ODDEMNITY

After a short detour, you're back to working hard in your new office (only 119 roads from your home!). The three of you eventually come up the idea sure to become the next Facebook: the Enemy-omatic.

Everyone has had the experience of disliking someone, but not having a good reason for it. Maybe their voice irritates you, or maybe you can't un-notice their unmatched socks, but regardless in no world can you articulate this type of reasoning to others. The Enemy-omatic solves this problem - giving one the perfect excuse for explaining away their aversion. If, for instance, you are A and dislike B, you could explain this by saying you're enemies with C, D is enemies with C and thus your friend, and B is enemies with D. With this flawless reasoning, nobody would give a second thought.

The app knows about N people, and specifically that M pairs of these people are **direct enemies** (being direct enemies is **symmetric**). Then, if a person A needs to explain why B is their enemy, it suffices to repeatedly apply the rules

- The *direct enemy* of my enemy is my friend.
- The *direct enemy* of my friend is my enemy.

This means that A can explain why B is their enemy if they find a list of k people, $p_1, \cdots, p_k$ such that $p_1 =$ A, $p_k =$ B, such that for all $1 \leq i < k$ the pair p_i, p_{i+1} are *direct enemies*, and k **is even**. Note in particular that the people in the list **do not have to be distinct**, so essentially you must determine whether one can go along an odd number of direct enemy relationships from A to B.

You must write the server for this app, and process Q queries in order, the i-th of which operates on two vertices, a_i and b_i. Type 1 queries represent adding a new *direct enemy* relationship between a_i and b_i. Type 2 queries represent a request for enmity excuse, and you must determine whether there exists a list of people, as above, explaining why a_i can consider b_i an enemy.

Input

The first line contains three space-separated integers, N, M, and Q.

The next M lines contain the initial pairs of enemies. The $(i+1)$-th ($1 \leq i \leq M$) line contains two space-separated integers p_i and q_i, which means p_i and q_i are direct enemies. It is guaranteed that $1 \leq p_i, q_i \leq N$.

The next Q lines contain the queries. The i-th will contain three space-separated integers, t_i, a_i, b_i, where t_i is the type of the query, and a_i, b_i are the relevant people. It is guaranteed that $t_i \in \{1, 2\}$, and $1 \leq a_i, b_i \leq N$.

Output

Let the answer to each query be a 1 if an explanation exists, and 0 otherwise. Output the sum of the answers to all queries.

Constraints

In all test cases, $1 \leq N, M, Q \leq 2 \cdot 10^5$. Beyond the sample input, the tests are divided into batches with additional constraints. Time limits below are for C/C++; Ocaml gets 2x, Java 3x, and Python 10x.

- 10 points worth satisfy that the initial direct enemy pairs will be $(i, i+1)$ for $1 \leq i < N$, and all queries will be of type 2. TL: 200ms.
- 30 points worth satisfy $N, M, Q \leq 1000$. TL: 200ms.
- 24 points worth satisfy all queries will be of type 2. TL: 300ms.
- 36 points worth satisfy no further constraints. TL: 500ms.

Sample explanation

At the first query of type 2, 1 and 2 are already direct enemies, so the answer is yes. For the second, 2 and 3 only have the odd-length list $(2, 1, 3)$ which doesn't explain enmity. However, after adding the direct enemy pair $(2, 3)$, the list $(2, 3)$ works! Note additionally that after all queries, the query $2, 1, 1$ somewhat peculiarly would have answer yes, with list $(1, 2, 3, 1)$.

View submissions (https://cs124.seas.harvard.edu/problem/ODDEMNITY/code-submission)

Test cases

Input	Output	Points	Timeout
3 1 5 1 2 2 1 2 1 1 3	2	0	100 ms
Hidden	*Hidden*	5	350 ms
Hidden	*Hidden*	5	350 ms
Hidden	*Hidden*	10	200 ms
Hidden	*Hidden*	10	200 ms
Hidden	*Hidden*	10	200 ms
Hidden	*Hidden*	8	500 ms
Hidden	*Hidden*	8	500 ms
Hidden	*Hidden*	8	500 ms
Hidden	*Hidden*	9	600 ms
Hidden	*Hidden*	9	600 ms
Hidden	*Hidden*	9	600 ms
Hidden	*Hidden*	9	600 ms

Download (https://cs124.seas.harvard.edu/problem/ODDEMNITY/test-cases)

Inspired by the "Ultra Cool Programming Contest Control Centre" by Sonny Chan.
Modified for CS 124 by Neal Wu (https://github.com/nealwu), with design help from Martin Camacho.
Further refined by Nikhil Benesch (https://github.com/benesch).