늑대인간

일본 이바라키 현에는 N개의 도시와 M개의 도로가 있다. 도시들은 **인구의 수가 작은 도시부터 큰 도시의 순서로** 0부터 $N-1$ 까지의 번호로 표현된다. 각각의 도로는 한 쌍의 서로 다른 도시를 연결하며, 양방향으로 이동할 수 있다. 어떤 두 도시를 고르더라도 한 도시에서 다른 도시로 한 개 이상의 도로를 따라서 이동할 수 있다.

당신은 Q 번 여행을 하려고 하는데, 각 여행은 0부터 $Q-1$까지 숫자로 표현된다. 여행 i $(0 \leq i \leq Q-1)$는 도시 S_i 에서 출발해서 도시 E_i에 도착하는 것이다.

당신은 늑대인간이다. 늑대인간에는 두 형태가 있는데, **인간 형태**와 **늑대 형태**이다. 매 여행마다, 당신은 인간 형태로 여행을 시작한다. 여행이 끝났을 때 당신은 반드시 늑대 형태여야 한다. 여행 도중에, **변신**(인간 형태에서 늑대 형태로 바뀌는 것)을 정확하게 한번만 해야 한다. 변신은 도시에 있을 때에만 할 수 있다 (S_i 또는 E_i에서도 변신할 수 있다).

늑대인간으로 사는 것은 쉽지 않다. 인간 형태일 때는 사람이 적은 도시를 피해야 하고 늑대 형태일 때는 사람이 많은 도시를 피해야 한다. 각 여행 i마다 두 경계값 L_i와 R_i가 있는데, $(0 \leq L_i \leq R_i \leq N-1)$ 어느 도시들을 방문하면 안되는지를 알려준다. 구체적으로는, 당신은 여행 i에서 인간 형태일 때는 도시 $0, 1, \ldots, L_i - 1$을 방문하면 안되고, 늑대 형태일 때는 도시 $R_i + 1, R_i + 2, \ldots, N-1$을 방문하면 안된다. 이는 당신은 도시 $L_i, L_i + 1, \ldots, R_i$ 중 하나에서만 변신할 수 있다는 뜻이다.

당신이 할 일은 각각의 여행마다 위에서 주어진 조건을 모두 만족하면서 도시 S_i 에서 출발해서 도시 E_i에 도착하는 것이 가능한지 판단하는 것이다. 만약 도착할 수 있다면 이 경로의 길이는 어떤 값이어도 상관없다.

Implementation details

다음 함수를 구현해야 한다.

```
int[] check_validity(int N, int[] X, int[] Y, int[] S, int[] E, int[]
 L, int[] R)
```

- N: 도시의 수
- X, Y: 길이 M인 배열. 각 j $(0 \leq j \leq M-1)$에 대해, 도시 X[j]는 도시 Y[j] 와 하나의 도로로 직접 연결되어 있다.
- S, E, L, R: 길이 Q인 배열로, 각각의 여행에 대한 정보를 표현한다.

M과 Q 값은 배열의 길이이며, 구현 공지사항에 알려준 방법으로 얻을 수 있다는데 주의하시오.

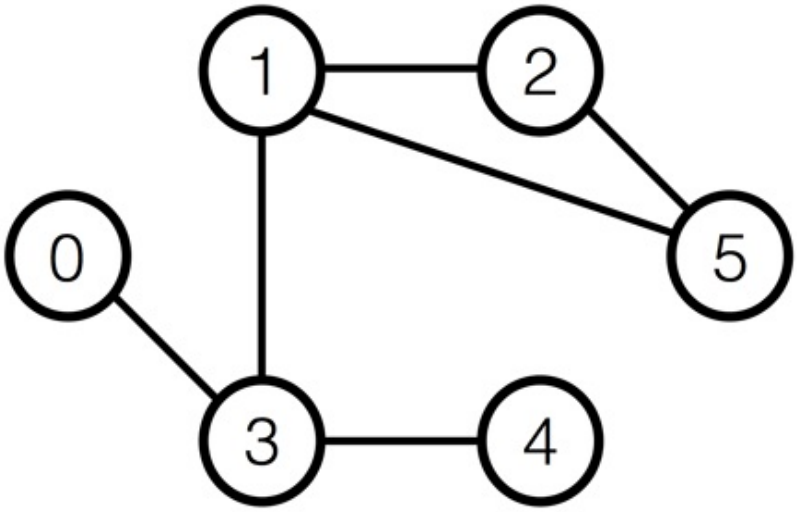
함수 check_validity 는 각 테스트 케이스마다 정확히 한번 호출된다. 이 함수는 길이 Q인 정수 배열

A를 리턴해야 한다. A_i $(0 \leq i \leq Q-1)$의 값은 주어진 조건을 만족하면서 여행 i를 할 수 있다면 1이고, 그렇지 않다면 0이다.

Example

$N = 6$, $M = 6$, $Q = 3$, $X = [5, 1, 1, 3, 3, 5]$, $Y = [1, 2, 3, 4, 0, 2]$, $S = [4, 4, 5]$, $E = [2, 2, 4]$, $L = [1, 2, 3]$, $R = [2, 2, 4]$ 이라고 하자.

그레이더는 `check_validity(6, [5, 1, 1, 3, 3, 5], [1, 2, 3, 4, 0, 2], [4, 4, 5], [2, 2, 4], [1, 2, 3], [2, 2, 4])`를 호출한다.



여행 0의 경우에서, 당신은 도시 4에서 도시 2로 다음과 같이 여행할 수 있다.

- 도시 4에서 출발한다. (당신은 인간 형태이다.)
- 도시 3으로 이동한다. (당신은 인간 형태이다.)
- 도시 1로 이동한다. (당신은 인간 형태이다.)
- 늑대 형태로 변신한다. (당신은 늑대 형태이다.)
- 도시 2로 이동한다. (당신은 늑대 형태이다.)

여행 1과 여행 2의 경우는 조건을 만족하면서 주어진 도시 사이를 이동할 수 없다.

따라서, 당신의 프로그램은 $[1, 0, 0]$을 리턴해야 한다.

압축된 첨부 패키지 파일의 `sample-01-in.txt`와 `sample-01-out.txt`는 이 예제에 대응한다. 다른 입출력 예제도 이 패키지에 포함되어 있다.

Constraints

- $2 \leq N \leq 200\,000$
- $N - 1 \leq M \leq 400\,000$
- $1 \leq Q \leq 200\,000$
- 모든 $0 \leq j \leq M - 1$에 대해서
    - $0 \leq X_j \leq N - 1$

    - $0 \le Y_j \le N-1$
    - $X_j \neq Y_j$
- 어떤 두 도시를 고르더라도 한 도시에서 다른 도시로 한 개 이상의 도로를 따라서 이동할 수 있다.
- 한 쌍의 도시는 최대 하나의 도로로 연결되어 있다. 달리 표현하면, 모든 $0 \le j < k \le M-1$에 대해서, $(X_j, Y_j) \neq (X_k, Y_k)$ 이고 $(Y_j, X_j) \neq (X_k, Y_k)$.
- 모든 $0 \le i \le Q-1$에 대해서
    - $0 \le L_i \le S_i \le N-1$
    - $0 \le E_i \le R_i \le N-1$
    - $S_i \neq E_i$
    - $L_i \le R_i$

Subtasks

1. (7 points) $N \le 100$, $M \le 200$, $Q \le 100$
2. (8 points) $N \le 3\,000$, $M \le 6\,000$, $Q \le 3\,000$
3. (34 points) $M = N-1$ 이고 각각의 도시는 최대 2개의 도로에 인접한다. (도시들은 하나의 직선 형태로 연결되어 있다)
4. (51 points) 추가적인 제약 조건이 없다.

Sample grader

샘플 그레이더는 다음 형식으로 입력을 받는다.

- line 1: N M Q
- line $2+j$ $(0 \le j \le M-1)$: X_j Y_j
- line $2+M+i$ $(0 \le i \le Q-1)$: S_i E_i L_i R_i

샘플 그레이더는 `check_validity`함수의 리턴값을 다음 형식으로 출력한다.

- line $1+i$ $(0 \le i \le Q-1)$: A_i