Fuzzy Ranking

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

In Pigeland, there are n universities, numbered from 1 to n. Each year, several ranking organizations publish rankings of these universities. This year, there are k ranking lists, where each list is a permutation of integers from 1 to n, representing the universities. In each ranking, the closer a university is to the beginning of the permutation, the better its ranking is in this list.

QS	Zhejiang University (#42)	>	Washington University in St. Louis (#118)
ARWU	Washington University in St. Louis (#23)	>	University of Michigan - Ann Arbor (#26)
QS	University of Michigan - Ann Arbor (#25)	>	University of Toronto (#34)
Times	University of Toronto (#18)	>	Cornell University (#22)
ARWU	Cornell University (#12)	>	University of Pennsylvania (#15)
QS	University of Pennsylvania (#13)	>	Princeton University (#16)
ARWU	Princeton University (#6)	>	California Institute of Technology - Caltech (#9)
Times	California Institute of Technology - Caltech (#2)	>	Massachusetts Institute of Technology - MIT (#5)
Therefore, Zhejiang University is better than Massachusetts Institute of Technology - MIT.			

A true story in the 2024 ICPC World Final.

Supigar, a year-4 student who wants to apply for PhD programs in Pigeland, has his own method to evaluate the n universities comprehensively. He considers that university x is *superior* to another university y if and only if:

- x is ranked better than y in at least one list, or
- x is ranked better than z ($z \neq x, z \neq y$) in at least one list, and z is superior to y.

Clearly, under this definition, there might exist some pairs of universities x and y ($x < y$) such that x is superior to y while y is also superior to x. Supigar calls such pairs *fuzzy*.

Supigar has q queries, where the i-th query can be represented by three integers id_i, l_i and r_i ($l_i \leq r_i$). For each query, he will consider the id_i-th rank list and all the universities between the l_i-th position and the r_i-th position (both inclusive) in that list. He wants to know, among these universities, how many pairs of them are fuzzy. Note that the definition of fuzzy pairs requires considering the superior relationships among all k rank lists.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 2 \times 10^5$) indicating the number of test cases. For each test case:

The first line contains three integers n, k, and q ($1 \leq n, k, q \leq 2 \times 10^5$, $1 \leq n \times k \leq 2 \times 10^5$) indicating the number of universities, rank lists, and queries, respectively.

For the following k lines, the i-th line contains n distinct integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ $(1 \leq a_{i,j} \leq n)$ indicating the i-th rank list.

For the following q lines, the i-th line contains three integers id'_i, l'_i, and r'_i $(0 \leq id'_i < k,\ 0 \leq l'_i, r'_i < n)$ indicating the **encoded** index of the rank list and the query range for the i-th query.

- The real value of id_i is equal to $((id'_i + v_{i-1}) \bmod k) + 1$.
- The real value of l_i is equal to $((l'_i + v_{i-1}) \bmod n) + 1$.
- The real value of r_i is equal to $((r'_i + v_{i-1}) \bmod n) + 1$.

Where v_{i-1} is the answer for the $(i-1)$-th query. Specifically, we define $v_0 = 0$. With the encoded queries, you're forced to calculate the answer to each query before processing the next one. It's guaranteed that $1 \leq id_i \leq k$ and $1 \leq l_i \leq r_i \leq n$ after decoding.

It is also guaranteed that neither the sum of $n \times k$ nor the sum of q of all test cases will exceed 2×10^5.

Output

For each test case output q lines. Each line contains a single integer representing the number of fuzzy pairs as the answer to the i-th query.

Example

standard input	standard output
2 5 2 2 1 2 3 4 5 5 4 3 2 1 1 0 2 1 2 1 5 3 3 1 2 3 4 5 1 3 2 4 5 1 2 3 5 4 0 0 2 0 2 3 1 0 3	3 10 1 1 2

Note

For the first sample test case, the two decoded queries are `2 1 3` and `1 1 5`.

For the second sample test case, the three decoded queries are `1 1 3`, `2 4 5`, and `3 2 5`.