문제 1. 경작

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	2초
메모리 제한:	256MB

21XX년, IOI 행성의 주민들은 최근 발견된 새로운 행성으로 이주하게 되었다.

새로 발견된 행성은 R행 C열 격자로 이루어진 밭이 있다. 열은 남-북 방향으로, 행은 동-서 방향으로 놓여있다. 북쪽에서 i 번째, 서쪽에서 j 번째 격자 칸의 좌표는 (i, j)이다. 가장 북서쪽에 있는 격자 칸의 좌표는 $(1, 1)$ 이고, 가장 남동쪽에 있는 격자 칸의 좌표는 (R, C)이다. 매년 IOI 행성의 주민들은 밭에 불 바람의 방향을 고른다. 방향은 동, 남, 서 혹은 북이다.

새 행성에서는 농업을 활성화하기 위해서, 밭의 모든 격자 칸에 "JOI 풀"을 심을 것이다. 이주한 첫 연도에는 N 개의 격자 칸에 JOI 풀이 심겨 있다.

JOI 풀은 바람으로 생활권을 늘려간다. 여름마다 JOI 풀의 씨앗이 IOI 행성의 주민들이 고른 바람의 방향으로 날아간다. 씨앗은 원래 JOI 풀이 심겨 있던 곳에서 바람 방향으로 한 칸 움직여 땅에 착지한다. 만약 그 격자 칸에 JOI 풀이 심겨 있지 않다면, 그 격자 칸에는 새로운 JOI 풀이 자란다. 한 격자 칸에 JOI 풀이 심겨 있으면 그 풀은 영구히 자란다.

바람의 방향을 적당히 설정했을 때, 밭의 모든 격자 칸에 JOI 풀을 심을 수 있으려면 최소 몇 년이 걸리는지 구하고 싶다.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 두 정수 R, C가 주어진다. 이는 격자가 R행 C열로 이루어져 있다는 의미이다.
- 둘째 줄에는 정수 N이 주어진다. 이는 이주 첫 연도에 JOI 풀이 심겨 있는 칸의 수가 N이라는 의미이다.
- 다음 N개의 줄의 i 번째 $(1 \leq i \leq N)$ 줄에는 공백으로 구분된 두 정수 S_i, E_i가 주어진다. 이는 이주 첫 연도에 (S_i, E_i) 좌표의 격자 칸에 JOI 풀이 심겨 있다는 의미이다.

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 출력은 우리가 방향을 적당히 설정했을 때, 모든 격자칸에 JOI 풀이 심겨 있기 위해 필요한 연수의 최솟값이다.

제한

- $1 \leq N \leq 300$.
- $1 \leq R \leq 1\ 000\ 000\ 000$.
- $1 \leq C \leq 1\ 000\ 000\ 000$.
- $1 \leq S_i \leq R$ $(1 \leq i \leq N)$.
- $1 \leq E_i \leq C$ $(1 \leq i \leq N)$.
- 이주 첫 연도에 JOI풀이 심겨 있지 않은 격자가 존재한다.

서브태스크 1 (5 점)

- $R \leq 4$

- $C \leq 4$

서브태스크 2 (10 점)

- $R \leq 40$
- $C \leq 40$

서브태스크 3 (15 점)

- $R \leq 40$

서브태스크 4 (30 점)

- $N \leq 25$

서브태스크 5 (20 점)

- $N \leq 100$

서브태스크 6 (20 점)

추가 제한조건이 없다.

예제

standard input	standard output
3 4 3 1 2 1 4 2 3	3

이 예제에서, 이주 첫 연도에 다음 격자 칸에 JOI 풀이 심겨 있다.

	0		0
		0	

새 행성의 밭. '0'이 쓰여 있는 격자 칸은 이주 첫 연도에 JOI 풀이 심겨 있다.

만약에 처음 3년 동안 바람을 서쪽, 남쪽, 남쪽으로 불어가게 만들면, 모든 격자 칸에는 3년 이후에는 JOI 풀이 심겨 있을 것이다. 다음 숫자는 각 격자 칸에 JOI 풀이 심긴 연도를 의미한다. 이는 최솟값이다.

1	0	1	0
2	1	0	2
3	2	2	3

standard input	standard output
4 4 4 1 1 1 4 4 1 4 4	4

문제 2. 항구 시설

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	3.5초
메모리 제한:	1024MB

JOI 항구에는 매일 많은 컨테이너가 운반되어 오고 전국 각지로 트럭을 통해 운반된다.

JOI 항구는 매우 좁아서 컨테이너를 넣을 수 있는 공간이 2개 밖에 없다. 각각의 공간에는 컨테이너를 수직으로 쌓아서 몇 개든 넣을 수 있다.

안전상의 이유로 컨테이너가 항구에서 운반되어 오면 두 공간 중 하나에 컨테이너를 놓아야 한다. 만약 이미 컨테이너가 그 위치에 있으면 이미 있는 컨테이너의 위에 새 컨테이너를 쌓는다. 트럭으로 운반할 때는 두 공간에 쌓인 컨테이너 중 가장 위에서부터 차례대로 운반해야 한다.

오늘 JOI 항구에는 N 개의 컨테이너가 배로 운반될 예정이다. 모든 컨테이너는 오늘 내로 트럭으로 운반될 예정이다.

당신은 JOI 항구의 항구시설 관리를 맡고 있어서, 모든 컨테이너가 배로 운반되는 시각과 트럭으로 운반되는 시간을 알고 있다. 컨테이너를 쌓고 가져가는 경우의 수를 1 000 000 007로 나눈 나머지를 구하여라.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 정수 N이 주어진다. 이는 JOI 항구에 운반될 컨테이너의 수가 N이라는 의미이다.
- 다음 N 개의 줄의 i 번째 $(1 \leq i \leq N)$ 줄에는 공백으로 구분된 두 정수 A_i, B_i가 주어진다. 이는 JOI 항구에 i 번째 컨테이너가 시각 A_i에 와서 시각 B_i에 트럭으로 운반된다는 의미이다.

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 출력은 컨테이너를 쌓고 가져가는 경우의 수를 1 000 000 007로 나눈 나머지이다.

제한

- $1 \leq N \leq 1\ 000\ 000$.
- $1 \leq A_i \leq 2N$ $(1 \leq i \leq N)$.
- $1 \leq B_i \leq 2N$ $(1 \leq i \leq N)$.
- $A_i < B_i$ $(1 \leq i \leq N)$.
- $2N$개의 정수 $A_1, \cdots, A_N, B_1, \cdots, B_N$은 서로 다르다.

서브태스크 1 (10 점)

- $N \leq 20$

서브태스크 2 (12 점)

- $N \leq 2\ 000$

서브태스크 3 (56 점)

- $N \leq 100\ 000$

서브태스크 4 (22 점)

추가 제한조건이 없다.

예제

standard input	standard output
4 1 3 2 5 4 8 6 7	4

컨테이너를 놓는 네 가지 방법이 있다. 각 공간을 A, B라고 하자. 다음 방법으로 컨테이너를 놓을 수 있다.

- 1, 2, 3, 4번 컨테이너를 각각 A, B, A, A에 놓는다.
- 1, 2, 3, 4번 컨테이너를 각각 A, B, A, B에 놓는다.
- 1, 2, 3, 4번 컨테이너를 각각 B, A, B, A에 놓는다.
- 1, 2, 3, 4번 컨테이너를 각각 B, A, B, B에 놓는다.

standard input	standard output
3 1 4 2 5 3 6	0
5 1 4 2 10 6 9 7 8 3 5	8
8 1 15 2 5 3 8 4 6 14 16 7 9 10 13 11 12	16

문제 3. 불꽃놀이 막대

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	2초
메모리 제한:	256MB

JOI군은 자신을 포함하여 N 명의 친구들과 불꽃놀이 막대를 가지고 놀 것이다. 이번에 사용할 불꽃놀이 막대는 불을 붙이면 정확히 T 초 동안 불이 붙는다.

처음에 JOI군과 친구들은 동서 방향으로 일직선으로 서 있고 한 사람당 하나의 불꽃놀이 막대를 들고 있다. JOI군과 친구들은 각각 1 이상 N 이하의 번호가 붙어있다. $i < j$를 만족하는 i와 j에 대해서, i 번째 사람은 j 번째 사람의 서쪽에 서 있거나 같은 장소에 서 있다. i 번째 사람과 가장 서쪽에 있는 첫 번째 사람의 거리는 X_i 미터이다. JOI군은 K 번째 사람이다.

불꽃놀이를 시작하려 할 때 라이터의 연료가 충분하지 않다는 사실을 알았다. 오직 하나의 불꽃놀이 막대에만 불을 붙일 수 있다.

그래서 일단 JOI군의 불꽃놀이 막대에 불을 붙이고 타고 있는 불꽃놀이 막대의 불을 옮겨가면서 불을 붙이기로 했다. 불꽃놀이 막대에서 불을 옮길 때는 다음 조건을 만족해야 한다.

- 불이 붙지 않은 불꽃놀이 막대를 불을 붙인 지 T 초 이내의 불꽃놀이 막대와 맞닿아야 한다. 불을 붙인 지 정확히 T 초가 지나도 불을 옮길 수 있다.
- 불을 붙이려는 불꽃놀이 막대는, 한 번도 불이 붙은 적이 없어야 한다.
- 불이 붙지 않은 불꽃놀이 막대와 불이 붙은 불꽃놀이 막대를 가진 사람이 같은 장소에 있어야 한다.

우리는 한 불꽃놀이 막대에서 다른 불꽃놀이 막대로 불이 붙기를 기다리는 시간 등을 무시할 것이다.

JOI군과 친구들이 서로 떨어져 서 있어, 불을 붙이기 위해서는 잘 이동해야 한다. 그들은 임의의 속도로 달릴 수 있지만 불꽃놀이를 하는 중 달리면 위험하므로 속도가 초당 s 미터를 넘지 않게 하고 싶다. 여기서, s는 음이 아닌 정수이다.

모든 불꽃놀이 막대에 불을 붙이기 위해서 속도 제한을 어떻게 정하는 게 좋을까?

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 세 정수 N, K, T가 주어진다. 이는 N 명의 사람이 있고, JOI군이 K 번째 사람이고, 막대 불꽃놀이에 불을 붙이면 T 초 동안 불이 붙어있다는 의미이다.
- 다음 N 개의 줄의 i 번째 ($1 \leq i \leq N$) 줄에는 공백으로 정수 X_i가 주어진다. 이는 i 번째 사람과 가장 서쪽에 있는 첫 번째 사람의 거리는 X_i 미터라는 의미이다.

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 출력은 모든 불꽃놀이에 불을 붙이기 위한 음이 아닌 정수 속도 제한 s이다.

제한

- $1 \leq K \leq N \leq 100\ 000$.
- $1 \leq T \leq 1\ 000\ 000\ 000$.
- $1 \leq X_i \leq 1\ 000\ 000\ 000$ ($1 \leq i \leq N$).

- $X_1 = 0$
- $X_i \leq X_j$ $(1 \leq i \leq N)$.

서브태스크 1 (30 점)

- $N \leq 20$

서브태스크 2 (20 점)

- $N \leq 1\,000$

서브태스크 3 (50 점)

추가 제한조건이 없다.

예제

standard input	standard output
3 2 50 0 200 300	2

이 예제에서, 속도 제한은 초당 2미터여도 된다.

첫 번째 사람이 동쪽으로, 두 번째와 세 번째 사람이 서쪽으로 움직인다. 속도는 초당 2미터이다. 50초 이후에 두 번째 사람은 첫 번째 사람에게 불을 옮길 수 있다.

그리고 첫 번째 사람이 동쪽으로, 세 번째 사람이 서쪽으로 움직인다. 속도는 초당 2미터이다. 25초 이후에 첫 번째 사람은 세 번째 사람에게 불을 옮길 수 있다.

속도 제한이 1미터였다면 모든 막대에 불을 붙일 수 없다.

standard input	standard output
3 2 10 0 200 300	8

이 예제에서, 속도 제한은 초당 8미터여도 된다.

첫 번째와 두 번째 사람이 동쪽으로, 세 번째 사람이 서쪽으로 움직인다. 속도는 초당 8미터이다.

3초 후에 두 번째 사람이 움직임을 멈춘다. 첫 번째와 세 번째 사람은 계속 움직인다.

6.5초가 더 지난 이후에 두 번째 사람과 세 번째 사람이 같은 장소에 모인다. 두 번째 사람과 세 번째 사람이 움직임을 멈춘다. 첫 번째 사람은 계속 움직인다.

0.5초가 더 지난 이후에 두 번째 사람은 세 번째 사람에게 불을 옮긴다. 첫 번째 사람은 계속 움직인다. 세 번째 사람은 서쪽으로 움직인다. 속도는 초당 8미터이다.

9초가 더 지난 이후에 첫 번째 사람과 세 번째 사람이 같은 장소에 모인다. 세 번째 사람은 첫 번째 사람에게 불을 옮긴다.

속도 제한이 7미터였다면 모든 막대에 불을 붙일 수 없다.

standard input	standard output
20 6 1 0 2 13 27 35 46 63 74 80 88 100 101 109 110 119 138 139 154 172 192	6

문제 4. 티켓 정리

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	4초
메모리 제한:	256MB

JOI 공화국에는 1번부터 N번까지 번호가 붙은 N 개의 기차역이 있다. 기차역은 원형 철도에 시계방향으로 위치해 있다.

철도를 이용하기 위한 티켓은 N 종류가 있고 1번부터 N번까지 번호가 붙어 있다. i번 ($1 \leq i \leq N-1$) 티켓을 하나 사용하면 i번 기차역에서 $i+1$번 기차역으로 혹은 $i+1$번 기차역에서 i번 기차역으로 사람 한 명이 이동할 수 있다. N번 티켓을 하나 사용하면 1번 기차역에서 N번 기차역으로 혹은 N번 기차역에서 1번 기차역으로 사람 한 명이 이동할 수 있다. 이 티켓은 N 종류의 티켓이 정확히 한 장씩 총 N 장이 들어 있는 묶음으로 판매되고 있다.

당신은 JOI 공화국의 여행회사에서 일하고 있다. 당신은 고객들에게 티켓을 나눠주어야 한다.

오늘 티켓을 나눠달라는 M 개의 요청이 있었다. i 번째 요청은 C_i 명의 사람이 A_i번 기차역에서 B_i번 기차역으로 이동하고 싶다는 요청이었다. C_i 명의 사람들이 모두 같은 경로로 이동 할 필요는 없다.

모든 요청을 처리하기 위해서 사야 할 묶음의 최소 개수를 알고 싶다.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 두 정수 N, M이 주어진다. JOI 공화국에 N 개의 기차역이 있으며 M 개의 요청을 오늘 받았다는 의미이다.
- 다음 N 개의 줄의 i 번째 ($1 \leq i \leq N$) 줄에는 공백으로 구분된 세 정수 A_i, B_i, C_i가 주어진다. 이는 i 번째 요청이 C_i명의 사람이 A_i번 기차역에서 B_i번 기차역으로 이동하고 싶다는 요청이라는 의미이다.

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 출력은 모든 요청을 처리하기 위해서 사야 할 묶음의 최소 개수이다.

제한

- $3 \leq N \leq 200\ 000$.
- $1 \leq M \leq 100\ 000$.
- $1 \leq A_i \leq N$ $(1 \leq i \leq M)$.
- $1 \leq B_i \leq N$ $(1 \leq i \leq M)$.
- $1 \leq C_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq M)$.
- $A_i \neq B_i$ $(1 \leq i \leq M)$

서브태스크 1 (10 점)

- $N \leq 20$
- $M \leq 20$
- $C_i = 1$ $(1 \leq i \leq M)$

서브태스크 2 (35 점)

- $N \leq 300$
- $M \leq 300$
- $C_i = 1$ $(1 \leq i \leq M)$

서브태스크 3 (20 점)

- $N \leq 3\ 000$
- $M \leq 3\ 000$
- $C_i = 1$ $(1 \leq i \leq M)$

서브태스크 4 (20 점)

- $C_i = 1$ $(1 \leq i \leq M)$

서브태스크 5 (15 점)

추가 제한조건이 없다.

예제

standard input	standard output
3 3 1 2 1 2 3 1 3 1 1	1

모두가 시계방향으로 이동하면 각 종류의 티켓이 하나씩 필요하다. 즉, 한 묶음만 사도 충분하다.

standard input	standard output
3 2 1 2 4 1 2 2	3

다음 방법으로 이동하면 각 종류의 티켓이 세 장씩 필요하다:

- 첫 번째 요청에서, 세 명이 시계방향으로, 한 명이 반시계방향으로 움직인다.
- 두 번째 요청에서, 두 명이 반시계방향으로 움직인다.

그래서 세 묶음을 사면 충분하다.
두 묶음을 사서 이동하는 것은 불가능하므로 3을 출력한다.

standard input	standard output
6 3 1 4 1 2 5 1 3 6 1	2

예를 들면 두 묶음을 사서 다음과 같이 나누어 주면 된다.

- 1번 역에서 4번 역으로 이동하고 싶은 사람에게 1, 2, 3번 티켓을 준다.
- 2번 역에서 5번 역으로 이동하고 싶은 사람에게 1, 6, 5번 티켓을 준다.
- 3번 역에서 6번 역으로 이동하고 싶은 사람에게 3, 4, 5번 티켓을 준다.

한 묶음을 사서 이동하는 것은 불가능하므로 2를 출력한다.

문제 5. 고장난 기기

시간 제한: 2초
메모리 제한: 256MB

고고학자 Anna와 Bruno는 이란의 유적을 조사하고 있다.

두 명이 역할을 나눠 Anna는 유적을 발견하고 유물을 발견하며 Bruno는 베이스캠프에서 결과를 분석한다.

조사는 총 $Q(= 1\ 000)$일 동안 계획이 되어있다. 매일 Anna는 Bruno에게 통신 기기를 사용하여 결과를 보낸다. 각 통신기기의 결과는 정수 X로 표시된다.

안나는 통신 기기를 하루에 한 번만 사용할 수 있다. 통신기기는 0 혹은 1로 되어있는 길이 $N(= 150)$의 수열을 보낼 수 있다.

하지만 이 기계가 고장 나 버려서 보내는 길이 N의 수열 중 기능이 제대로 작동하지 않는 위치가 생겨버렸다. 기능하지 않는 위치에는 어떤 값을 설정해도 0이 보내지게 된다. Anna가 수열을 보낼 때 고장 난 위치가 어딘지 알 수 있다. 하지만, Bruno는 그 위치를 모른다. 고장 난 위치가 몇 개인지, 어딘지는 매일 바뀐다.

이 조사가 지연되면 문제가 생길 수 있다. Anna와 Bruno는 이란에서 열리는 국제 프로그래밍 대회의 후보인 당신에게 조사 결과를 보내는 프로그램을 작성해달라고 요청했다.

당신은 Anna와 Bruno 사이에서 통신하는 두 개의 프로그램을 작성하여야 한다.

- 수열의 길이 N, 보내야 할 정수 X, 고장 난 위치의 개수 K, 고장 난 위치 P가 주어졌을 때, 첫 번째 프로그램은 Anna가 보낼 수열 S를 정한다.
- Bruno가 수열 A를 받았을 때, 두 번째 프로그램은 정수 X를 복구한다.

통신 장치가 고장 난 위치가 아닌 곳에서는, 수열 S와 수열 A는 같은 값을 가진다. 통신 장치가 고장 난 곳에서는, 수열 A는 수열 S의 값과 관계없이 0이다.

구현 명세

당신은 *같은 프로그래밍 언어로 작성된* 파일 두 개를 작성해야 한다.

첫 번째 파일의 이름은 `Anna.c` 혹은 `Anna.cpp`이다. 이 파일은 Anna가 보낼 수열을 정하는 역할을 하며, 다음 함수를 구현해야 한다. 이 파일은 `Annalib.h`를 include해야 한다.

- `void Anna(int N, long long X, int K, int P[])`

  이 함수는 각 테스트 케이스마다 정확히 $Q = 1\ 000$ 번 불린다.

  - 인자 `N`은 보낼 수열의 길이를 나타낸다.
  - 인자 `X`는 보낼 숫자를 나타낸다.
  - 인자 `K`는 부서진 위치의 개수를 나타낸다.
  - 인자 `P[]`는 부서진 위치를 나타내는 길이 K의 수열이다.

  함수 `Anna`는 다음 함수를 호출해야 한다.

  - `void Set(int pos, int bit)`
    이 함수는, 통신 기기로 보낼 수열 S를 설정한다.
    - 인자 `pos`는 수열의 값을 설정할 위치이다. `pos`는 0 이상 $N - 1$ 이하이다. *위치가 0부터 시작함에 유의하여라.* 만약에 이 범위를 벗어나서 함수를 호출한 경우 **오답** [1]이 된다. 같은 `pos`를 인자로 하여 함수를 두 번 이상 호출한 경우 **오답** [2]이 된다.
    - 인자 `bit`는 `pos` 번째에 설정할 값이다. `bit`의 값은 0 혹은 1이어야 한다. 다른 인자로 함수를 호출한 경우 **오답** [3]이 된다.

  함수 `Set`은 함수 `Anna` 안에서 정확히 N번 호출되어야 한다. `Anna` 함수가 종료되었을 때, `Set`이 호출된 횟수가 N과 다르면 **오답** [4]이 된다.

  만약 `Anna`가 함수를 올바르지 않게 호출할 경우 프로그램이 종료된다.

두 번째 파일의 이름은 `Bruno.c` 혹은 `Bruno.cpp`이다. 이 파일은 탐사 결과를 복구하는 역할을 하며, 다음 함수를 구현해야 한다. 이 파일은 `Brunolib.h`를 include해야 한다.

- `long long Bruno(int N, int A[])`
  이 함수는 각 테스트 케이스마다 정확히 $Q = 1\ 000$ 번 불린다.
  - 인자 `N`은 Bruno가 받은 수열의 길이를 나타낸다.
  - 인자 `A[]`는 Bruno가 받은 길이 N의 수열이다.
  - 함수 Bruno는 X를 찾아서 반환해야 한다.

채점은 다음과 같은 방식으로 진행된다. 만약 프로그램이 오답으로 판단된 경우, 즉시 채점은 종료된다.

1. `cnt=0`으로 설정한다.
2. 함수 `Anna`를 한 번 호출한다.
3. 함수 `Anna`에 호출된 수열을 S라고 하자. S중 P에 포함된 위치를 0으로 바꾸는 작업을 `A`에 한 후, 함수 `Bruno`를 한 번 호출한다.
4. `cnt=cnt+1`로 설정한다. `cnt`<Q이면 2.로 돌아간다. `cnt`=Q이면, 5.로 간다.
5. 채점이 완료된다.

참고 사항

- 실행 시간과 메모리 사용량은 채점 방식의 1, 2, 3, 4에서 계산된다.
- 당신의 프로그램은 채점 방식 2.의 `Anna` 혹은 채점 방식 3.의 `Bruno`에서 오답으로 판단되면 안 된다. 당신의 프로그램은 런타임 에러 없이 실행되어야 한다.
- 당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다. 제출한 프로그램은 그레이더와 함께 컴파일되어 하나의 실행파일이 된다. 모든 전역변수나 내부 함수는 다른 파일과의 충돌을 피하기 위해 `static`으로 선언되어야 한다. Anna와 Bruno는 2개의 별개의 프로세스로 실행되기 때문에 채점될 때 전역변수를 공유하지 않는다.
- 각 프로세스에서, `Anna`와 `Bruno`는 각각 $Q = 1\ 000$번 호출된다. **사용할 변수의 초기화는 적절히 진행되어야 한다.**
- 당신의 프로그램은 표준 입출력을 사용해서는 안 된다. 당신의 프로그램은 어떠한 방법으로도 다른 파일에 접근해서는 안 된다.

당신은 대회 홈페이지의 아카이브에서 프로그램을 테스트하기 위한 목적의 샘플 그레이더를 받을 수 있다. 아카이브는 당신의 프로그램의 예제 소스 또한 첨부되어 있다. 샘플 그레이더는 파일 `grader.c` 혹은 `grader.cpp`이다. 당신의 프로그램이 `Anna.c`와 `Bruno.c` 혹은, `Anna.cpp`와 `Bruno.cpp` 인 경우 다음 커맨드로 컴파일할 수 있다.

- C `g++ -std=c11 -O2 -o grader grader.c Anna.c Bruno.c -lm`
- C++ `g++ -std=c++14 -O2 -o grader grader.cpp Anna.cpp Bruno.cpp`

컴파일이 성공적이면, 파일 `grader`가 생성된다.

실제 그레이더와 샘플 그레이더는 다름에 주의하여라. 샘플 그레이더는 하나의 프로세스에서 실행되며, 입력을 표준 입력으로부터 받고, 출력을 표준 출력에 출력한다.

입력 형식

샘플 그레이더는 다음 형식으로 표준 입력으로 부터 데이터를 입력받는다.

- 첫째 줄에 정수 Q가 주어진다.
- 그리고 Q개의 쿼리의 정보가 주어진다.
- 각 쿼리의 정보는 다음과 같은 두 줄로 되어있다.
  - 첫 번째 줄은 공백으로 구분된 세 정수 N, X, K가 주어진다. 이는 보낼 수열의 길이가 N이고, Anna가 보낼 정수가 X고, 부서진 위치가 K개라는 의미이다.
  - 두 번째 줄은 공백으로 구분된 K개의 정수 $P_0, P_1, \cdots, P_{K-1}$ 이 주어진다. 이는 각 i $(0 \leq i \leq K-1)$에 대해, 수열의 P_i번째 위치가 고장 났다는 의미이다.

출력 형식

프로그램이 정상적으로 종료되었다면, 샘플 그레이더는 다음과 같은 정보를 표준 출력에 출력한다. (따옴표는 출력하지 않는다.)

- 오답으로 판단된 경우, 오답의 종류를 "`Wrong Answer [1]`"과 같은 형식으로 출력하고, 프로그램이 종료된다.
- 만약 모든 Anna의 호출이 오답으로 판단되지 않을 경우, "`Accepted`"와 L^*을 표준 출력으로 출력한다. L^*의 값은 배점 항목을 참고하여라.

프로그램이 다양한 오답의 종류에 속해 있으면 샘플 그레이더는 그중 하나만 출력할 것이다.

제한

- $Q = 1000$.
- $N = 150$.
- $0 \leq X \leq 1\ 000\ 000\ 000\ 000\ 000\ 000$.
- $1 \leq K \leq 40$.
- $0 \leq P_i \leq N-1$ $(0 \leq i \leq K-1)$.
- $P_i < P_{i+1}$ $(0 \leq i \leq K-2)$.

배점

- L^*를 이 문제의 모든 테스트 케이스의 최솟값이라고 하자.
  - $K \leq L$인 모든 쿼리에 대해서 Bruno가 정답을 말한 최대 정수 $L \leq 40$.
- 이 문제의 점수는 다음과 같이 계산된다.
  - $L^* = 0$인 경우, 점수는 0점이다.
  - $1 \leq L^* \leq 14$인 경우, 점수는 8점이다.
  - $15 \leq L^* \leq 37$인 경우, 점수는 $(L^* - 15) \times 2 + 41$점 이다.
  - $38 \leq L^* \leq 40$인 경우, 점수는 $(L^* - 38) \times 5 + 90$점 이다.

예제

예제 입력과 이에 해당하는 함수 호출을 보여준다. 이 예제는 $Q = 2, N = 3$이기 때문에 문제의 제한을 만족하지 않음에 유의하여라.

예제 입력	예제 함수 호출			
	호출	반환값	호출	반환값
2 3 14 1 2 3 9 2 0 1	`Anna(...)`			
			`Set(0,0)`	
				(없음)
			`Set(1,0)`	
				(없음)
			`Set(2,1)`	
				(없음)
		(없음)		
	`Bruno(...)`			
		14		
	`Anna(...)`			
			`Set(0,0)`	
				(없음)
			`Set(1,1)`	
				(없음)
			`Set(2,1)`	
				(없음)
		(없음)		
	`Bruno(...)`			
		9		

여기서 `Anna(...)`, `Bruno(...)`, `Anna(...)`, `Bruno(...)` 호출의 인자들은 다음과 같다.

인자	`Anna(...)`	`Bruno(...)`	`Anna(...)`	`Bruno(...)`
`N`	3	3	3	3
`K`	14		9	
`X`	1		2	
`P`	{2}		{0, 1}	
`A`		{0, 0, 0}		{0, 0, 1}

문제 6. 철도 여행

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	2초
메모리 제한:	512MB

JOI 전철은 철도 하나를 운영하는 회사이다. JOI 전철의 철도에는 1번부터 N번까지의 번호가 붙어있는 N 개의 철도역이 일직선상에 놓여있다. 각 i $(1 \leq i \leq N-1)$에 대해서, i번 역과 $i+1$번 역은 선로로 연결되어있다.

JOI 전철에는 양방향으로 달리는 K 종류의 열차가 있다. 열차의 종류는 1 이상 K 이하의 정수로 표현된다. 각 역에는 **중요도**라고 하는 1 이상 K 이하의 정수가 하나씩 붙어 있다. i번 $(1 \leq i \leq N)$ 역의 중요도는 L_i 이다. 양 끝에 있는, 즉 1번과 N번 역은 중요도가 K 이다.

j번 $(1 \leq j \leq K)$ 종류의 열차는 중요도가 j 이상인 철도역에서만 멈추고 다른 철도역에서는 멈추지 않는다. 양 끝에 있는 1번과 N번 철도역은 중요도가 K이기 때문에 모든 열차는 이 철도역에서는 멈춘다.

많은 승객이 JOI 전철을 이용하고 있다. 여행 중에 승객들은 목적지와 반대 방향으로 움직이거나 목적지를 통과 할 수도 있지만 결국에는 목적지에 멈추어야 한다. 승객들은 역에 멈추는 것을 좋아하지 않는다. 그러므로 도중에 정차하는 철도역의 수를 최소화하고 싶다. 어떤 승객이 열차를 갈아타기 위해 역에 멈춘 경우에도 한 번으로 계산하고, 출발역과 도착역은 세지 않는다.

당신의 업무는 승객의 출발역과 도착역이 주어졌을 때, 이 두 역의 사이를 이동하는 도중에 정차하는 철도역의 수를 구하는 프로그램을 작성해야 한다.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 세 정수 N, K, Q가 주어진다. 이는 JOI 전철에는 N 개의 역이 있고, K 종류의 열차가 있으며, 질문의 개수가 Q 개라는 의미이다.
- 다음 N 개의 줄의 i 번째 $(1 \leq i \leq N)$ 줄에는 정수 L_i가 주어진다. 이는 i번 역의 중요도가 L_i라는 의미이다.
- 다음 Q 개의 줄의 k 번째 $(1 \leq k \leq Q)$ 줄에는 공백으로 구분된 두 정수 A_k, B_k가 주어진다. 이는 k 번째 승객의 출발역이 A_k번 역이고, 도착역이 B_k번 역이라는 의미이다.

출력 형식

표준 출력으로 Q 개의 줄을 출력하여라. k 번째 $(1 \leq k \leq Q)$ 줄에는 A_k번 역에서 B_k번 역으로 이동할 때, 도중에 정차하는 철도역의 최솟값을 출력하여라.

제한

- $2 \leq N \leq 100\ 000$.
- $1 \leq K \leq N$.
- $1 \leq Q \leq 100\ 000$.
- $1 \leq L_i \leq K$ $(1 \leq i \leq N)$.
- $1 \leq A_k \leq N$ $(1 \leq k \leq Q)$.
- $1 \leq B_k \leq N$ $(1 \leq k \leq Q)$.
- $A_k \neq B_k$ $(1 \leq k \leq Q)$.

서브태스크 1 (5 점)

- $N \leq 100$
- $K \leq 100$
- $Q \leq 50$

서브태스크 2 (15 점)

- $Q \leq 50$

서브태스크 3 (25 점)

- $K \leq 20$

서브태스크 4 (55 점)

추가 제한조건이 없다.

예제

standard input	standard output
9 3 3 3 1 1 1 2 2 2 3 3 2 4 4 9 6 7	1 3 0

이 예제에서, 질문은 세 가지가 있다.

- 첫 번째 질문은 2번 역에서 4번 역까지 이동하는 것이다. 이때 2번 역에서 4번 역까지 1번 종류의 열차를 이용하면 도중에 정차하는 역은 3번 역 하나뿐이 된다.
- 두 번째 질문은 4번 역에서 9번 역까지 이동하는 것이다. 이때 우선 4번 역에서 5번 역까지 1번 종류의 열차를 이용하고, 다음에 5번 역에서 1번 역까지 2번 종류의 열차를 이용하고, 마지막으로 1번 역에서 9번 역까지 3번 종류의 열차를 이용하면 도중에 정차하는 역은 5번 역, 1번 역, 8번 역의 셋이 된다.
- 세 번째 질문은 6번 역에서 7번 역까지 이동하는 것이다. 이때 6번 역에서 7번 역까지 2번 종류의 열차를 이용하면 도중에 정차하는 역 없이 이동하는 것이 가능하다.

standard input	standard output
5 2 1 2 1 1 1 2 1 4	1

도중에 목적지가 있는 역을 지나쳐도 되는 점에 주의하여라.

standard input	standard output
15 5 15 5 4 1 2 3 1 1 2 4 5 4 1 5 3 5 8 1 11 1 5 3 6 11 9 12 15 14 15 2 3 12 2 1 4 8 15 5 12 6 1 13 13 8 14 9	2 1 1 3 2 0 3 4 0 1 3 4 1 2 2

문제 7. 장거리 버스

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	2초
메모리 제한:	256MB

IOI 나라에는 도시 I와 도시 O를 잇는 장거리 버스가 달리고 있다. 버스 안에는 정수기가 있다. 탑승객은 정수기에서 물을 받아 마실 수 있다. 버스는 시각 0에 도시 I로부터 출발하여 시각 X에 도시 O로 도착한다. 버스가 가는 도중 정수기에 물을 보충할 수 있는 장소가 N개 있다. 버스는 i 번째 $(1 \leq i \leq N)$ 장소에 시각 S_i에 도착할 것이다.

처음에 정수기에는 물이 들어있지 않다. 버스가 출발하기 전에 정수기에 물을 채울 수 있다. 또한 버스가 물을 보충할 수 있는 장소에 도착했을 때에 물을 채울 수 있다. 물은 버스가 어디 있든 1리터를 채우는 데에 W 엔이 든다.

도시 I에서 M 명의 승객이 버스에 탑승한다. 승객들은 1번부터 M번까지 번호가 붙어있다. 도시 I를 제외하고는 승객이 버스에 타지 않는다. j번 $(1 \leq j \leq M)$ 승객은 시각 D_j에 1리터의 물을 마시고 싶어 한다. 또한, 물을 마시면 시간 T가 지난 후에 1리터의 물을 마시고 싶어 한다. 다른 말로 j번 승객은 시각 $D_j + kT$에 $(k = 0, 1, 2, \cdots)$ 물을 마시고 싶어 한다. 여기서 $1 \leq D_j < T$를 만족하며 모든 승객에 대해 T는 모두 같은 값이다. 만약 승객이 물을 마시고 싶을 때 정수기에 물이 담겨있지 않다면 승객은 버스에서 내리게 된다. 만약 j번 손님이 도시 O에 도착하기 전에 버스에서 내리게 된다면 승객에게 C_j 엔을 환불해 줘야 한다.

버스 기사도 역시 물을 마셔야 한다. 버스 기사는 시각 0에 1리터의 물을 마시고 싶어 한다. 승객과 마찬가지로 물을 마시면 그는 시간 T가 지난 후에 1리터의 물을 마시고 싶어 한다. 다른 말로, 버스기사는 시각 kT에 $(k = 0, 1, 2, \cdots)$ 물을 마시고 싶어 한다. 만약 버스 기사가 물을 마시고 싶을 때 원할 때 정수기에 물이 담겨있지 않다면 버스는 더 운행할 수 없다.

어떠한 두 사람도 물을 마셔야 하는 시간이 같지 않다. 또한, 버스가 도시 O나 물을 보충할 수 있는 장소에 도달했을 때 물을 마셔야 하는 승객 혹은 버스기사가 있는 경우는 없다.

물을 보충할 수 있는 각 장소에서 물을 담는 양을 조절해서 도시 O까지 가는 도중에 물의 비용과 환불을 해 주는 데 드는 비용의 합을 최소로 하고 싶다. 당신은 여행하는 도중 어디서 얼마나 물을 담아야 하는지를 결정해야 한다.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 다섯 정수 X, N, M, W, T가 주어진다. 이는 도시 O에 버스가 시각 X에 도착하며, 물을 보충할 수 있는 장소가 N 개 있으며, M 명의 손님이 버스에 타고 있으며, 물의 가격이 1리터당 W 엔이며, 승객과 버스 기사가 물을 마시는 간격이 T라는 의미이다.
- 다음 N개의 줄의 i 번째 $(1 \leq i \leq N)$ 줄에는 정수 S_i가 주어진다. 이는 버스가 i 번째 물을 보충할 수 있는 장소에 시간 S_i에 도착한다는 의미이다.
- 다음 M개의 줄의 j 번째 $(1 \leq j \leq M)$ 줄에는 공백으로 구분된 두 정수 D_j, C_j가 주어진다. 이는 j번 승객이 시간 D_j에 처음 물을 마시기를 원하며, 환불할 때 C_j엔을 환불해 주어야 한다는 의미이다.

출력 형식

표준 출력으로 한 개의 줄을 출력하여라. 출력은 최소 비용이다.

제한

- $1 \leq X \leq 1\ 000\ 000\ 000\ 000$.
- $1 \leq N \leq 200\ 000$.

- $1 \leq M \leq 200\ 000$.
- $1 \leq W \leq 1\ 000\ 000$.
- $1 \leq T \leq X$.
- $1 \leq S_i < X$ $(1 \leq i \leq N)$.
- $1 \leq D_j < T$ $(1 \leq j \leq M)$.
- $1 \leq C_j \leq 1\ 000\ 000\ 000$ $(1 \leq j \leq M)$.
- D_j $(1 \leq j \leq M)$ 는 서로 다르다.
- 버스가 도시 O나 물을 보충할 수 있는 장소에 도달했을 때 물을 마셔야 하는 승객 혹은 버스 기사가 있는 경우는 없다.

서브태스크 1 (16 점)

- $N \leq 8$
- $M \leq 8$

서브태스크 2 (30 점)

- $N \leq 100$
- $M \leq 100$

서브태스크 3 (25 점)

- $N \leq 2\ 000$
- $M \leq 2\ 000$

서브태스크 4 (29 점)

추가 제한조건이 없다.

예제

standard input	standard output
19 1 4 8 7 10 1 20 2 10 4 5 6 5	103

이 예제에서, 우리가 출발 전에 7리터의 물을 넣고 물을 보충할 수 있는 장소에서 4리터의 물을 넣은 경우 버스는 다음과 같이 운행된다:

1. 버스가 도시 I를 떠난다. 이 때, 정수기는 7리터의 물이 담겨있다.
2. 버스 기사와 1, 2, 3, 4번 승객이 각각 시각 0, 1, 2, 4, 6에 물을 마신다. 정수기에 남은 물의 양은 2 리터이다.

3. 버스 기사와 1번 승객이 각각 1리터의 물을 시각 7, 8에 마신다. 정수기에 남은 물의 양은 0리터이다.
4. 시간 9에, 2번 승객은 물을 마시고 싶어 하지만 정수기에 물이 없기 때문에 버스를 떠난다.
5. 시간 10에, 물을 보충할 수 있는 장소에서 4리터의 물을 정수기에 보충한다. 이때, 정수기에는 4리터의 물이 담겨있다.
6. 3, 4번 승객, 버스 기사, 1번 승객이 각각 시각 11, 13, 14, 15에 물을 마신다. 정수기에 남은 물의 양은 0리터이다.
7. 시간 18에, 3번 승객은 물을 마시고 싶어 하지만 정수기에 물이 없기 때문에 버스를 떠난다.
8. 시간 19에, 버스는 도시 O에 도착한다.

사용한 물의 총량은 11리터다. 물값은 88엔이다. 2, 3번 승객을 환불해 줄 때 사용한 돈의 양은 총 15엔이다. 총 사용한 돈의 양은 103엔이다.

103엔보다 더 작은 양을 사용하여 버스를 운영하는 것은 불가능하므로, 103엔을 출력한다.

standard input	standard output
105 3 5 9 10 59 68 71 4 71 6 32 7 29 3 62 2 35	547
1000000000000 1 1 1000000 6 999999259244 1 123456789	333333209997456789

문제 8. 긴 저택

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	3초
메모리 제한:	256MB

JOI군의 집 근처에는 긴 저택이 있다. 이 저택은 동에서 서로 일렬로 N 개의 방이 있고, 가장 동쪽에서 i 번째 방을 i번 방이라고 부르며, i번 ($1 \leq i \leq N-1$) 방과 $i+1$번 방을 잇는 통로가 있어서 양방향으로 오갈 수 있다. 방에서 통로로 들어가기 위해서는 열쇠가 필요하다. 각 열쇠에는 종류를 나타내기 위한 수가 하나씩 쓰여 있다. 여러 열쇠에 같은 수가 쓰여 있을 수도 있다.

i번 방 혹은 $i+1$번 방으로부터 두 방을 잇는 통로로 들어가기 위해서는 수 C_i가 쓰인 열쇠가 필요하다.

i번 방에는 B_i 개의 열쇠가 있다. 이 열쇠에는 각각 수 $A_{i,j}$ ($1 \leq j \leq B_i$) 가 붙어있다. 만약 JOI군이 방에 들어가면 그는 방에 있는 모든 열쇠를 집을 것이다. 그 이후에는 집은 열쇠들을 통로를 출입하는데 사용할 수 있다.

JOI군은 열쇠를 원하는 횟수만큼 사용할 수 있다. 가끔 JOI군은 같은 수가 쓰인 열쇠를 여러 개 가지고 있는 경우도 있다. 하지만, 그 열쇠가 하나 있는 것과 특별히 다른 점은 없다.

JOI군이 길을 잃는 경우를 방지하기 위해서 다음 종류의 질문에 대답하는 프로그램을 작성하려고 한다.

- JOI군이 어떠한 열쇠도 가지지 않고 x번 방에서 시작하여 y번 방으로 이동할 수 있을까?

JOI군을 대신하여 이 문제를 답할 수 있는 프로그램을 작성하여 주자.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 정수 N이 주어진다. 이는 저택에 있는 방의 개수다.
- 둘째 줄에는 $N-1$ 개의 공백으로 구분된 정수 $C_1, C_2, \cdots, C_{N-1}$이 주어진다. 이는 우리가 i번 방과 $i+1$번 방을 잇는 통로를 오가기 위해서 수 C_i가 쓰인 열쇠가 필요하다는 의미이다.
- 다음 N 개의 줄의 i 번째 ($1 \leq i \leq N$) 줄에는 정수 B_i와, B_i 개의 공백으로 구분된 정수 $A_{i,1}, A_{i,2}, \cdots, A_{i,B_i}$가 주어진다. 이는 i번 방에 B_i 개의 열쇠가 있으며, 열쇠에 각각 수 $A_{i,j}$ ($1 \leq j \leq B_i$) 가 쓰여 있다는 의미이다.
- 다음 줄에는, 질문의 개수 Q가 주어진다.
- 다음 Q 개의 줄의 k 번째 ($1 \leq k \leq Q$) 줄에는 공백으로 구분된 두 정수 X_k, Y_k가 주어진다. 이는 k 번째 질문이 JOI군이 어떠한 열쇠도 가지지 않고 X_k번 방에서 시작하여, Y_k번 방으로 이동할 수 있는지를 묻는다는 의미이다.

출력 형식

표준 출력으로 Q개의 줄을 출력하여라. Q개의 줄의 k 번째 ($1 \leq k \leq Q$) 줄은 JOI군이 어떠한 열쇠도 가지지 않고 X_k번 방에서 시작하여, Y_k번 방으로 이동할 수 있으면 `YES`, 아니면 `NO`여야 한다.

제한

- $2 \leq N \leq 500\ 000$.
- $1 \leq Q \leq 500\ 000$.
- $1 \leq B_1 + B_2 + \cdots + B_N \leq 500\ 000$.

- $1 \leq B_i \leq N$ $(1 \leq i \leq N)$.
- $1 \leq C_i \leq N$ $(1 \leq i \leq N-1)$.
- $1 \leq A_{i,j} \leq N$ $(1 \leq i \leq N, 1 \leq j \leq B_i)$.
- B_i개의 정수 $A_{i,1}, A_{i,2}, \cdots, A_{i,B_i}$는 서로 다르다.
- $1 \leq X_k \leq N$ $(1 \leq k \leq Q)$.
- $1 \leq Y_k \leq N$ $(1 \leq k \leq Q)$.
- $X_k \neq Y_k$ $(1 \leq k \leq Q)$.

서브태스크 1 (5 점)

- $N \leq 5\ 000$
- $Q \leq 5\ 000$
- $1 \leq B_1 + B_2 + \cdots + B_N \leq 5\ 000$.

서브태스크 2 (5 점)

- $N \leq 5\ 000$
- $1 \leq B_1 + B_2 + \cdots + B_N \leq 5\ 000$.

서브태스크 3 (15 점)

- $N \leq 100\ 000$
- $C_i \leq 20$ $(1 \leq i \leq N-1)$.
- $1 \leq A_{i,j} \leq 20$ $(1 \leq i \leq N, 1 \leq j \leq B_i)$.

서브태스크 4 (75 점)

추가 제한조건이 없다.

예제

standard input	standard output
5 1 2 3 4 2 2 3 1 1 1 1 1 3 1 4 4 2 4 4 2 1 5 5 3	YES NO NO YES

- 첫째 질문에서, JOI군이 방을 2, 1, 2, 3, 4번 방 순서로 방문한다면 4번 방에 도착할 수 있다.
- 둘째 질문에서, JOI군은 3, 4번 방 밖에 방문할 수 없다. 1과 3이 쓰여 있는 열쇠 밖에 얻을 수 없으므로 2번 방에 들어가지 못한다.
- 셋째 질문에서, JOI군은 4번 방에서 5번 방으로 가기 위한 4가 쓰인 열쇠를 얻을 수가 없으므로 5번 방에 들어가지 못한다.
- 넷째 질문에서, JOI군이 방을 5, 4, 3번 방 순서로 방문한다면 3번 방에 도착할 수 있다.

standard input	standard output
5 2 3 1 3 1 3 1 2 1 1 1 3 1 2 4 1 3 3 1 4 3 2 5	NO YES NO YES
7 6 3 4 1 2 5 1 1 1 5 1 1 1 1 2 2 3 1 4 1 6 3 4 1 5 3 4 7	YES NO YES

문제 9. 자연공원

시간 제한: 2초
메모리 제한: 256MB

JOI 섬은 섬 전체가 자연공원으로 지정된 관광지이다.

JOI 섬에는 N 개의 광장과 몇 개의 도로가 있다. 광장에는 0번부터 $N-1$번까지 번호가 붙어있다. 도로는 섬 내에 서로 다른 2개의 광장을 잇고, 양방향으로 이동할 수 있다. 어떤 광장에 대해서도 이 광장에 직접 연결된 도로는 최대 7개이다. 서로 다른 두 광장에 대해서 두 광장을 잇는 도로는 최대 한 개 존재한다. 몇 개의 도로를 거치면 한 광장에서부터 다른 광장으로 갈 수 있다.

당신과 당신의 친구 IOI양은 JOI 섬을 탐험할 것이다. 효율적인 탐사를 위해 JOI 섬의 구조를 파악할 필요가 있다. 섬에는 위험한 야생동물이 많아 위험하므로 운동신경이 좋은 IOI양이 도시를 탐색하고 IOI양의 보고를 받아 당신이 섬의 구조를 파악하게 되었다.

당신은 IOI양에게 두 광장의 번호 A, B와 경유 가능한 광장을 몇 개 지정하여 광장 A부터 광장 B까지 지정된 광장만 경유하여 이동하는 것이 가능하냐는 질문을 한다. IOI양은 질문의 내용을 따라 섬을 탐색하여 결과를 보고한다.

조사에 긴 시간이 사용되면 위험하기 때문에 질문 횟수는 45 000번 이내여야 한다.

구현 명세

당신은 섬의 구조를 파악하는 방법이 구현된 하나의 파일을 작성해야 한다. 이 파일은 `park.h`를 include해야 한다.

이 파일에는, 다음 함수가 작성되어 있어야 한다.

- `void Detect(int T, int N)`

  이 함수는 한 번만 불린다.

  - 인자 `T`는 서브태스크의 번호, `N`은 광장의 개수를 의미한다.

  프로그램 안에서 다음의 함수를 호출하여, JOI 섬의 구조를 출력해야 한다.

  - `void Answer(int A, int B)`

    이 함수의 호출 횟수는 JOI 섬의 도로의 개수와 같아야 한다.

    * 인자 `A`, `B`는 `A`번 광장과 `B`번 광장 사이에 도로가 있다는 것을 의미한다.

    함수 인자는 다음과 같은 조건을 만족해야 한다:

    * `A`, `B`는 $0 \leq$`A`$<$`B`$\leq N-1$을 만족해야 한다. 이 조건을 만족하지 않으면 **오답 [1]**이 된다.
    * 함수가 인자 (`A`, `B`)로 호출되었으면 `A`번 광장과 `B`번 광장을 잇는 도로가 있어야 한다. 이 조건을 만족하지 않으면 **오답 [2]**이 된다.
    * 이 함수는 같은 인자 (`A`, `B`)로 두 번 이상 호출되면 안 된다. 이 조건을 만족하지 않으면 **오답 [3]**이 된다.

  또한, 당신의 프로그램은 다음 함수를 호출 할 수 있다.

- `int Ask(int A, int B, int Place[])`

  이 함수는 IOI양에게 질문을 하는 데에 쓰인다.

  - `Place`는 경유할 수 있는 광장들의 배열을 가리키는 포인터이다. 각 i $(0 \leq i \leq N-1)$에 대해, `Place[`i`]` $=1$이면, i번 광장을 경유할 수 있다는 의미이고, `Place[`i`]` $=0$이면, i번 광장을 경유할 수 없다는 의미이다.
  - 이 함수의 반환 값은 `A`번 광장에서 `B`번 광장으로 배열 `Place[]`에 주어진 광장만으로 이동할 수 있으면 1이고, 아니면 0이다.

  함수 인자는 다음과 같은 조건을 만족해야 한다:

- $0 \leq$ `A` $<$ `B` $\leq N-1$
- $0 \leq$ `Place[`i`]` ≤ 1 $(0 \leq i \leq N-1)$
- `Place[A]` $= 1$
- `Place[B]` $= 1$

위 조건들을 만족하지 않으면 **오답** [4]이 된다. 하지만 배열 `Place[]`의 길이가 N이 아니면 이 함수의 동작은 보장되지 않는다.

함수 `Ask`는 45 000번 초과로 호출되어서는 안 된다. 만약 초과하는 경우, **오답** [5]이 된다.

함수가 종료되었을 때, `Answer`의 호출로 출력되지 않은 도로가 있는 경우에는 **오답** [6]이 된다.

당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다. 당신의 프로그램은 표준 입출력을 사용해서는 안 된다. 당신의 프로그램은 어떠한 방법으로도 다른 파일에 접근해서는 안 된다.

당신은 대회 홈페이지의 아카이브에서 프로그램을 테스트하기 위한 목적의 샘플 그레이더를 받을 수 있다. 아카이브는 당신의 프로그램의 예제 소스 또한 첨부되어 있다. 샘플그레이더는 파일 `grader.c` 혹은 `grader.cpp`이다. 당신의 프로그램이 `park.c` 혹은, `park.cpp` 인 경우 다음 커맨드로 컴파일할 수 있다.

- C `g++ -std=c11 -O2 -o grader grader.c park.c -lm`
- C++ `g++ -std=c++14 -O2 -o grader grader.cpp park.cpp`

컴파일이 성공적이면, 파일 `grader`가 생성된다.

실제 그레이더와 샘플 그레이더는 다름에 주의하여라. 샘플 그레이더는 하나의 프로세스에서 실행되며, 입력을 표준 입력으로부터 받고, 출력을 표준 출력에 출력한다.

입력 형식

샘플 그레이더는 다음 형식으로 표준 입력으로부터 데이터를 입력받는다.

- 첫 번째 줄에 서브태스크의 번호 T가 주어진다.
- 두 번째 줄에 광장의 개수 N이 주어진다.
- 세 번째 줄에 도로의 개수 M이 주어진다.
- 다음 M 개의 줄의 i 번째 $(1 \leq i \leq M)$ 줄에는 공백으로 구분된 두 정수 A_i, B_i가 주어진다. 이는 A_i번 광장과 B_i번 광장을 잇는 도로가 있어서 양방향으로 오갈 수 있다는 의미이다.

출력 형식

프로그램이 정상적으로 종료되었다면, 샘플 그레이더는 다음과 같은 정보를 표준 출력에 출력한다. (따옴표는 출력하지 않는다.)

- 프로그램이 정답으로 판단된 경우에는, "`Accepted.`"를 출력한다.
- 오답으로 판단된 경우, 오답의 종류를 "`Wrong Answer [1]`"과 같은 형식으로 출력하고 프로그램이 종료된다.

프로그램이 다양한 오답의 종류에 속해 있으면 샘플 그레이더는 그중 하나만 출력할 것이다.

제한

T, N, M의 의미에 대해서는 입력 형식을 참고하여라.

- $1 \leq T \leq 5$.

- $2 \leq N \leq 1\ 400$.
- $1 \leq M \leq 1\ 500$.
- 어떤 광장에 대해서도 이 광장에 직접 연결된 도로는 최대 7개이다.
- 몇 개의 도로를 거치면 한 광장에서부터 다른 광장으로 갈 수 있다.
- 서로 다른 두 광장에 대해서 두 광장을 잇는 도로는 최대 한 개 존재한다.

서브태스크 1 (10 점)

- $T = 1$.
- $N \leq 250$.

서브태스크 2 (10 점)

- $T = 2$.
- $M = N - 1$.
- 0번과 $N - 1$번 광장에 대해 다른 광장으로 가는 도로가 정확히 한 개 존재한다. 다른 모든 장소에 대해서는 다른 광장으로 가는 도로가 정확히 두 개 존재한다.

서브태스크 3 (27 점)

- $T = 3$.
- $M = N - 1$.
- 모든 i $(0 \leq i \leq N - 1)$에 대해, 0번 광장으로부터 i번 광장으로까지 최대 8개의 다른 광장을 거치면 오갈 수 있다.

서브태스크 4 (30 점)

- $T = 4$.
- $M = N - 1$.

서브태스크 5 (23 점)

- $T = 5$.

예제

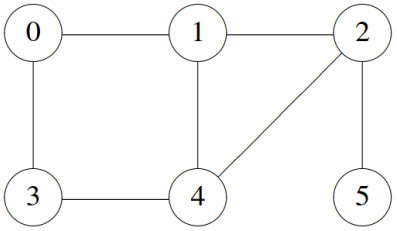
예제 입력과 이에 해당하는 함수 호출을 보여준다.

예제 입력	예제 함수 호출	
	호출	반환값
1 6 7 0 1 0 3 1 2 1 4 2 4 2 5 3 4	`Ask(3, 5, {0,0,1,1,1,1})`	1
	`Answer(2, 4)`	
	`Answer(3, 5)`	
	`Answer(3, 4)`	
	`Ask(0, 4, {1,0,1,0,1,0})`	0
	`Answer(0, 1)`	
	`Answer(0, 3)`	
	`Answer(1, 4)`	
	`Answer(1, 2)`	

이 함수 호출이 꼭 의미가 있는 것은 아니다.

이 예제에서, 함수 `Detect`는 인자 `T`= 1, `N`= 6으로 호출되었다.

이 예제에서, JOI 섬의 구조는 다음과 같다.



JOI 섬의 구조

원과 숫자는 광장과 그 번호를 의미한다. 선분은 도로를 의미한다.

- 첫 번째 `Ask` 함수 호출은 3번 광장부터 5번 광장까지 2, 3, 4, 5번 광장만 경유하여 갈 수 있는지를 물어보았다. 가능하므로, 함수 `Ask`는 1을 반환한다.
- 두 번째 `Ask` 함수 호출은 0번 광장부터 4번 광장까지 0, 2, 4번 광장만 경유하여 갈 수 있는지를 물어보았다. 불가능하므로, 함수 `Ask`는 0을 반환한다.

문제 10. 유괴2

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	5초
메모리 제한:	512MB

어느 화창한 날 도시에서 유괴사건이 일어났다. 범인은 Anna와 Bruno고 차를 통해 유괴 현장에서 도망쳤다고 추정하고 있다. 차는 아직 발견되지 않았다. 경찰은 아직도 차의 행방을 쫓고 있다.

유괴범은 H개의 동서 방향 도로가 있고 W개의 남북 방향 도로가 있는 격자 모양의 도시에서 차를 운전하고 있다. 두 교차로의 사이는 1km이다.

각 도로는 **혼잡도**라고 불리는 정수가 붙어있다. 북쪽에서 i 번째 ($1 \leq i \leq H$) 동서 방향 도로의 혼잡도는 A_i 이고, 서쪽에서 j 번째 ($1 \leq j \leq W$) 남북 방향 도로의 혼잡도는 B_j 이다. 이 $H + W$개의 값은 서로 다르다. 각 도로에 대해 혼잡도는 어느 지점에 있든 같다.

조사는 유괴범이 다음과 같은 방법으로 이동했다는 것을 알아냈다.

- 도시 밖이나, 도로 밖으로 나가지는 않았다.
- 처음 유괴범은 유괴한 교차로으로부터 이동 가능한 방향 중 어떤 한 방향을 택해 움직였다.
- 어떤 교차로에 도착했을 때, 현재 달리는 방향의 도로보다 교차하는 다른 도로의 혼잡도가 더 크면 그 교차로에서 회전한다. 회전 할 수 있는 방향이 둘이면 양 쪽 모두 고를 가능성이 있다.
- 어떤 교차로에 도착했을 때, 현재 달리는 방향의 도로가 교차하는 다른 도로보다 혼잡도가 더 크면 계속 직진한다. 만약 도시의 경계에 도달하여 직진할 수 없을 때는 그 자리에서 멈춘다.

유괴사건이 일어났을 거라고 추정되는 Q개의 후보지가 있다. 이 Q개의 후보지는 서로 다르다. 조사팀의 사람을 정하기 위해서 경찰은 각 후보지에 대해서 유괴사건이 그 후보지에서 일어났을 경우에 범죄자가 운전할 수 있는 최대 거리를 알고 싶다.

각 Q개의 질의에 대해, 후보지에 대해서 범죄자가 운전할 수 있는 최대 거리를 구하여라.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 두 정수 H, W, Q가 주어진다. 이는 도시가 동서방향 도로가 H 개, 남북방향 도로가 W 개 있으며, 범죄 후보지가 Q 개 라는 의미이다.
- 둘째 줄에는 공백으로 구분된 H 개의 정수 A_1, A_2, $\cdots$, A_H가 주어진다. 이는 북쪽에서 i 번째 ($1 \leq i \leq H$) 동서 방향 도로의 혼잡도가 A_i라는 의미이다.
- 셋째 줄에는 공백으로 구분된 W 개의 정수 B_1, B_2, $\cdots$, B_W가 주어진다. 이는 서쪽에서 j 번째 ($1 \leq j \leq W$) 남북 방향 도로의 혼잡도가 B_j라는 의미이다.
- 다음 Q 개의 줄의 k 번째 ($1 \leq k \leq Q$) 줄에는 공백으로 구분된 두 정수 S_k, T_k가 주어진다. k 번째 유괴사건 후보지가 북쪽에서 S_k 번째 동서 방향 도로와 서쪽에서 T_k 번째 남북 방향 도로의 교차로라는 의미이다.

출력 형식

표준 출력으로 Q 개의 줄을 출력하여라. k 번째 줄은 k 번째 후보지에 대해서 범죄자가 운전할 수 있는 최대 거리를 (km단위로) 출력해야 한다.

제한

- $2 \leq H \leq 50\ 000$.
- $2 \leq W \leq 50\ 000$.
- $2 \leq Q \leq 100$.
- $1 \leq A_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq H)$.
- $1 \leq B_j \leq 1\ 000\ 000\ 000$ $(1 \leq j \leq W)$.
- $H + W$개의 정수 $A_1, A_2, \cdots, A_H, B_1, B_2, \cdots, B_W$는 서로 다르다.
- $1 \leq S_k \leq H$ $(1 \leq k \leq Q)$.
- $1 \leq T_k \leq W$ $(1 \leq k \leq Q)$.
- $(S_k, T_k) \neq (S_l, T_l)$ $(1 \leq k < l \leq Q)$.

서브태스크 1 (13 점)

- $H \leq 8$
- $W \leq 8$
- $Q = 1$

서브태스크 2 (10 점)

- $H \leq 2\ 000$
- $W \leq 2\ 000$
- $Q = 1$

서브태스크 3 (17 점)

- $Q = 1$

서브태스크 4 (4 점)

- $H \leq 2\ 000$
- $W \leq 2\ 000$

서브태스크 5 (56 점)

추가 제한조건이 없다.

예제

standard input	standard output
3 3 5 3 2 6 1 4 5 1 1 1 2 2 2 3 1 3 3	4 5 4 4 2

예를 들어, 세 번째 질의에 대해서 운전자가 이동한 거리는 다음 방법으로 최대가 된다.

- 북쪽에서 두 번째 동서 방향 도로와 서쪽에서 두 번째 남북 방향 도로의 교차로에서 동쪽으로 1km 움직였다.
- 북쪽에서 두 번째 동서 방향 도로와 서쪽에서 세 번째 남북 방향 도로의 교차로에서 남쪽 혹은 북쪽으로 움직일 수 있다. 남쪽을 골라서 1km 움직였다.
- 북쪽에서 세 번째 동서 방향 도로와 서쪽에서 세 번째 남북 방향 도로의 교차로에서 서쪽으로만 움직일 수 있다. 서쪽으로 1km 움직였다.
- 북쪽에서 세 번째 동서 방향 도로와 서쪽에서 두 번째 남북 방향 도로의 교차로에서 서쪽으로만 움직일 수 있다. 서쪽으로 1km 움직였다.
- 북쪽에서 세 번째 동서 방향 도로와 서쪽에서 첫 번째 남북 방향 도로의 교차로에서 더 움직일 수 없다. 그 장소에서 멈췄다.

위와 같이 움직인 경우에 이동한 거리는 4km 이다.

standard input	standard output
4 5 6 30 10 40 20 15 55 25 35 45 1 3 4 3 2 2 4 1 2 5 3 3	7 6 9 4 6 9

문제 11. 도시

시간 제한: 3초
메모리 제한: 256MB

JOI 왕국에는 다양한 도시가 있다. 도로 체계는 다음 조건을 만족한다:

- (조건 1) 도시는 0번부터 $N-1$번 까지의 번호가 붙어있다. 여기서, N은 JOI 왕국의 도시의 개수다.
- (조건 2) 도시는 $N-1$개의 도로로 연결되어 있다. 도로는 양방향으로 통행할 수 있다. 어떤 도시에서 다른 도시까지 몇개의 도로를 거치면 통행할 수 있다.
- (조건 3) 0번 도시로부터 다른 도시까지 최대 18개의 도로를 거치면 갈 수 있다.

어느 날 JOI 왕국에서 많은 사람이 0번 도시에서 다른 도시로 옮겨갔다. 많은 JOI 왕국의 국민이 서로 다른 두 개의 도시를 목적지로 하고 있었기 때문에 다음과 같은 질문을 했다. 2개의 서로 다른 두 도시 X, Y에 대해 (0), (1), (2) 중에 어느 것이 성립하는가?

(0) 0번 도시부터 도시 X 까지 가는 도중에 도시 Y를 반드시 거친다.

(1) 0번 도시부터 도시 Y 까지 가는 도중에 도시 X를 반드시 거친다.

(2) (0)과 (1) 모두 아니다.

위의 조건에서 (0), (1), (2) 중에 정확히 하나가 성립한다. 단, X가 0번 도시인 경우에 Y에 상관없이 (1)이 성립하는 것으로 한다. 또한, Y가 0번 도시인 경우에 X에 상관없이 (0)이 성립하는 것으로 한다.

JOI 왕국 뿐만 아니라 다른 나라에 대해서도 도로 체계의 (조건 1)~(조건 3)이 성립하는 것이 잘 알려져 있다. 그렇기 때문에 다른 나라에서도 쓸 수 있게 JOI 왕국에는 다음과 같은 2개의 기계를 만들려고 한다.

- (기계 1) 도시의 개수 N과 도로의 정보를 보고, 각각의 도시에 0 이상 $2^{60}-1$ 이하의 정수를 붙인다.
- (기계 2) 도시 X, Y에 대해 (기계 1)이 도시 X, Y에 부여한 번호를 보고 질문에 대해 대답한다.

큰 번호가 붙을 경우 이를 관리하는 것은 힘들다. 그렇기 때문에 가능한 한 적은 번호가 붙게 기계를 개발하려 한다.

기계 2가 사용될 때 도시의 수 N이나 도로의 정보를 주지 않음에 유의하여라.

구현 명세

당신은 *같은 프로그래밍 언어로 작성된* 파일 두 개를 작성해야 한다.

첫 번째 파일의 이름은 `Encoder.c` 혹은 `Encoder.cpp`이다. 이 파일은 기계 1을 구현한 파일이며, 다음 함수를 구현해야 한다. 이 파일은 `Encoder.h`를 include해야 한다.

- `void Encode(int N, int A[], int B[])`
  이 함수는 각 테스트 케이스마다 정확히 한 번 불린다.
  - 인자 `N`은 도시의 개수 N을 의미한다.
  - 인자 `A[]`, `B[]`는 길이 $N-1$의 배열이고, 도로의 정보를 의미한다. 원소 `A[i]`, `B[i]` ($0 \leq$ `i` $\leq N-2$)는 `A[i]`번 도시와 `B[i]`번 도시를 직접 연결하는 도로가 있다는 것을 의미한다.

  당신의 프로그램은 다음의 함수를 호출해야 한다.
  - `void Code(int city, long long code)`
    이 함수는 도시에 정수를 붙이는 것을 의미한다.

* 인자 city는 정수를 붙일 도시의 번호를 의미한다. city는 0 이상 $N-1$ 이하이다. 만약에 이 범위를 벗어나서 함수를 호출 한 경우 **오답** [1]이 된다. 같은 city를 인자로 하여 함수를 두 번 이상 호출한 경우 **오답** [2]이 된다.
* 인자 code는 city번 도시에 붙일 정수이다. code는 값은 0 이상 $2^{60}-1$이어야 한다. 만약에 이 범위를 벗어나서 함수를 호출한 경우 **오답** [3]이 된다.

함수 Code는 프로그램에서 정확히 N번 호출되어야 한다. Encode 함수가 종료되었을 때, Code이 호출된 횟수가 N과 다르면 **오답** [4]이 된다.

만약 Encode가 함수를 올바르지 않게 호출될 경우 프로그램이 종료된다.

두 번째 파일의 이름은 Device.c 혹은 Device.cpp이다. 이 파일은 기계 2를 구현한 파일이며, 다음 함수를 구현해야 한다. 이 파일은 Device.h를 include해야 한다.

- `void InitDevice()`

  이 함수는 기계 2의 초기화에 대응한다. 다음 Answer 함수가 불리기 전에 InitDevice는 정확히 한 번만 불린다.

- `int Answer(long long S, long long T)`

  이 함수는 각 질문에 대응한다. 각 질문에 대응하여 Answer가 한 번 불린다.

  - 인자 S, T는 두 개의 서로 다른 도시 X, Y에 붙은 정수이다.
  - 함수 Answer는 질문에 답하기 위해, 다음의 조건을 만족하는 값을 반환해야 한다.
    * 0번 도시부터 도시 X 까지 가는 도중에 도시 Y를 반드시 거치는 경우, 0을 반환한다.
    * 0번 도시부터 도시 Y 까지 가는 도중에 도시 X를 반드시 거치는 경우, 1을 반환한다.
    * 위 두 경우 모두 아닌 경우 2를 반환한다.

    즉, Answer의 반환값은 0 이상 2 이하의 정수여야 한다. 이 범위 밖의 수를 반환한 경우 **오답** [5]이 된다. 또한, 범위 내의 정수여도, 위 조건을 만족하지 않는 값을 반환한 경우 **오답** [6]이 된다.

채점은 다음과 같은 방식으로 진행된다. 만약 프로그램이 오답으로 판단된 경우, 즉시 채점은 종료된다.

1. 함수 Encode를 1회 호출한다.
2. 함수 InitDevice를 1회 호출한다.
3. 각 테스트케이스에 대해, 기계 2에게 Q개의 질문을 준다. j 번째 $(1 \leq j \leq Q)$ 질문에 대해, 함수 Answer는 인자 S에 S_j를, 인자 T에 T_j를 넣어서 호출되고, S_j와 T_j는 함수 Encode가 X_j번과 Y_j번 도시에 설정한 값이다.
4. 당신의 프로그램은 정답이 된다.

참고 사항

- 실행 시간과 메모리 사용량은 채점 방식의 1, 2, 3에서 계산된다.
- 당신의 프로그램은 채점 방식 1.의 Encode, 채점 방식 2.의 InitDevice 혹은 에서 채점 방식 3.의 Answer에서 오답으로 판단되면 안된다. 당신의 프로그램은 런타임 에러 없이 실행되어야 한다.
- 당신의 프로그램은 내부에서 사용할 목적으로 함수나 전역변수를 사용할 수 있다. 제출한 프로그램은 그레이더와 함께 컴파일 되어 하나의 실행파일이 된다. 모든 전역변수나 내부 함수는 다른 파일과의 충돌을 피하기 위해 static으로 선언되어야 한다. 기계 1과 기계 2는 2개의 별개의 프로세스로 실행되기 때문에 채점될 때 전역변수를 공유하지 않는다.

- 당신의 프로그램은 표준 입출력을 사용해서는 안 된다. 당신의 프로그램은 어떠한 방법으로도 다른 파일에 접근해서는 안 된다.

당신은 대회 홈페이지의 아카이브에서 프로그램을 테스트하기 위한 목적의 샘플 그레이더를 받을 수 있다. 아카이브는 당신의 프로그램의 예제 소스 또한 첨부되어 있다. 샘플 그레이더는 파일 `grader.c` 혹은 `grader.cpp`이다. 당신의 프로그램이 `Encoder.c`와 `Device.c` 혹은, `Encoder.cpp`와 `Device.cpp` 인 경우 다음 커맨드로 컴파일할 수 있다.

- C `g++ -std=c11 -O2 -o grader grader.c Encoder.c Device.c -lm`
- C++ `g++ -std=c++14 -O2 -o grader grader.cpp Encoder.cpp Device.cpp`

컴파일이 성공적이면, 파일 `grader`가 생성된다.

실제 그레이더와 샘플 그레이더는 다름에 주의하여라. 샘플 그레이더는 하나의 프로세스에서 실행되며, 입력을 표준 입력으로 부터받고, 출력을 표준 출력에 출력한다.

입력 형식

샘플 그레이더는 다음 형식으로 표준 입력으로 부터 데이터를 입력받는다.

- 첫째 줄에 두 개의 정수 N, Q가 공백으로 구분되어 주어진다. 이는 N 개의 도시가 있고, 질문이 Q 개라는 의미이다.
- 다음 $N-1$ 개의 줄의 $i+1$ 번째 $(0 \leq i \leq N-2)$ 줄에는, 두 개의 정수 A_i와 B_i가 공백으로 구분되어 주어진다. 이는 A_i번 도시와 B_i번 도시가 도로로 직접 연결되어 있다는 의미이다.
- 다음 Q개의 줄의 j 번째 $(1 \leq j \leq Q)$ 줄에는, 3개의 정수 X_j, Y_j, E_j가 공백으로 구분되어 주어진다. 이는 j번째 질문에 대해, $X = X_j$이고 $Y = Y_j$라는 것을 의미하고, 당신의 프로그램이 질문에 대해 E_j 이외의 답을 반환한 경우에는 샘플 그레이더가 오답으로 판단한다는 의미이다.

출력 형식

프로그램이 정상적으로 종료되었다면, 샘플 그레이더는 다음과 같은 정보를 표준 출력에 출력한다. (따옴표는 출력하지 않는다.)

- 정답인 경우, 도시에 붙은 정수의 최댓값을 "`Accepted max_code=123456`"과 같은 형식으로 출력한다.
- 오답으로 판단 된 경우, 오답의 종류를 "`Wrong Answer [1]`"과 같은 형식으로 출력한다.

프로그램이 다양한 오답의 종류에 속해 있으면 샘플 그레이더는 그중 하나만 출력할 것이다.

제한

N, Q, A_i, B_i, X_j, Y_j의 의미에 대해서는 입력 형식을 참고하여라.

- $2 \leq N \leq 250\ 000$.
- $1 \leq Q \leq 250\ 000$.
- $0 \leq A_i \leq N-1$ $(0 \leq i \leq N-2)$.
- $0 \leq B_i \leq N-1$ $(0 \leq i \leq N-2)$.
- $A_i \neq B_i$. $(0 \leq i \leq N-2)$
- 도시 0부터 어떤 다른 도시에 대해서도, 18개 이하의 도로를 사용하여 오가는 것이 가능하다.

- $0 \leq X_j \leq N-1$ $(1 \leq j \leq Q)$.
- $0 \leq Y_j \leq N-1$ $(1 \leq j \leq Q)$.
- $X_j \neq Y_j$ $(1 \leq j \leq Q)$.

서브태스크 1 (8 점)

- $N \leq 10$.

서브태스크 2 (92 점)

추가 제한조건이 없다. 이 서브태스크에 대해서는 다음과 같이 점수가 정해진다.

- 이 서브태스크의 모든 테스트케이스에 대해 도시에 할당된 정수의 최댓값을 L이라고 하자.
- 이 때, 이 서브태스크의 점수는
  - $2^{38} \leq L$인 경우, 0점.
  - $2^{36} \leq L \leq 2^{38}-1$인 경우, 10점.
  - $2^{35} \leq L \leq 2^{36}-1$인 경우, 14점.
  - $2^{34} \leq L \leq 2^{35}-1$인 경우, 22점.
  - $2^{28} \leq L \leq 2^{34}-1$인 경우, $\lfloor 372 - 10\log_2(L+1) \rfloor$점 ($\lfloor x \rfloor$는 x를 넘지 않는 최대의 정수)
  - $L \leq 2^{28}-1$ 인 경우, 92점

채점 시스템에 대해서 $2^{38} \leq L$인 경우에, 이 서브태스크의 정보가 "정답: 0점"과 같이 표시되는 것이 아니라, "오답" 으로 표시됨에 주의하여라.

예제

예제 입력과 이에 해당하는 함수 호출을 보여준다.

예제 입력	예제 함수 호출	
	기계 1	기계 2
	`Encode(...)`	
`6 5`	`Code(0,0)`	
`4 1`	`Code(2,4)`	
`0 3`	`Code(4,16)`	
`4 5`	`Code(1,1)`	
`3 2`	`Code(3,9)`	
`3 4`	`Code(5,25)`	
`2 4 2`		`InitDevice()`
`1 0 0`		`Answer(4,16)`
`5 1 2`		`Answer(1,0)`
`5 3 0`		`Answer(25,1)`
`4 1 1`		`Answer(25,9)`
		`Answer(16,1)`

여기서 `Encode(...)` 호출의 인자들은 다음과 같다.

인자	`Encode(...)`
`N`	6
`A`	{4, 0, 4, 3, 3}
`B`	{1, 3, 5, 2, 4}

문제 12. 용2

입력 파일:	`standard input`
출력 파일:	`standard output`
시간 제한:	3초
메모리 제한:	256MB

JOI 평원에서 사람들은 용과 함께 살고 있다.

JOI 평원은 광활한 좌표평면이고, 직각으로 교차하는 X축과 Y축이 설정되어 있다. X 좌표가 x, Y 좌표가 y인 점을 (x, y)로 표시한다.

JOI 평원은 N 마리의 용이 생활하고 있고 1번부터 N번까지 번호가 붙어있다. 또한 용은 M 종류의 종족이 있어서 1번부터 M번까지 번호가 붙어있다. i번 $(1 \leq i \leq N)$ 용은 평소에는 JOI 평원의 (A_i, B_i)에 살고 있고, C_i번 종족이다. JOI 평원에는 모든 종족의 용이 생활하고 있지 않을 수도 있다.

JOI 평원에 사람이 사는 두 마을은 (D_1, E_1)과 (D_2, E_2)에 있다. 두 도시는 도로로 연결되어 있으며, 이는 두 점을 잇는 선분이다.

점 $(A_1, B_1), \cdots, (A_N, B_N)$과 (D_1, E_1), (D_2, E_2)는 서로 다르며, 어떠한 세 점도 일직선 위에 있지 않다.

가끔 용의 종족 사이에서 대립이 벌어진다. a번 $(1 \leq a \leq M)$ 종족이 b번 $(1 \leq b \leq M,\ a \neq b)$ 종족에 대해 적의를 가지면 a번 종족의 모든 용이 b번 종족의 모든 용을 향해 화염구를 뿜는다. 화염구는 일직선으로 날아가고 맞은 이후에도 계속 날아간다. 즉, 화염구의 궤적은 반직선이다.

종족 사이의 대립이 일어났을 때 도로와 화염구의 궤적이 교차하면 도로는 손상을 입을 것이다. 일어날 수 있는 Q개의 대립이 주어졌을 때, 각각의 대립에 대해서 도로와 교차하는 화염구의 개수를 알고 싶다.

입력 형식

다음 정보가 표준 입력으로 주어진다.

- 첫째 줄에는 공백으로 구분된 두 정수 N, M이 주어진다. 이는 N 마리의 용이 JOI 평원에 살고 있고, M 종류의 종족이 존재한다는 의미이다.
- 다음 N 개의 줄의 i 번째 $(1 \leq i \leq N)$ 줄에는 공백으로 구분된 세 정수 A_i, B_i, C_i가 주어진다. 이는 i번 $(1 \leq i \leq N)$ 용이 (A_i, B_i)에 살고 있고, C_i번 종족이라는 의미이다.
- 다음 줄에는 공백으로 구분된 네 정수 D_1, E_1, D_2, E_2가 존재한다. 이는 사람이 사는 두 마을이 (D_1, E_1), (D_2, E_2)라는 의미이다.
- 다음 줄에는 정수 Q가 주어진다. 이는 일어날 수 있는 대립의 개수가 Q개라는 의미이다.
- 다음 Q개의 줄의 j 번째 $(1 \leq j \leq Q)$ 줄에는 공백으로 구분된 두 정수 F_j, G_j가 주어진다. 이는 가능한 j 번째 대립이 F_j번 종족이 G_j번 종족에게 적의를 품는다는 의미이다.

출력 형식

표준 출력으로 Q 개의 줄을 출력하여라. j 번째 $(1 \leq j \leq Q)$ 줄은 j번째 대립이 일어났을 때 도로와 교차하는 화염구의 개수여야 한다.

제한

- $2 \leq N \leq 30\ 000$.
- $2 \leq M \leq N$.
- $-1\ 000\ 000\ 000 \leq A_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.
- $-1\ 000\ 000\ 000 \leq B_i \leq 1\ 000\ 000\ 000$ $(1 \leq i \leq N)$.

- $1 \leq C_i \leq M$ $(1 \leq i \leq N)$.
- $-1\ 000\ 000\ 000 \leq D_1 \leq 1\ 000\ 000\ 000$.
- $-1\ 000\ 000\ 000 \leq E_1 \leq 1\ 000\ 000\ 000$.
- $-1\ 000\ 000\ 000 \leq D_2 \leq 1\ 000\ 000\ 000$.
- $-1\ 000\ 000\ 000 \leq E_2 \leq 1\ 000\ 000\ 000$.
- $N+2$개의 점 $(A_1, B_1), \cdots, (A_N, B_N)$, (D_1, E_1), (D_2, E_2)는 서로 다르며, 어떠한 세 점도 일직선 위에 있지 않다.
- $1 \leq Q \leq 100\ 000$.
- $1 \leq F_j \leq M$ $(1 \leq j \leq Q)$.
- $1 \leq G_j \leq M$ $(1 \leq j \leq Q)$.
- $F_j \neq G_j$ $(1 \leq j \leq Q)$.
- $(F_j, G_j) \neq (F_k, G_k)$ $(1 \leq j < k \leq Q)$.

서브태스크 1 (15 점)

- $N \leq 3\ 000$

서브태스크 2 (45 점)

- $Q \leq 100$

서브태스크 3 (40 점)

추가 제한조건이 없다.

예제

standard input	standard output
4 2 0 1 1 0 -1 1 1 2 2 -6 1 2 -2 0 2 0 2 1 2 2 1	1 2

첫 번째 종족간의 대립에서, 다음을 만족한다.

- 1번 용이 3번 용에게 발사한 화염구는 도로와 교차하지 않는다.
- 1번 용이 4번 용에게 발사한 화염구는 도로와 교차하지 않는다.
- 2번 용이 3번 용에게 발사한 화염구는 도로와 교차한다.
- 1번 용이 4번 용에게 발사한 화염구는 도로와 교차하지 않는다.

그러므로 하나의 화염구가 도로를 교차한다.

두 번째 종족간의 대립에서, 다음을 만족한다.

- 3번 용이 1번 용에게 발사한 화염구는 도로와 교차한다.
- 3번 용이 2번 용에게 발사한 화염구는 도로와 교차한다.
- 4번 용이 1번 용에게 발사한 화염구는 도로와 교차하지 않는다.
- 4번 용이 2번 용에게 발사한 화염구는 도로와 교차하지 않는다.

그러므로 두 개의 화염구가 도로를 교차한다.

standard input	standard output
3 2 -1000000000 -1 1 -999999998 -1 1 0 0 2 999999997 1 999999999 1 1 1 2	1
6 3 2 -1 1 1 0 1 0 3 2 2 4 2 5 4 3 3 9 3 0 0 3 3 6 1 2 1 3 2 1 2 3 3 1 3 2	4 2 4 0 2 1