L. Berland.Taxi

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Berland.Taxi is a new taxi company with k cars which started operating in the capital of Berland just recently. The capital has n houses on a straight line numbered from 1 (leftmost) to n (rightmost), and the distance between any two neighboring houses is the same.

You have to help the company schedule all the taxi rides which come throughout the day according to the following rules:

- All cars are available for picking up passengers. Initially the j-th car is located next to the house with the number x_j at time 0.
- All cars have the same speed. It takes exactly 1 minute for any car to travel between neighboring houses i and $i + 1$.
- The i-th request for taxi ride comes at the time t_i, asking for a passenger to be picked up at the house a_i and dropped off at the house b_i. All requests for taxi rides are given in the increasing order of t_i. All t_i are distinct.

When a request for taxi ride is received at time t_i, Berland.Taxi operator assigns a car to it as follows:

- Out of cars which are currently available, operator assigns the car which is the *closest* to the pick up spot a_i. Needless to say, if a car is already on a ride with a passenger, it won't be available for any rides until that passenger is dropped off at the corresponding destination.
- If there are several such cars, operator will pick one of them which *has been waiting the most* since it became available.
- If there are several such cars, operator will pick one of them which *has the lowest number*.

After a car gets assigned to the taxi ride request:

- The driver immediately starts driving from current position to the house a_i.
- Once the car reaches house a_i, the passenger is immediately picked up and the driver starts driving to house b_i.
- Once house b_i is reached, the passenger gets dropped off and the car becomes available for new rides staying next to the house b_i.
- It is allowed for multiple cars to be located next to the same house at the same point in time, while waiting for ride requests or just passing by.

If there are no available cars at time t_i when a request for taxi ride comes, then:

- The i-th passenger will have to wait for a car to become available.
- When a car becomes available, operator will immediately assign it to this taxi ride request.
- If multiple cars become available at once while the passenger is waiting, operator will pick a car out of them according to the rules described above.

Operator processes taxi ride requests one by one. So if multiple passengers are waiting for the cars to become available, operator will not move on to processing the $(i + 1)$-th ride request until the car gets assigned to the i-th ride request.

Your task is to write a program that will process the given list of m taxi ride requests. For each request you have to find out which car will get assigned to it, and how long the passenger will have to wait for a car to arrive. Note, if there is already car located at the house a_i, then the corresponding wait time will be 0.

Input

The first line of input contains integers n, k and m ($2 \le n \le 2 \cdot 10^5$, $1 \le k, m \le 2 \cdot 10^5$) — number of houses, number of cars, and number of taxi ride requests. The second line contains integers $x_1, x_2, \dots, x_k$ ($1 \le x_i \le n$) — initial positions of cars. x_i is a house number at which the i-th car is located initially. It's allowed for more than one car to be located next to the same house.

The following m lines contain information about ride requests. Each ride request is represented by integers t_j, a_j and b_j ($1 \le t_j \le 10^{12}$, $1 \le a_j, b_j \le n$, $a_j \ne b_j$), where t_j is time in minutes when a request is made, a_j is a house where passenger needs to be picked up, and b_j is a house where passenger needs to be dropped off. All taxi ride requests are given in the increasing order of t_j. All t_j are distinct.

Output

Print m lines: the j-th line should contain two integer numbers, the answer for the j-th ride request — *car number* assigned by the operator and *passenger wait time*.

Examples

input

```
10 1 2
3
5 2 8
9 10 3
```

output

```
1 1
1 5
```

input

```
5 2 1
1 5
10 3 5
```

output

```
1 2
```

input

```
5 2 2
1 5
10 3 5
20 4 1
```

output

```
1 2
2 1
```

Note

In the first sample test, a request comes in at time 5 and the car needs to get from house 3 to house 2 to pick up the passenger. Therefore wait time will be 1 and the ride will be completed at time $5+1+6=12$. The second request comes in at time 9, so the passenger will have to wait for the car to become available at time 12, and then the car needs another 2 minutes to get from house 8 to house 10. So the total wait time is $3+2=5$.

In the second sample test, cars 1 and 2 are located at the same distance from the first passenger and have the same "wait time since it became available". Car 1 wins a tiebreaker according to the rules because it has the lowest number. It will come to house 3 at time 3, so the wait time will be 2.