이에 있다. (L열과 R열도 포함된다.)

$(0, S)$과 $(0, D)$는 건조한 칸임이 보장된다.

Implementation Details

처음으로 구현해야 하는 함수는 다음과 같다:

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

- T: 각 행의 온도를 나타내는 길이가 N인 배열
- H: 각 열의 습도를 나타내는 길이가 M인 배열
- 이 함수는 각 테스트 케이스에 대해서 오직 한 번만 호출되며 모든 `can_reach` 함수 호출보다 이전에 호출된다.

두번째로 구현해야 하는 함수는 다음과 같다:

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

- L, R, S, D: 각 질문에 주어진 정수들
- 이 함수는 각 테스트 케이스에 대해서 Q번 호출된다.

이 함수는 $(0, S)$에서 $(0, D)$까지 유효 경로가 있을 때만 `true`를 반환해야 한다. 그 유효 경로의 모든 칸은 L열과 R열 사이에 있어야 한다. (L열과 R열도 포함된다.)

Constraints

- $1 \le N, M, Q \le 200\,000$
- $0 \le T[i] \le 10^9$ $(0 \le i < N)$.
- $0 \le H[j] \le 10^9$ $(0 \le j < M)$.
- $0 \le L \le R < M$
- $L \le S \le R$
- $L \le D \le R$
- $(0, S)$과 $(0, D)$은 건조한 칸이다.

Subtasks

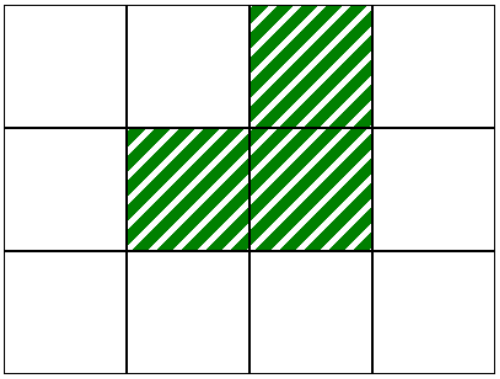
Subtask	Score	Additional Constraints
1	10	$L = 0$, $R = M - 1$ $N = 1$.
2	14	$L = 0$, $R = M - 1$ $T[i - 1] \le T[i]$ $(1 \le i < N)$.
3	13	$L = 0$, $R = M - 1$ $N = 3$, $T = [2, 1, 3]$.
4	21	$L = 0$, $R = M - 1$ $Q \le 10$.
5	25	$L = 0$, $R = M - 1$
6	17	추가제약조건 없음

Example

다음은 함수 호출 예제이다:

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

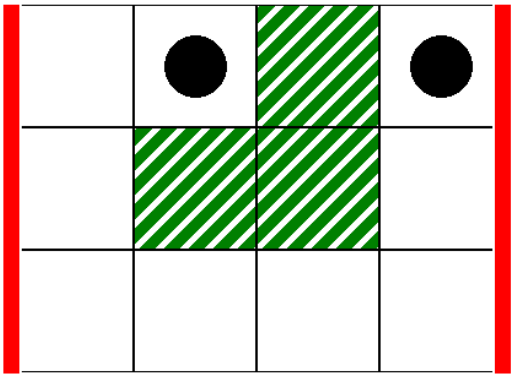
이 함수 호출에 대응되는 지도는 다음과 같다. 흰 칸은 건조한 칸이다:



첫번째 질문으로 다음 호출을 보자:

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

이 호출을 그림으로 표현하면 다음과 같다. 굵은 수직선은 L이 0이고 R이 3일때 포함되는 열들을 나타낸다. 검은 동그라미는 출발 칸과 도착 칸을 나타낸다.



이 경우 라마는 다음 유효 경로를 이용해서 $(0,1)$에서 출발해서 $(0,3)$에 도착할 수 있다:

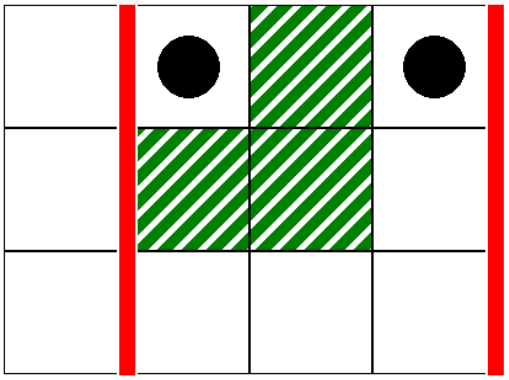
$$(0,1),(0,0),(1,0),(2,0),(2,1),(2,2),(2,3),(1,3),(0,3)$$

따라서 이 호출은 `true`를 반환해야 한다.

두번째 질문으로 다음 호출을 보자:

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

이 호출을 그림으로 표현하면 다음과 같다:



이 경우 1열과 3열 사이만 이용해서 $(0, 1)$에서 출발해서 $(0, 3)$에 도착하는 유효 경로는 없다. 따라서 이 호출은 `false`를 반환해야 한다.

Sample Grader

입력 형식:

```
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]
```

여기서 $L[k], R[k], S[k], D[k]$ $(0 \leq k < Q)$는 각 `can_reach` 호출의 인자이다.

출력 형식:

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

여기서 `can_reach(L[k], R[k], S[k], D[k])` 호출이 `true`를 반환하면 $A[k]$ $(0 \leq k < Q)$ 는 1이고 그렇지 않으면 $A[k]$는 0이다.