Problem J
Worldwide Playlist
Time limit: 2 seconds

You are planning a trip around the world. For this trip, you have installed a music app containing n songs, numbered from 1 to n.

Initially, the app generates a playlist for these n songs in the form of a permutation of $1, 2, \ldots, n$, denoted by $a_1, a_2, \ldots, a_n$. It plays the n songs in the order $a_1, a_2, \ldots, a_n$: song a_1 plays first, song a_2 plays second, and so on. The playlist is *infinite*: each time after song a_n plays, it starts all over from song a_1.

The next song starts playing automatically when the current song finishes. Before a song finishes, you may instead press a *skip* button to immediately jump to the next song in the playlist.

You have a desired permutation of the n songs, denoted by $b_1, b_2, \ldots, b_n$. This means that you want to listen to n songs *in full* in the order of $b_1, b_2, \ldots, b_n$ by strategically pressing the skip button zero or more times. In other words, you want the first song you listen to in full (not skipped) to be song b_1, the second to be song b_2, and so on, until the n-th is song b_n. After listening to these n songs in full, you stop listening.

You use this playlist for d days. Between each pair of consecutive days, you update the permutations using three integers c, x, and y as follows:

- If $c = 1$, then you swap a_x and a_y.
- If $c = 2$, then you swap b_x and b_y.

Note that the effect of each update persists to subsequent days.

For each day, assuming you start listening from song a_1, determine the minimum number of times you need to press the skip button so that the songs you listen to in full are $b_1, b_2, \ldots, b_n$ in this order.

Input

The first line of input contains two integers n and d ($2 \le n \le 200\,000$; $2 \le d \le 200\,000$).

The second line contains n integers representing the initial values of $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$; $a_i \ne a_j$ for all $i \ne j$).

The third line contains n integers representing the initial values of $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$; $b_i \ne b_j$ for all $i \ne j$).

The k-th of the next $d - 1$ lines contains three integers c, x, and y ($c \in \{1, 2\}$; $1 \le x < y \le n$), representing the update between the k-th and $(k + 1)$-th days.

Output

Output d lines, where the k-th line should contain the minimum number of skips for the k-th day.

Sample Input #1	Sample Output #1
4 3 1 4 2 3 3 2 1 4 1 3 4 2 1 3	6 2 6

Explanation for the sample input/output #1

On the first day, $(a_1, \ldots, a_4) = (1, 4, 2, 3)$ and $(b_1, \ldots, b_4) = (3, 2, 1, 4)$. You can listen to these songs in full in the desired order with 6 skips:

- Song 1 plays. Skip this song.
- Song 4 plays. Skip this song.
- Song 2 plays. Skip this song.
- Song 3 plays. Listen to this song in full.
- Song 1 plays. Skip this song.
- Song 4 plays. Skip this song.
- Song 2 plays. Listen to this song in full.
- Song 3 plays. Skip this song.
- Song 1 plays. Listen to this song in full.
- Song 4 plays. Listen to this song in full.

On the second day, the permutations are $(a_1, \ldots, a_4) = (1, 4, \mathbf{3}, \mathbf{2})$ and $(b_1, \ldots, b_4) = (3, 2, 1, 4)$. The minimum number of skips is 2.

On the third day, the permutations are $(a_1, \ldots, a_4) = (1, 4, 3, 2)$ and $(b_1, \ldots, b_4) = (\mathbf{1}, 2, \mathbf{3}, 4)$. The minimum number of skips is 6.

Sample Input #2	Sample Output #2
7 5 4 7 1 2 6 5 3 2 6 5 1 4 3 7 1 2 5 2 6 7 1 6 7 2 1 5	16 26 21 20 6