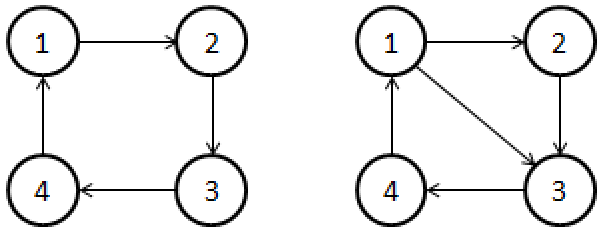
G	Graph Guessing **Input:** Standard Input **Output:** Standard Output	

Graph Guessing

There is a strongly-connected graph (i.e. you can reach any node from any other node) with n nodes and m edges. I will choose some of the edges to make another strongly connected graph. Your task is to guess that graph. Too difficult, right? Don't worry, you only need to guess k edges. If all the edges exist in my graph, you win. I promise that from all possible graphs, the answer will be chosen uniformly. **The original graph will not have self-loops or duplicated edges**.

You already have a guess, but you are a bit unsure. Why not write a program to calculate the probability you win? For example, if n=4, m=5, the original graph has 5 edges: 1->2, 2->3, 3->4, 4->1, 1->3, there are only two possible answers:



If k=2, the best way is to guess edge 1->2 and 2->3 (or 1->2 and 3->4 etc.) which will guarantee a win. But if you would like to risk by guessing edges 1->3 and 2->3, the probability you win is 0.5.

Input

There will be at most 10 test cases. Each case begins with two integers n, m (3<=n<=15, 2<=m<=50). Each of the following m lines contains two different integers u, v (1<=u,v<=n), that means u->v is in the original graph. Edges are numbered 1 to m in the same order they appear in the input. The last line begins with an integer k (1<=k<=m) and k different integers, the edges you guess.

Output

For each test case, print the case number and the probability you win. Absolute error of 10^{-4} is allowed.

Sample Input

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

Output for Sample Input

```
Case 1: 1.0000
Case 2: 0.5000
```

Problemsetter: Rujia Liu, Special Thanks: Md. Mahbubul Hasan, Shiplu Hawlader
Based on a problem in Chinese IOI Training camp, by Lijie Chen