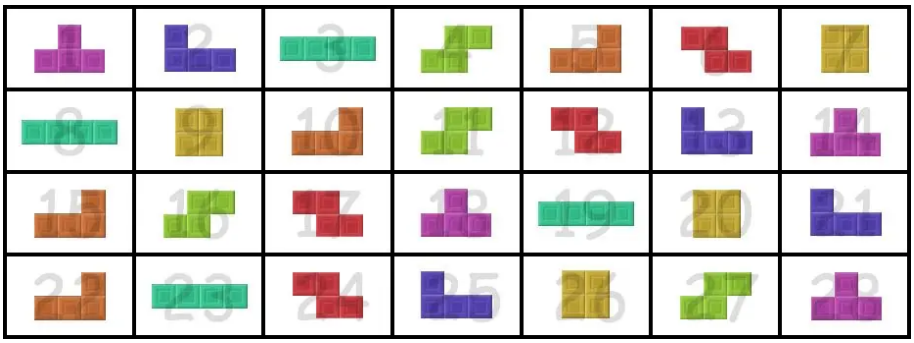
Problem J. Tetris

Time limit: 3 seconds
Memory limit: 512 megabytes

Little C is obsessed with Tetris. In a typical Tetris game, there are 7 types of tetrominoes, each represented by a letter according to its shape: I, T, O, J, L, S, and Z. Players receive a tetromino each time and need to place it in a certain position.

However, the tetrominoes that players receive are not completely random. Instead, every 7 distinct tetrominoes are grouped into a bag, and at the start of the game, players will receive each tetromino in the first bag in sequence, followed by each tetromino in the second bag, and so on. The order of receiving tetrominoes within each bag is completely random, and the order between bags is completely independent. This method of receiving tetrominoes is referred to as the 7-BAG.



In this problem, we are not limited to the 7 different tetrominoes, but rather consider a more general case where there are n different tetrominoes, and the method of receiving tetrominoes is n-BAG (i.e., every n distinct tetrominoes are grouped into a bag, with the order of receiving tetrominoes within each bag being completely independent). Each tetromino is numbered from 1 to n, and we have the following problem:

Little C has received x tetrominoes from the start of the game until now. Starting from the $(x+1)$-th tetromino, Little C begins to remember the types of tetrominoes he received, up to the $(x+m)$-th tetromino. Therefore, we can represent the types of these tetrominoes as $f_1, f_2, \cdots, f_m$.

Since Little C is not a memory master, he may forget the type numbers of some (or even all) of the m tetrominoes he remembers, and he uses 0 to represent the forgotten type numbers.

Unfortunately, Little C has also forgotten the exact value of x, but this value is very important for his future game planning. Therefore, Little C hopes you can infer which non-negative integers x could possibly be based on the information he remembers.

Since there could be infinitely many possible values for x, Little C is only interested in the possible values of x mod n. To reduce the output, you only need to provide the **count** of all values of x mod n that satisfy the known conditions, as well as their **binary XOR sum**.

Since Little C is not a memory master, he cannot guarantee that the types of the tetrominoes he has not forgotten are all correct. In this case, contradictions may arise, leading to the absence of any x that satisfies the known conditions. In this case, simply output two 0s to indicate the answer.

Little C will also correct his remembered information q times. Specifically, he will modify the type number of the $(x+a)$-th tetromino he remembers, changing it from f_a to b (in particular, if $b=0$, it means Little

C has temporarily forgotten the type of the $(x+a)$-th tetromino). After each modification, you need to output the **count** of all values of x mod n that satisfy the known conditions, as well as the **binary XOR sum**.

Note: Each modification is not independent; its effects will persist until the end.

Input

The first line contains three integers n, m, q ($1 \le n \le 10^{18}, 1 \le m \le 5 \times 10^5, 0 \le q \le 5 \times 10^5$).

The second line contains m integers, where the i-th integer f_i ($0 \le f_i \le n$) represents the type number of the $(x+i)$-th tetromino that Little C remembers. In particular, if $f_i = 0$, it means Little C has forgotten the type of the $(x+i)$-th tetromino.

The next q lines each contain two integers a, b ($1 \le a \le m, 0 \le b \le n$), representing the modification of the type number of the $(x+a)$-th tetromino that Little C remembers to b. In particular, if $b = 0$, it means Little C has temporarily forgotten the type of the $(x+a)$-th tetromino.

Output

Output a total of $q+1$ lines. The i-th line should contain two integers representing the **count** of all values of x mod n that satisfy the known conditions after the first $i-1$ modifications, as well as the **binary XOR sum**.

Examples

standard input	standard output
3 17 0 1 0 0 3 0 1 0 1 2 0 2 0 1 1 0 0 1	1 2
4 10 4 0 4 0 0 1 0 0 3 1 0 5 0 5 3 10 4 9 3	4 0 4 0 3 0 3 0 0 0
999999999999999999 10 10 0 0 0 0 314 15 0 0 9 0 1 0 9 287665588162745942 10 0 8 953846340212508779 10 275685051906270282 10 0 5 288735940850628909 9 14189307401575225 2 0 1 0	999999999999999999 999999999999999999

Note

[Sample #1 Explanation]

This memory sequence originally came from the Tetris sequence: [3,2,1],[3,2,**1**],[2,1,**3**],[2,**1**,3],[**1**,**2**,3],[**2**,3,**1**],[**1**,2,3],[**1**,2,3],[1,2,3],[2,1,3]. (The bolded numbers are the information Little C remembers.)

In this case, $x = 5$. Of course, there are also valid cases for $x = 2, 8, 11, \ldots$, but when x mod $3 \neq 2$, there are no valid cases. Therefore, the unique set of possible values for x mod 3 is $\{2\}$, so the count is 1, and

the binary XOR sum is 2.

[Sample #2 Explanation]

Initially, the unique set of possible values for $x \bmod n$ is $\{0, 1, 2, 3\}$.

The first modification changes f_5 to 0, resulting in the memory sequence $[0, 4, 0, 0, 0, 0, 0, 3, 1, 0]$, and the possible values for $x \bmod n$ remain $\{0, 1, 2, 3\}$.

After the second modification, the unique set of possible values for $x \bmod n$ is $\{1, 2, 3\}$.

After the third modification, the unique set of possible values for $x \bmod n$ remains $\{1, 2, 3\}$.

[Sample #3 Hint]

Note: n can be very large