Coding Competitions Farewell Rounds - Round D

Ring-Preserving Networks

Problem

A research consortium is building a new datacenter. In the datacenter, a set of computers is set up to work together and communicate via a network. The network works only with direct bidirectional links between computers. After the success of their name-preserving networks, they decided to do other designs with guaranteed properties.

The consortium has asked you to submit a design of a ring-preserving network. We define a *ring* of a network as an ordering of all the computers of the network such that any two computers that are consecutive in the ordering have a direct link between them, and the same is true for the first and last computers in the ordering.

A ring-preserving network is a network design that can efficiently find a ring of itself even after losing the original computer identifications. You need to submit several network designs that are ring-preserving.

To evaluate your network designs, the research consortium has set up an automated program. You will be asked for network designs specifying the exact number of computers $\mathbf{C}$ and the exact number of bidirectional links $\mathbf{L}$ it must contain. You must assign each computer a unique ID between 1 and $\mathbf{C}$ and list the $\mathbf{L}$ links using the IDs to refer to the links' endpoints. The evaluating program will receive that design and send back a copy of the network design with the following changes:

- the unique IDs have been permuted uniformly at random (that is, each ID is now equally likely to be on any of the computers),
- every link is listed with the smallest ID first (using the new IDs), and
- the set of links is listed in increasing order of the first endpoint (using the new IDs), breaking ties by smallest second endpoint (i.e., lexicographical order).

You need to be able to find a ring of the modified network. It does not need to be the original ring.

Input and output

This is an interactive problem. You should make sure you have read the information in the Interactive Problems section of our FAQ.

Initially, your program should read a single line containing an integer, $\mathbf{T}$, the number of test cases. Then, $\mathbf{T}$ test cases must be processed.

For each test case, your program must first read a line containing two integers $\mathbf{C}$ and $\mathbf{L}$: the number of computers and links to include in the network design.

Then, you need to create a network design with $\mathbf{C}$ computers and $\mathbf{L}$ links and print exactly $\mathbf{L}$ lines representing that design. Each of these lines must contain two integers A and B each, representing a different link between computers A and B, where $A \neq B$. Notice that if you list link A B, you may not list A B nor B A again.

Upon reading your network design, the judge will send you $\mathbf{L}$ lines back representing the permuted design. The i-th of these lines contains two integers $\mathbf{U_i}$ and $\mathbf{V_i}$ representing a bidirectional link between the computers with new ids $\mathbf{U_i}$ and $\mathbf{V_i}$. The copy is generated using a permutation chosen uniformly at random from all possible permutations, and independently of any other choices.

To finish a test case, you need to send the judge a single line with $\mathbf{C}$ integers $x_1, x_2, \ldots, x_{\mathbf{C}}$, representing a ring of the permuted design. That is, the set of lines the judge sent back must include either $x_1\ x_{\mathbf{C}}$ or $x_{\mathbf{C}}\ x_1$, and, for all i, it must include either $x_i\ x_{i+1}$ or $x_{i+1}\ x_i$.

You should not send additional information to the judge after solving all test cases. In other words, if your program keeps printing to standard output after printing the list of xs for the last test case, you will receive a Wrong Answer judgment.

If at any point the judge reads from your program malformed input (wrong number of tokens, non-integer tokens or out of range numbers) it will immediately stop and assume Wrong Answer. However, if your program happens to remain waiting for input from the judge, it may end up exceeding the time limit and receiving a Time Limit Exceeded judgement. On the other hand, if you commit a recoverable error (sending over a network with a repeated connection, or a connection from a computer to itself, or sending a ring that repeats a computer or that uses an edge that does not exist in the permuted version) the judge will continue to communicate with your program trying to finish, but the overall judgement will be Wrong Answer.

Notice that you are allowed to submit the same network design for different test cases, as long as that design complies with all restrictions for both cases. Additionally, the seed for random generation in the judge is fixed, so sending the same set of original network designs in the same order will get back the same set of copies.

Limits

Time limit: 30 seconds.
Memory limit: 2 GB.
$1 \leq \mathbf{T} \leq 100$.
$3 \leq \mathbf{C} \leq 10000$.
$\mathbf{C} \leq \mathbf{L} \leq \mathbf{C} \times (\mathbf{C}-1)/2$.
$1 \leq \mathbf{U_i} < \mathbf{V_i} \leq \mathbf{C}$, for all i.
$\mathbf{U_i} \leq \mathbf{U_{i+1}}$, for all i.
If $\mathbf{U_i} = \mathbf{U_{i+1}}$ then $\mathbf{V_i} \leq \mathbf{V_{i+1}}$, for all i.
There exist permutations f of length $\mathbf{C}$ and g of length $\mathbf{L}$ such that, for each i, if $A\ B$ is the $g(i)$-th line in your original design, then $\{\mathbf{U_i}, \mathbf{V_i}\} = \{f(A), f(B)\}$. (The given links result from applying a permutation of computer IDs to the ones you gave, and then sorting the links lexicographically).

Test Set 1 (Visible Verdict)

$\mathbf{L} \leq \mathbf{C} + 10$.

Test Set 2 (Hidden Verdict)

$\mathbf{L} \leq 20000$.

Testing Tool

You can use this testing tool to test locally or on our platform. To test locally, you will need to run the tool in parallel with your code; you can use our interactive runner for that. For more information, read the instructions in comments in that file, and also check out the Interactive Problems section of the FAQ.

Instructions for the testing tool are included in comments within the tool. We encourage you to add your own test cases. Please be advised that although the testing tool is intended to simulate the judging system, it is **NOT** the real judging system and might behave differently. If your code passes the testing tool but fails the real judge, please check the Coding section of the FAQ to make sure that you are using the same compiler as us.

Download testing tool

Sample Interaction

Judge **Solution**

Number of cases $\mathbf{T} = 2$.

```
2
```

Case 1

```
4 5
```

$\mathbf{C} = 4, \mathbf{L} = 5.$

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

User gives 5 connections.

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

Judge happens to apply the permutation that sends each number to itself, but it still gives the connections back sorted.

```
1 2 4 2
```

User gives a list of the right size, but not a ring. Because this is a recoverable error, the judging continues.

Case 2.

```
4 4
```

New case with $\mathbf{C} = \mathbf{L} = 4$.

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

User gives a cycle.

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

The judge maintains the cycle, but changes the ids.

```
2 1 3 4
```

The user correctly returns a ring using the changed ids.

Case 2 is correct, but Case 1 is not, so the final judgement is Wrong Answer.