Balloon Robot

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

The 2017 China Collegiate Programming Contest Qinhuangdao Site is coming! There will be n teams participating in the contest, and the contest will be held on a huge round table with m seats numbered from 1 to m in clockwise order around it. The i-th team will be seated on the s_i-th seat.

BaoBao, an enthusiast for competitive programming, has made p predictions of the contest result before the contest. Each prediction is in the form of (a_i, b_i), which means the a_i-th team solves a problem during the b_i-th time unit.

As we know, when a team solves a problem, a balloon will be rewarded to that team. The participants will be unhappy if the balloons take almost centuries to come. If a team solves a problem during the t_a-th time unit, and the balloon is sent to them during the t_b-th time unit, then the unhappiness of the team will increase by $(t_b - t_a)$. In order to give out balloons timely, the organizers of the contest have bought a balloon robot.

At the beginning of the contest (that is to say, at the beginning of the 1-st time unit), the robot will be put on the k-th seat and begin to move around the table. If the robot moves past a team which has won themselves some balloons after the robot's last visit, it will give all the balloons they deserve to the team. During each unit of time, the following events will happen ***in order***:

1. The robot moves to the next seat. That is to say, if the robot is currently on the i-th $(1 \le i < m)$ seat, it will move to the $(i + 1)$-th seat; If the robot is currently on the m-th seat, it will move to the 1-st seat.
2. The participants solve some problems according to BaoBao's prediction.
3. The robot gives out balloons to the team seated on its current position if needed.

BaoBao is interested in minimizing the total unhappiness of all the teams. Your task is to select the starting position k of the robot and calculate the minimum total unhappiness of all the teams according to BaoBao's predictions.

Input

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains three integers n, m and p ($1 \le n \le 10^5$, $n \le m \le 10^9$, $1 \le p \le 10^5$), indicating the number of participating teams, the number of seats and the number of predictions.

The second line contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le m$, and $s_i \ne s_j$ for all $i \ne j$), indicating the seat number of each team.

The following p lines each contains two integers a_i and b_i ($1 \le a_i \le n$, $1 \le b_i \le 10^9$), indicating that the a_i-th team solves a problem at time b_i according to BaoBao's predictions.

It is guaranteed that neither the sum of n nor the sum of p over all test cases will exceed 5×10^5.

Output

For each test case output one integer, indicating the minimum total unhappiness of all the teams according to BaoBao's predictions.

Example

standard input	standard output
4 2 3 3 1 2 1 1 2 1 1 4 2 3 5 1 2 1 1 2 1 1 2 1 3 1 4 3 7 5 3 5 7 1 5 2 1 3 3 1 5 2 5 2 100 2 1 51 1 500 2 1000	1 4 5 50

Note

For the first sample test case, if we choose the starting position to be the 1-st seat, the total unhappiness will be $(3 - 1) + (1 - 1) + (6 - 4) = 4$. If we choose the 2-nd seat, the total unhappiness will be $(2 - 1) + (3 - 1) + (5 - 4) = 4$. If we choose the 3-rd seat, the total unhappiness will be $(1 - 1) + (2 - 1) + (4 - 4) = 1$. So the answer is 1.

For the second sample test case, if we choose the starting position to be the 1-st seat, the total unhappiness will be $(3-1)+(1-1)+(3-2)+(3-3)+(6-4) = 5$. If we choose the 2-nd seat, the total unhappiness will be $(2-1)+(3-1)+(2-2)+(5-3)+(5-4) = 6$. If we choose the 3-rd seat, the total unhappiness will be $(1-1)+(2-1)+(4-2)+(4-3)+(4-4) = 4$. So the answer is 4.