장난감 기차

Arezou와 Borzou는 쌍둥이다. 생일에 장난감 기차 세트를 받아서 그 세트로 n개의 역과 m개의 **단방향** 철로가 있는 철도시스템을 만들었다. 역들은 0번 부터 $n-1$번까지 번호가 붙어 있다. 각 철로는 한 역에서 출발해, 자기 자신이나 다른 역에 도착한다. 각 역에는 그 역에서 출발하는 철로가 최소 하나 있다.

어떤 역들은 **충전역**이다. 충전역에 기차가 도착하면 기차의 배터리가 완전히 충전된다. 완전히 충전된 기차는 n개의 철로를 지날 수 있다. 즉, 완전히 충전된 상태의 기차는 (더 이상의 충전이 없다면) n개의 철로를 지나고 $n+1$번째의 철로에 들어가는 시점에 배터리가 방전되어 멈춘다.

각 역에는 그 역을 출발하는 어떤 철로로든 기차를 보낼 수 있는 스위치가 있다. 기차가 어떤 역에서 출발할 때는 그 역의 스위치가 지정한 방향으로만 떠난다.

쌍둥이는 철로시스템을 가지고 게임을 하려고 한다. 쌍둥이는 모든 역의 소유권을 정했다. 즉, 하나의 역은 Arezou와 Borzou 두 명 중 정확히 한 명이 소유한다. 기차는 한 대를 사용한다. 게임의 시작에 기차는 어떤 역 s에 있고 완전히 충전되어 있다. 게임의 시작에 s의 소유자가 s의 스위치를 세팅하여 기차가 특정한 철로로 출발하도록 한다. 그 다음, 기차의 전원을 켜고 기차가 움직인다.

기차가 어떤 역에 최초로 도착하면, 그 역의 소유자가 역의 스위치를 세팅한다. 역의 스위치를 한번 세팅하면 게임이 진행되는 동안 바뀌지 않는다. 즉, 기차가 동일한 역에 다시 도착한다면 지난번에 출발했던 철로로 다시 출발하게 된다.

역의 개수가 유한하므로, 기차는 어떤 **사이클**에 들어갈 수밖에 없다. 사이클은 **서로 다른** 역 $c[0], c[1], \cdots, c[k-1]$의 순서인데, 모든 $0 \leq i < k-1$에 대해 $c[i]$에서 출발해서 $c[i+1]$에 철로를 통해 도착하며, $c[k-1]$에서 출발해서 $c[0]$에 철로를 통해 도착한다. 어떤 역 $c[0]$에서 기차가 출발해서 자기 자신 $c[0]$에 철로를 통해 도착한다면, 사이클이 단 하나의 역으로 (즉, $k=1$) 구성되는 것도 가능하다.

기차가 무한히 오랫동안 달릴 수 있으면 Arezou가 이기고, 기차가 멈추는 경우 Borzou가 이긴다. 다르게 설명하면, 사이클 $c[0], c[1], \cdots, c[k-1]$에 충전역이 하나라도 있다면 기차가 충전을 반복하며 무한히 달릴 수 있어서 Arezou가 이긴다. 그렇지 않다면 (어쩌면 사이클을 몇 바퀴 돈 후에) 전력이 떨어져서 기차는 멈출 것이고, Borzou가 이긴다.

당신은 철도시스템의 구성을 입력으로 받는다. Arezou와 Borzou는 n번의 게임을 할 것이다. 이들 중 s번째 ($0 \leq s \leq n-1$) 게임에서, 게임의 시작에 기차는 역 s에 있다. 당신은 각 게임에서 (Borzou가 어떤 일을 하더라도) Arezou가 항상 이길 수 있는 방법이 있는지 알아내는 프로그램을 작성해야 한다.

Implementation details

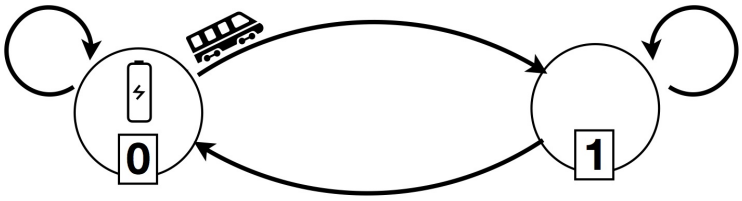
다음 함수를 작성해야 한다.

```
int[] who_wins(int[] a, int[] r, int[] u, int[] v)
```

- a: 크기 n인 배열. Arezou가 역 i를 소유한 경우 $a[i] = 1$. 아니면 Borzou가 역 i를 소유한 것이고 $a[i] = 0$이다.
- r: 크기 n인 배열. 역 i가 충전역인 경우 $r[i] = 1$이고 그렇지 않다면 $r[i] = 0$이다.
- u and v: 크기 m인 배열. 모든 $0 \leq i \leq m - 1$에 대해 $u[i]$에서 출발해서 $v[i]$에 도착하는 철로가 존재한다.
- 이 함수는 크기 n인 배열 w를 리턴한다. 모든 $0 \leq i \leq n - 1$에 대해, 기차가 시작하는 역이 역 i인 경우 (Borzou가 어떻게 하더라도) Arezou가 이긴다면 w[i] = 1 이어야 하고, 아니라면 w[i] = 0 이어야 한다.

Example

```
who_wins([0, 1], [1, 0], [0, 0, 1, 1], [0, 1, 0, 1])
```



- 역이 2개가 있다. Borzou가 역 0의 소유자이다. 역 0은 충전역이다. Arezou가 역 1의 소유자이다. 역 1은 충전역이 아니다.
- 철로는 4개, $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 1)$가 있다. 여기서 (i, j)는 역 i에서 출발해서 역 j에 도착하는 철로가 있다는 뜻이다.
- 기차가 게임의 시작에 역 0에 있는 게임을 생각해 보자. Borzou가 역 0의 스위치를 철로 $(0, 0)$을 따르게 정한다면 기차는 역 0만 포함하는 사이클을 무한히 돌게 된다. (역 0은 충전역이다.) 이 경우 Arezou가 이긴다. 만약, Borzou가 역 0의 스위치를 철로 $(0, 1)$을 따르게 정한다면 기차는 처음에 역 1로 간다. 이때, Arezou가 역 1의 스위치를 철로 $(1, 0)$을 따르게 정하면 기차는 두 역을 포함하는 사이클을 무한히 돌게 된다. 이 경우에도 역 0이 충전역이기 때문에 기차는 멈추지 않고 Arezou가 이긴다. 이 경우, Borzou가 어떤 일을 하더라도 Arezou가 이김을 알 수 있다.
- 기차가 게임의 시작에 역 1에 있는 경우도 비슷한 논리로, Borzou가 어떤 일을 하더라도 Arezou가 항상 이긴다는 것을 알수 있다. 따라서, 함수는 [1, 1]을 리턴해야 한다.

Constraints

- $1 \leq n \leq 5000$.
- $n \leq m \leq 20\,000$.
- 충전역은 최소 1개가 존재한다.
- 각 역에서 출발하는 철로가 적어도 하나는 존재한다.
- 한 역에서 출발해서 그 역에 도착하는 철로가 있을 수 있다 (즉, $u[i] = v[i]$).
- 각 철로는 유일하다. 다르게 말하면, $u[i] = u[j]$, $v[i] = v[j]$가 되는 i, j $(0 \leq i < j \leq m - 1)$는 존재하지 않는다.

- $0 \le u[i], v[i] \le n-1$ (모든 $0 \le i \le m-1$에 대해).

Subtasks

1. (5 points) 모든 $0 \le i \le m-1$에 대해, $v[i] = u[i]$ 혹은 $v[i] = u[i]+1$이다.
2. (10 points) $n \le 15$.
3. (11 points) Arezou가 모든 역을 소유한다.
4. (11 points) Borzou가 모든 역을 소유한다.
5. (12 points) 충전역이 단 하나 존재한다.
6. (51 points) 추가적인 제약이 없다.

Sample grader

Sample grader는 다음의 형식으로 입력을 받는다:

- line 1: n m
- line 2: $a[0]$ $a[1]$ $\ldots$ $a[n-1]$
- line 3: $r[0]$ $r[1]$ $\ldots$ $r[n-1]$
- line $4+i$ (for $0 \le i \le m-1$): $u[i]$ $v[i]$

Sample grader는 `who_wins`의 리턴 값을 다음 형식으로 출력한다:

- line 1: $w[0]$ $w[1]$ $\ldots$ $w[n-1]$