Pyramids

Everyone knows that Pharaoh Khufu was a great ruler, but many are unaware that he was also a fashion enthusiast. Back in the day, he had N pyramids numbered from 0 to $N-1$, with pyramid i $(0 \leq i < N)$ consisting of $A[i]$ stones. He also had the latest catalogue of the most fashionable pyramids of the year. The catalogue consists of N pyramids numbered from 0 to $N-1$, with pyramid i $(0 \leq i < N)$ consisting of $B[i]$ stones.

For any x and y, such that $0 \leq x \leq y < N$, we define a **range of pyramids** $A[x..y]$ to be a sequence $A[x], A[x+1], \ldots, A[y]$. We also define a range of pyramids $B[x..y]$ analogously.

Every day, Khufu would browse the catalogue and choose two ranges of pyramids $A[L..R]$ and $B[X..Y]$ where $R - L = Y - X$ (the values of L, R, X and Y may be different every day). After that, he would like to know whether it's possible to **transform** his range $A[L..R]$ to become equal to the catalogue's range $B[X..Y]$. Transforming a range consists of performing the following step an arbitrary number of times: take one stone from a pyramid within the range and move it to an adjacent pyramid within the range.

Your task is to answer multiple questions of the following form. Given four integers L, R, X, and Y, determine whether it is possible to transform $A[L..R]$ into $B[X..Y]$. Note that **the number of stones in each pyramid never actually changes**, Khufu only wonders if one range **could** be transformed into the other one.

Implementation Details

You should implement the following procedures:

```
void init(std::vector<int> A, std::vector<int> B)
```

- A, B: two arrays of length N, describing the number of stones in Khufu's pyramids and in the catalogue respectively.
- This procedure is called exactly once, before any calls to `can_transform`.

```
bool can_transform(int L, int R, int X, int Y)
```

- L, R: starting and ending indices of Khufu's pyramids range.
- X, Y: starting and ending indices of catalogue's pyramids range.
- This procedure should return `true` if it's possible to transform $A[L..R]$ into $B[X..Y]$ and `false` otherwise.

- This procedure is called exactly Q times, once for each day.

Example

Consider the following call:

```
init([1, 2, 3, 4, 5], [2, 2, 2, 4, 5])
```

Assume the grader then calls `can_transform(0, 2, 0, 2)`. This call should return whether sequence of pyramids $A[0..2] = [1, 2, 3]$ can be transformed into $B[0..2] = [2, 2, 2]$. This is indeed possible by moving 1 stone from the last to the first pyramid in the range. Therefore, this call should return `true`.

Assume the grader then calls `can_transform(3, 4, 3, 4)`. This call should return whether we can transform Khufu's pyramids $A[3..4] = [4, 5]$ to $B[3..4] = [4, 5]$ or not. The pyramids already look alike. Therefore, this call should return `true`.

Assume the grader then calls `can_transform(0, 2, 1, 3)`. This call should return whether sequence of pyramids $A[0..2] = [1, 2, 3]$ can be transformed into $B[1..3] = [2, 2, 4]$. This is not possible, and thus this call should return `false`.

Constraints

- $1 \leq N \leq 100\,000$
- $1 \leq Q \leq 100\,000$
- $1 \leq A[i] \leq 10^9$
- $1 \leq B[i] \leq 10^9$

In each call to `can_transform`:

- $0 \leq L \leq R < N$
- $0 \leq X \leq Y < N$
- $R - L = Y - X$

Subtasks

Subtask	Score	Additional Constraints
1	10	$N \leq 5$; $Q \leq 10$; $A[i] \leq 5$, $B[i] \leq 5$ for each i such that $0 \leq i < N$
2	40	$N \leq 1000$; $Q \leq 1000$
3	20	$A[i] \leq 2$ and $B[i] \leq 2$ for each i such that $0 \leq i < N$
4	30	No additional constraints.

Sample grader

Input format:

```
N  Q
A[0]  A[1]  ...  A[N-1]
B[0]  B[1]  ...  B[N-1]
L[0]  R[0]  X[0]  Y[0]
L[1]  R[1]  X[1]  Y[1]
...
L[Q-1]  R[Q-1]  X[Q-1]  Y[Q-1]
```

Here, $L[i]$, $R[i]$, $X[i]$, and $Y[i]$ denote the values of L, R, X and Y in the i-th call to `can_transform`, respectively.

Output format:

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

Here, $P[i]$ is 1 if the i-th call to `can_transform` returns `true` and 0 otherwise.