XOR(emember)

You have a DB which stores at most I entries. An entry is defined to be a pair of *key* and *value*. The DB stores the entries sorted in an ascending order of the key. The key can be any natural number and the value is an unsigned integer of 16 bits. Initially the DB is empty

The input to your program is a sequence of two kinds of commands. One kind is the INSERT command. The argument to an INSERT command is a pair of key and value. The meaning of an INSERT command is to add an entry. In case the key already exists in the DB, then the meaning of the command is to update the value of the affected entry.

The other kind is the QUERY command. The argument to a QUERY command is the triple **s, i, j**. The result of the QUERY command must be the XOR result of the values whose keys are between **i** and **j**, inclusive, *at the time when the s-th insert command has been performed.* One can see that the result of the QUERY command when no relevant entries exist, is 0.

[Input]

The number of cases T is given in the first line of the input file. Here, T ≤ 30. In the first line of each case, the maximum of key **N**, the number of INSERT commands **I**, and the number of QUERY commands **Q** are given. (1 ≤ **N, I, Q** ≤ 500,000) In the next **I** lines, an INSERT command is given per line. An INSERT command is composed of key k and value x. (1 ≤ k ≤ N, 1 ≤ x ≤ 2^{16}) In the next **Q** lines, a QUERY command is given per line. A QUERY command is composed of time **s**, lower bound **i**, and upper bound **j.** (1 ≤ s ≤ I, 1 ≤ i ≤ j ≤ N)

NOTE: For the first QUERY command, the arguments **s, i,** and **j** are given as actual values. However, from the second QUERY command and on, the actual values of **s**, **i**, and **j** must be obtained by XOR'ing the given values with the result of the previous command after storing each of them in a 32-bit unsigned integer variable.

The input is given from the following two sets.

- Set 1: 1 ≤ **N, I, Q** ≤ 5,000
- Set 2: 1 ≤ **N, I, Q** ≤ 500,000

[Output]
In the only line, print the result of the final query.

[I/O example]

Input

```
3
2 3 3
2 5
```

```
1 4
1 3
3 1 2                         <- result is 6
7 7 7                         <- after XOR'ing with 6, becomes 1 1 1, result is 0
2 1 2                         <- after XOR'ing with 0, becomes 2 1 2, result is 1
3 3 3
1 5
2 8
3 9
3 1 3
6 7 7
1 1 2
4 2 2
2 1
3 4
1 1 2
0 0 0
```

Output

```
1
5
0
```