K-Walk Count

Submit solution

All submissions
Best submissions

Points: 100 (partial)
Time limit: 1.0s
Memory limit: 256M

Allowed languages
C, C++

You are given an undirected graph with n nodes and m edges. Each edge is provided on a separate line as a pair of integers i and j (1-indexed). Your task is to output the number of k-length walks between every pair of nodes i and j modulo M. Note that M can be very large. ($M = 10^9 + 7$)

A k**-length walk** is defined as a sequence of $k + 1$ nodes \((v_0, v_1, ..., v_k)\) such that for every $0 \leq i < k$, there exists an edge between \(v_i\) and \(v_{i+1}\) in the graph. Unlike simple paths, walks are allowed to revisit the same node or edge multiple times.

The problem consists of three subparts based on the value of k:

- **Subpart 1 (40 points):** $k = 2$.
- **Subpart 2 (40 points):** $k \leq 30$.
- **Subpart 3 (20 points):** $k \leq 10^9$.

Input

The first line contains four integers separated by spaces:

- n ($1 \leq n \leq 100$): the number of nodes,
- m: the number of edges, ($0 \leq m \leq \frac{n*(n-1)}{2}$)
- k: the length of walks,

The next m lines each contain two integers i and j indicating there is an undirected edge between node i and node j.

Output

Each row should be printed on a new line with elements separated by spaces.

Example 1

Input

Copy

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

Output

Copy

```
2 0 2 0
0 2 0 2
2 0 2 0
0 2 0 2
```

Explanation

A **2-length walk** consists of three nodes u, v, w such that there is an edge between u and v, and an edge between v and w. To count the number of 2-length walks from a starting node to an ending node, you simply count all possible intermediate nodes that allow you to travel from the start to the finish in exactly two steps.

For example, consider the sample graph with nodes 1, 2, 3, and 4 and these undirected edges:

- 1 is connected to 2 and 4.
- 2 is connected to 1 and 3.
- 3 is connected to 2 and 4.
- 4 is connected to 1 and 3.

Let's explain the counts using this idea:

- **From node 1 to node 1:**
  The valid 2-length walks are:
  1 → 2 → 1 (using node 2 as the intermediate)
  1 → 4 → 1 (using node 4 as the intermediate)
  This gives a total of 2 walks.

- **From node 1 to node 2:**
  To form a 2-length walk from 1 to 2, you need an intermediate node x such that there is an edge from 1 to x and from x to 2.

Thus, there are 0 valid walks from 1 to 2.

- **From node 1 to node 3:**
  The valid 2-length walks are:
  1 → 2 → 3 (using node 2 as the intermediate)
  1 → 4 → 3 (using node 4 as the intermediate)
  This gives a total of 2 walks.

- **From node 1 to node 4:**
  To form a 2-length walk from 1 to 4:
  - If $x = 2$, then the walk would be 1 → 2 → 4, but node 2 is not connected to node 4.
  - If $x = 4$, then the walk would be 1 → 4 → 4, but node 4 does not have a self-loop.
    Thus, there are 0 valid walks from 1 to 4.

Repeating similar reasoning for walks starting at nodes 2, 3, and 4, you arrive at the final output matrix:

Clarifications

Request clarification

No clarifications have been made at this time.