Hoax Spreading

Pak Dengklek has just developed a social media site. There are N users using the site, numbered 0 to $N-1$. There are S hours in one day. All of these users have a fixed usage schedule from day to day. User i will be online $T[i]$ times every day:

- from the start of the $A[i][0]$-th hour to the end of the $B[i][0]$-th hour,
- from the start of the $A[i][1]$-th hour to the end of the $B[i][1]$-th hour,
- ...
- and from the start of the $A[i][T[i]-1]$-th hour to the end of the $B[i][T[i]-1]$-th hour.

At any time, all users on the site like to share news to all other online users. Unfortunately, one of the N users has a hoax at the start of the first day and will spread it. Hence, all users who meet the user with the hoax will also have the hoax at the end of the first day. Two users are stated to be met if there is at least an hour where the two users are both online.

This hoax will also be spread on the second day. Therefore, all users who meet the user with the hoax at the end of the first day will also have the hoax at the end of the second day. This continued the following days.

There are Q scenarios, each can be represented as an integer P. For each scenario, the user who has the hoax at the start of the first day is user P. A different scenario might cause a different hoax spreading. Therefore, for each scenario, Pak Dengklek wonders on the number of users with the hoax at the end of the N-th day, where N is the number of users.

Implementation Details

You should implement the following procedures:

```
void init(int N, int S, int[] T, int[][] A, int[][] B)
```

- N: the number of elements in the array T, A, and B.
- S: the number of hours in one day.
- T: an array of length N, where $T[i]$ is the number of times user i will be online every day.
- A: an array of length N, where $A[i]$ is an array of length $T[i]$.
- B: an array of length N, where $B[i]$ is an array of length $T[i]$.
- This procedure is called exactly once, before any calls to `count_users`.

```
int count_users(int P)
```

- P: the index of the user who has the hoax at the start of the first day.
- This procedure should return the number of users with the hoax at the end of the N-th day, where N is the number of users.
- This procedure is called exactly Q times.

Example

Consider the following call:

```
init(5, 10, [2, 1, 1, 1, 1],
     [[2, 7], [1], [1], [9], [5]], [[4, 9], [3], [1], [10], [6]])
```

After initialization has been done, consider the following call:

```
count_users(0)
```

This means user 0 has the hoax at the start of the first day. User 0 meets user 1 (at the second hour to the third hour) and user 3 (at the ninth hour). Therefore, at the end of the first day, the two users will have the hoax. User 1 also meets user 2 (at the first hour). Therefore, at the end of the second day, user 2 will also have the hoax. There will be no hoax spreading on the third, fourth, and fifth day, thus 4 users will have the hoax at the end of the fifth day. Therefore, the procedure `count_users` should return 4.

Consider another possible call:

```
count_users(4)
```

This means user 4 has the hoax at the start of the first day. User 4 does not meet other users, thus other users will not have the hoax. Therefore, the procedure `count_users` should return 1.

Constraints

- $1 \leq N \leq 200\,000$
- $1 \leq S \leq 10^9$
- $1 \leq Q \leq 100\,000$
- $1 \leq T[i] \leq S$ (for all $0 \leq i \leq N-1$)
- The sum of all elements of T does not exceed $200\,000$.
- $1 \leq A[i][j] \leq B[i][j] \leq S$ (for all $0 \leq i \leq N-1$ and $0 \leq j \leq T[i]-1$)
- $B[i][j-1] < A[i][j]$ (for all $0 \leq i \leq N-1$ and $1 \leq j \leq T[i]-1$)
- $0 \leq P \leq N-1$

- The values of P among all scenarios are pairwise distinct.

Subtasks

1. (15 points) $N, S, Q \leq 50$ and the sum of all elements of T does not exceed 50.
2. (17 points) $N, Q \leq 50$ and the sum of all elements of T does not exceed 50.
3. (21 points) $N, Q \leq 300$ and the sum of all elements of T does not exceed 300.
4. (26 points) $N, Q \leq 2000$ and the sum of all elements of T does not exceed 2000.
5. (21 points) No additional constraints.

Sample Grader

The sample grader reads the input in the following format:

- line 1: $N\ S\ Q$
- line $2 + i$ ($0 \leq i \leq N - 1$): $T[i]\ A[i][0]\ B[i][0]\ \ldots\ A[i][T[i] - 1]\ B[i][T[i] - 1]$
- line $2 + N + i$ ($0 \leq i \leq Q - 1$): P for the i-th call to `count_users`

The sample grader prints the result in the following format:

- line $1 + i$ ($0 \leq i \leq Q - 1$): return value of the i-th call to `count_users`