G. Shortest Path Queries

You are given an undirected connected graph with weighted edges. The length of some path between two vertices is the bitwise xor of weights of all edges belonging to this path (if some edge is traversed more than once, then it is included in bitwise xor the same number of times).

There are three types of queries you have to process:

- $1\ x\ y\ d$ — add an edge connecting vertex x to vertex y with weight d. It is guaranteed that there is no edge connecting x to y before this query;
- $2\ x\ y$ — remove an edge connecting vertex x to vertex y. It is guaranteed that there was such edge in the graph, and the graph stays connected after this query;
- $3\ x\ y$ — calculate the length of the shortest path (possibly non-simple) from vertex x to vertex y.

Print the answers for all queries of type 3.

Input

The first line contains two numbers n and m $(1 \leq n, m \leq 200000)$ — the number of vertices and the number of edges in the graph, respectively.

Then m lines follow denoting the edges of the graph. Each line contains three integers x, y and d $(1 \leq x < y \leq n, 0 \leq d \leq 2^{30} - 1)$. Each pair (x, y) is listed at most once. The initial graph is connected.

Then one line follows, containing an integer q $(1 \leq q \leq 200000)$ — the number of queries you have to process.

Then q lines follow, denoting queries in the following form:

- $1\ x\ y\ d$ $(1 \leq x < y \leq n, 0 \leq d \leq 2^{30} - 1)$ — add an edge connecting vertex x to vertex y with weight d. It is guaranteed that there is no edge connecting x to y before this query;
- $2\ x\ y$ $(1 \leq x < y \leq n)$ — remove an edge connecting vertex x to vertex y. It is guaranteed that there was such edge in the graph, and the graph stays connected after this query;
- $3\ x\ y$ $(1 \leq x < y \leq n)$ — calculate the length of the shortest path (possibly non-simple) from vertex x to vertex y.

It is guaranteed that at least one query has type 3.

Output

Print the answers for all queries of type 3 in the order they appear in input.

Example

input

```
5 5
1 2 3
2 3 4
3 4 5
4 5 6
1 5 1
5
3 1 5
1 1 3 1
3 1 5
2 1 5
3 1 5
```

```
output
1
1
2
```