Swap Swap Sort

Time Limit: 3 seconds

Problem Description

There is an array of N integers, each with a value between 1 and K. Your friend has a an algorithm that can sort this array according to any ordering of the numbers from 1 to K. The algorithm performs a sequence of **swap** operations, that exchange two **adjacent** elements of the array. The algorithm performs exactly the minimum number of such swaps to sort the array.

The desired ordering of the numbers from 1 to K is given by a **target permutation**. A target permutation is a sequence of each number from 1 to K appearing exactly once, in the same order that is desired by the corresponding ordering.

For example, the array [1 4 2 1 2] sorted by the target permutation $4, 1, 2, 3$ results in the array [4 1 1 2 2].

You are interested in the number of swaps performed by your friend's algorithm for different target permutations. To explore this, you start with a target permutation of $1, 2, \ldots, K$ and perform Q operations on it. Each operation swaps two **adjacent** elements of the target permutation. After performing each operation, find the number of swaps your friend's algorithm would make if it was run with the current target permutation. The Q operations cumulatively change the target permutation, but do not affect the array.

Input Specification

The first line contains the three integers N, K, and Q ($1 \leq K \leq N \leq 100\,000$, $1 \leq Q \leq 1\,000\,000$).

The next line contains N integers $a_1, a_2, \ldots, a_N$ ($1 \leq a_i \leq K$) specifying the array .

The next Q lines each contains a single integer j ($1 \leq j \leq K-1$), representing the operation of swapping the elements of the target permutation at indices j and $j + 1$.

For 3 of the 25 available marks, $N, Q \leq 5\,000$.

For an additional 3 of the 25 available marks, $Q \leq 100$.

For an additional 5 of the 25 available marks, $K \leq 500$.

Output Specification

For each of the Q operations, output a line containing a single integer; the answer for the current target permutation.

Sample Input
```
5 4 3
1 4 2 1 2
3
2
1
```

Output for Sample Input
```
4
2
2
```

Explanation of Output for Sample Input
The three target permutations are $1, 2, 4, 3$, then $1, 4, 2, 3$, then $4, 1, 2, 3$. For the final target permutation, your friend's algorithm uses two swaps to sort the array `[1 4 2 1 2]` to `[4 1 1 2 2]`.