Lazy Bob and Lime Bike

Filename: `limebike`

Lazy Bob is back and he is lazier than ever. In fact, the only time he is not lazy is when he is solving flow problems. One day, Lazy Bob was heading to class on campus but was running extremely late. He did what any UCF student would do and hopped on the nearest Lime Bike. And just like most other Lime Bike users, when he got to class he left his bicycle in a very inconvenient place: blocking an entire walkway!

UCF's campus can be modeled as a series of nodes and edges, with the buildings being the nodes and walkways (or bikeways) between them being edges.

The Problem:

Given the edge Lazy Bob blocks, determine how many unordered pairs of buildings are no longer connected (if they were before).

The Input:

The first line of input is a positive integer t, the number of days where Lazy Bob took a lime bike to class.

For each day the input is as follows. The first line has two integers $1 \leq n, m \leq 10^5$, the number of buildings and walkways, respectively. Then m lines follow, each with two positive integers $1 \leq a, b \leq n$, $a \neq b$, indicating that there is a walkway between buildings a and b. One line follows with a single integer $1 \leq e \leq m$, the index of the walkway Lazy Bob blocks. Walkways are indexed by the order they appear in the input (starting from ***1*** in each case).

The Output:

For each test case, output a single line saying "`Day #i: c`" without the quotes, where `i` is the test case number, and `c` is the number of unordered pairs of buildings such that one could reach one from the other before Lazy Bob left his Lime Bike but not after.

(Sample Input and Output are on the next page)

Sample Input:

```
2
3 3
1 2
2 3
3 1
1
4 3
1 2
1 3
3 4
2
```

Sample Output:

```
Day #1: 0
Day #2: 4
```