Toy Train玩具火车

Arezou和她的兄弟Borzou是双胞胎。他们收到的生日礼物是一套好玩的玩具火车。他们用它建了一个有n个车站和m段*单向轨道*的铁路系统。这些车站的编号是从0到$n-1$。每段轨道都始于某一车站，然后终于同一车站或其他车站。每个车站至少会有一段轨道以它为起点。

其中有些车站是*充电车站*。无论何时，如果火车抵达某个充电车站，它都会被充到满电。满电火车拥有足够的动力连续地驶过n段轨道，但是如果不再充电的话，在即将进入第$n+1$段轨道时它就会因电已用光而停车。

每个车站都有一个轨道开关，可以扳向任一以该车站为起点的轨道。火车从某个车站驶出时，驶向的正是该车站的开关所扳向的轨道。

这对双胞胎打算用他们的火车玩个游戏。他们已经分完了所有的车站：每个车站要么归Arezou，要么归Borzou。游戏里面只有一列火车。游戏开始时，这列火车停在车站s，并且充满了电。为启动游戏，车站s的拥有者把车站s的开关扳向某个以s为起点的轨道。随后他们启动火车，火车也就开始沿着轨道行驶。

无论何时，在火车首次进入某一车站时，该车站的拥有者都要扳定车站开关。开关一旦扳定，它就会保持状态不变直到游戏结束。因此，火车如果开到了某个曾经进过的车站，就会沿着与之前相同的轨道开出该车站。

由于车站数量是有限的，火车的行驶最终都会落入某个*环路*。环路是指一系列*不同*的车站$c[0], c[1], \cdots, c[k-1]$，其中火车在离开车站$c[i]$ $(0 \leq i < k-1)$后驶上连向车站$c[i+1]$的轨道，在离开车站$c[k-1]$后驶上连向车站$c[0]$的轨道。一个环路可能只包括一个车站(此时$k=1$)，即火车从车站$c[0]$驶出后又驶上了连向$c[0]$的轨道。

如果火车能够连续行驶跑个没完，Arezou就赢了。否则火车最后会把电用光而停车，这样Borzou就赢了。换句话说，如果在车站$c[0], c[1], \cdots, c[k-1]$中至少有一个充电车站，且使得火车能够不断地充电而沿着环路跑个没完，Arezou赢。否则，它就会最终把电用光（有可能是在沿着环路跑好几圈后），Borzou赢。

现在给你一个这样的铁路系统。Arezou和Borzou将会玩n轮游戏。其中在第s轮游戏中（$0 \leq s \leq n-1$），火车最初停在车站s上。你的任务是，对每一轮游戏，判断是否无论Borzou怎么玩，Arezou都必胜。

实现细节

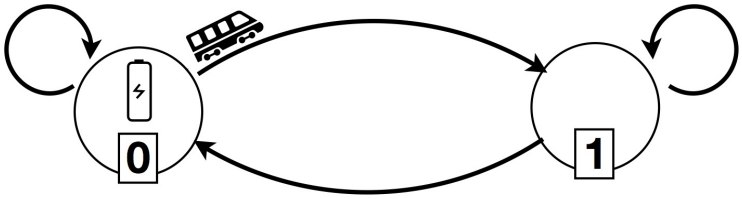
你需要实现下面的函数：

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

- a: 长度为n的数组。如果Arezou拥有车站i，则$a[i] = 1$；否则Borzou拥有车站i，且$a[i] = 0$。
- r: 长度为n的数组。如果车站i是充电车站，则$r[i] = 1$。否则$r[i] = 0$。
- u和v: 长度为m的数组。对于所有$0 \le i \le m - 1$，存在某一单向轨道，其起点为$u[i]$，终点为$v[i]$.
- 该函数需要返回一个长度为n的数组w。对于每个$0 \le i \le n - 1$，如果在火车最初停在车站i的游戏中，不管Borzou怎么玩，Arezou都能赢，则$w[i]$的值应为1。否则$w[i]$的值应为0。

例子

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



- 这里有2个车站。Borzou拥有充电车站0。Arezou拥有车站1，但是它不是充电车站。
- 这里有4段轨道$(0, 0), (0, 1), (1, 0)$和$(1, 1)$，其中(i, j)表示一个以车站i为起点、车站j为终点的单向轨道。
- 考虑火车最初停在车站0的游戏。如果Borzou将车站0的开关扳向轨道$(0, 0)$，那么火车就会沿着这个环形轨道绕个没完（注意，车站0是一个充电车站）。在这种情况下，Arezou赢。否则，如果Borzou把车站0的开关扳向轨道$(0, 1)$，Arezou可以把车站1的开关扳向轨道$(1, 0)$。这样的话，火车将会在两个车站之间绕个不停。Arezou还是会赢，因为车站0是充电车站，火车将跑个没完。因此，无论Borzou怎么玩，Arezou都会赢。
- 根据类似的逻辑，在火车最初停在车站1的游戏中，无论Borzou怎么玩，Arezou也都会赢。因此，函数应当返回$[1, 1]$。

数据范围和限制

- $1 \le n \le 5000$。
- $n \le m \le 20\,000$。
- 至少会有一个充电车站。
- 每个车站至少会有一段轨道以它为起点。
- 可能会有某个轨道的起点和终点是相同的（即$u[i] = v[i]$）。
- 所有轨道两两不同。也就是说，不存在这样的两个下标i和j（$0 \le i < j \le m - 1$），使得$u[i] = u[j]$且$v[i] = v[j]$。
- 对于所有$0 \le i \le m - 1$，都有$0 \le u[i], v[i] \le n - 1$。

子任务

1. (5分) 对于所有$0 \leq i \leq m-1$，都有$v[i] = u[i]$或者$v[i] = u[i] + 1$。
2. (10分) $n \leq 15$。
3. (11分) Arezou拥有所有车站。
4. (11分) Borzou拥有所有车站。
5. (12分) 充电车站的数量为1。
6. (51分) 无任何限制。

评分程序样例

评分程序样例会按照下述格式来读取输入数据：

- 第1行： n m
- 第2行： $a[0]$ $a[1]$ $\ldots$ $a[n-1]$
- 第3行： $r[0]$ $r[1]$ $\ldots$ $r[n-1]$
- 第$4+i$行(对于所有$0 \leq i \leq m-1$)： $u[i]$ $v[i]$

评分程序样例会按照下述格式打印出`who_wins`的返回结果：

- 第1行： $w[0]$ $w[1]$ $\ldots$ $w[n-1]$