Square dance

You are hired by french NSA to break the RSA code used on the Pink Card. The easiest way to do that is to factor the public modulus and you have found the fastest algorithm to do that, except that you have to solve a subproblem that can be modeled in the following way.
Let $\mathcal{P} = \{p_1, p_2, ..., p_n\}$be a set of prime numbers. If $S = \{s_1, s_2, ..., s_u\}$ and $T = \{t_1, ..., t_v\}$ are formed with elements of $\mathcal{P}$, then S*T will denote the quantity

$$s_1 * s_2 * \cdots * s_u * t_1 * t_2 * \cdots * t_v.$$

We call relation a set of two primes p,q, where p and q are distinct elements of $\mathcal{P}$. You dispose of a collection of R relations $S_i = \{p_i, q_i\}$ and you are interested in finding sequences of these, $S_{i_1}, S_{i_2}, ..., S_{i_k}$ such that

$$S_{i_1} * S_{i_2} * \cdots * S_{i_k}$$

is a perfect square.

The way you look for these squares is the following. The ultimate goal is to count squares that appear in the process. Relations arrive one at a time. You maintain a collection $\mathcal{C}$ of relations that do not contain any square subproduct. This is easy: at first, $\mathcal{C}$ is empty. Then a relation arrives and $\mathcal{C}$ begins to grow. Suppose a new relation $\{p, q\}$ arrives. If no square appears when adding $\{p, q\}$ to $\mathcal{C}$, then $\{p, q\}$ is added to the collection. Otherwise, a square is about to appear, we increase the number of squares, but we do not store this relation, hence $\mathcal{C}$ keeps the desired property.
Let us consider an example. First arrives $S_1 = \{2, 3\}$ and we put it in $\mathcal{C}$; then arrives $S_2 = \{5, 11\}, S_3 = \{3, 7\}$ and they are stored in $\mathcal{C}$. Now enters the relation $S_4 = \{2, 7\}$. This relation could be used to form the square:

$$S_1 * S_3 * S_4 = (2 * 3) * (3 * 7) * (2 * 7) = (2 * 3 * 7)^2.$$

So we count 1 and do not store S_4 in $\mathcal{C}$. Now we consider $S_5 = \{5, 11\}$ that could make a square with S_2, so we count 1 square more. Then $S_6 = \{2, 13\}$ is put into $\mathcal{C}$. Now $S_7 = \{7, 13\}$ could make the square $S_1 * S_3 * S_6 * S_7$. Eventually, we get 3 squares.

Input

The first line of the input contains a number T <= 30 that indicates the number of test cases to follow. Each test case begins with a line containing two integers P and R: $P \leq 10^5$ is the number of primes occurring in the test case; R ($\leq 10^5$) is the number of sets of primes that arrive. The subsequent R lines each contain two integers i and j making a set $\{p_i, q_i\} (1 \leq i, j \leq P)$. Note that we actually do not deal with the primes, they are irrelevant to the solution.

Output

For each test case, output the number of squares that can be formed using the preceding rules.

Example

Input:
```
2
6 7
1 2
3 5
2 4
1 4
3 5
1 6
4 6
2 3
1 2
1 2
1 2
```

Output:
```
3
2
```

Warning: large Input/Output data, be careful with certain languages