**Simple Path**

Time Limit: 4000/2000 MS (Java/Others)    Memory Limit: 65536/65536 K (Java/Others)

**Problem Description**

Everybody knows that totalfrank has absolutely no sense of direction. Getting lost in the university or nearly supermarket is very common for him. We always worry about whether he can find his way back into our sweet base whenever he goes out alone for his class. In general, if totalfrank get lost again, we need to check his starting point and destination just in order to find out where he could be (you know this task is very common for us).

Unfortunately, poor totalfrank sometimes forgot taking his mobile phone, when this situation happens, we can’t get in touch with him. But it is so lucky that totalfrank can remember places where he had gone before in his trip from his starting point and destination at this trip so that he won’t go to such place again (he can’t remember places which he had gone during his previous trip). As we are all familiar with map, we can find out which place he couldn’t be.

However, totalfrank can always get lost, doing this same boring work makes us sleepy. So we ask totalfrank for all possible starting point and destination for him, and try to find out how many places he wouldn’t be when he chooses any pair of starting point and destination. Here comes the problem, since our university’s map is so complex (there can be many buildings which can be considered as points in our university, some pair of these point has a way while others hasn’t), we need a program to help us work out this problem.

**Input**

The input contains no more than 10 test cases.

For each test case:

The first line includes two integers N (1 <= N <= 100000) and M (0 <= M <= 200000). N is the total number of nodes. M is the number of edges.

The next M lines each describe an edge of this graph in the following format:

X (0 <= X < N) Y (0 <= Y < N)

It means that there is an edge from point X to point Y. Ways are bidirectional and there are no duplicates or self-loop edges.

The next line includes only one integer Q (1 <= Q <= 100000) which indicates the total number of queries.

The next Q lines are all in the following format:

A B, which means in this query totalfrank choose A as his starting point and B as his destination.

**Output**

For each test case, first you should print “Case #x:” in a line, where x stands for the case number started with 1. Then for each query output a line contains a single integer indicating the number of places which he absolutely couldn’t be during the simple path between his starting point and destination. If there is no simple path between the starting point and destination just output N.

Leave an empty line after each test case.

**Sample Input**

4 3

0 1

1 2

1 3

1

0 2

4 4

0 1

1 2

1 3

2 3

1

0 2

**Sample Output**

Case #1:

1

Case #2:

0