ETH Zurich
Institut für Theoretische Informatik
Prof. Dr. Angelika Steger
Prof. Dr. Emo Welzl
Prof. Dr. Peter Widmayer

Algorithms Lab

Exercise – *The Great Game*

Smart as they are—and gentlemen along the way—Sherlock Holmes and Professor Moriarty have decided to settle the scores by playing The Great Game.

The Great Game is played on a board with n positions, labeled from 1 to n. Some positions are connected via a transition. Every transition goes from a position with a lower label to a position with a higher label. The position labeled n is called *target* position.

Holmes and Moriarty alternate in making moves. There are *two meeples* on the board, red and black. Each move consists of moving one meeple along exactly one transition. If Holmes has played an even number of times so far, then in his next turn he moves the red meeple; otherwise, he moves the black meeple. Similarly, if Moriarty has played an even number of times so far, then in his next turn he moves the black meeple; otherwise, he moves the red meeple (see the example described on the next page). Holmes has the first move (thus moving the red meeple). The game ends as soon as one of the meeples reaches the target position. If the red meeple reaches the target position, then Holmes wins; if it is the black meeple, then Moriarty wins. You may assume that the target position is reachable from every position via a sequence of transitions. It follows that the game always ends with either Moriarty's or Sherlock's win.

Given the board and the starting positions of the two meeples, your job is to determine who has a winning strategy. A player has a *winning strategy*, if he can win no matter how the other player plays. As the game cannot end in a draw, exactly one player has a winning strategy.

Remark: The two meeples are allowed to be at the same position.

Input The first line of the input contains the number $1 \leq t \leq 200$ of test cases. Each of the t test cases is described as follows.

- It starts with a line that contains two integers `n m`, separated by a space and such that $2 \leq n \leq 5 \cdot 10^4$ and $1 \leq m \leq 5 \cdot 10^4$. Here n denotes the number of positions on the table and m denotes the number of transitions.
- The following line contains two integers `r b`, separated by a space and such that $1 \leq r, b \leq n-1$. Here r denotes the starting position of the red meeple and b denotes the starting position of the black meeple.
- The following m lines define the m transitions. Each transition is defined by two integers `u v`, separated by a space and such that $1 \leq u < v \leq n$, which indicates that there exists a transition from position u to position v.

Between every pair of positions there is at most one transition.

Output Output a line with one character "0" or "1" per test case. For each test case output "0", if Sherlock has a winning strategy, and "1", if Moriarty has a winning strategy.

Points We say that a sequence of positions $p_1, p_2, \ldots, p_{k-1}, p_k$ is a *path* from p_1 to p_k if there is a transition from p_i to p_{i+1}, for every $i \in \{1, \ldots, k-1\}$.

There are four test sets, worth 100 points in total.

1. For the first group of test sets, worth 25 points, you may assume $2 \leq n \leq 50$ and $1 \leq m \leq 50$. Moreover, there is a *unique path* from r to n and *at most two* different paths from b to n.
2. For the second group of test sets, worth 25 points, you may assume $2 \leq n \leq 100$ and $1 \leq m \leq 100$. Moreover, there is a *unique path* from r to n and *at most six* different paths from b to n.
3. For the third group of test sets, worth 25 points, you may assume $2 \leq n \leq 2 \cdot 10^3$ and $1 \leq m \leq 2 \cdot 10^3$. Moreover, there is a *unique path* from r to n and no additional assumptions on paths from b to n.
4. For the forth group of test sets, worth 25 points, there are no additional assumptions.

Remark: two different paths between positions u and v can intersect at positions beside u and v.

Corresponding sample test sets are contained in `test`i`.in/out`, for $i \in \{1, 2, 3, 4\}$.

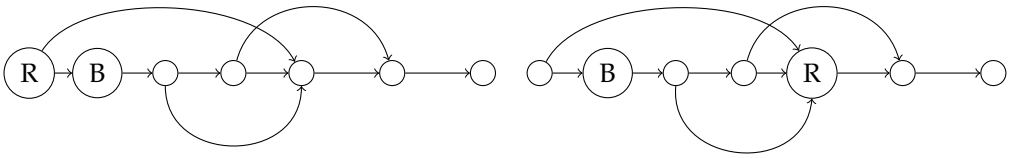
Hint: Put `std::ios_base::sync_with_stdio(false)` as the first line of the main procedure for faster I/O.

Sample Input

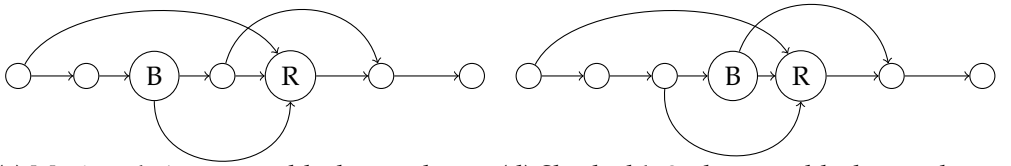
```
2
3 2
1 1
1 2
2 3
7 9
1 2
1 2
2 3
3 4
4 5
5 6
6 7
1 5
3 5
4 6
```

Sample Output

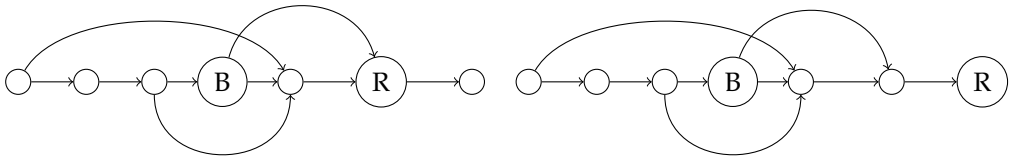
```
1
0
```



(a) Initial position. (b) Sherlock's 1st move - red meeple.



(c) Moriarty's 1st move - black meeple. (d) Sherlock's 2nd move - black meeple.



(e) Moriarty's 2nd move - red meeple. (f) Sherlock's 3rd move - red meeple.

Figure 1: Example game for the second test case.