ESO207

Data Structures and Algorithms

Indian Institute of Technology, Kanpur

Date of Submission: March 19, 2024

Programming Assignment

3

General Instructions.

- Solution of Programming Assignments(PA) are accepted on Gradescope. Individual submissions via emails will not be graded.
- Gradescope will accept solutions only in the following programming languages - C and C++.
- The autograding process will evaluate your assignment against a predefined set of *visible* test cases immediately upon submission.
- Following the deadline, the final version of your submitted code will undergo autograding with a set of *hidden* test cases.
- Use of algorithm header file like `<algorithm.h>` and primitive functions for operations like sorting, searching and implementing data structures constitutes plagiarism. This course is on Data Structure and Algorithm; hence, we expect students to write their code rather than using predefined function.
- Use of generative tools like ChatGPT is unacceptable and will lead to heavy penalties.

Problem 1: Bad Roads

Guidelines

- Please submit your single file solutions via Gradescope, ensuring that the filename is either `badroads.c` or `badroads.cpp`. The solution should consist of a single file and not multiple files.
- Number of test cases (visible and hidden): 14

Statement

Your friend is coming to campus, and you want to create a map for them. Your campus can be visualized as a **complete graph** G with n vertices. Each vertex denotes a location of interest, while the edges between vertices denote roads. Unfortunately, some of the roads are bad. There are exactly m bad roads.

You want to create a map but do not want to clutter it. So you draw a map containing all n vertices and $n - 1$ roads so that the resultant graph is still connected. Since you are a good friend, you do not want your buddy to walk on bad roads. So, you want to create a map with the minimum number of bad roads. Calculate the number of bad roads in a map with the minimum number of bad roads. Note that you need to output the minimum number of bad roads and not the entire map.

Input Format

The first line will denote n, m, the number of locations of interest and the number of bad roads. The following m lines contains two integers a and b such that $1 \leq a, b \leq n$ and $a \neq b$. Each line denotes a bad road between location a and b. It is guaranteed that each edge appears exactly once in the input.

Output Format

Output the minimum number of bad roads possible in a map.

Constraints

- $1 \leq n \leq 10^5$
- $0 \leq m \leq \min\left(\frac{n(n-1)}{2}, 10^5\right)$

Example

Input

```
6 11
1 3
1 4
1 5
1 6
2 3
2 4
2 5
2 6
3 4
3 5
3 6
```

Output

```
2
```

Explanation

The map with the following edges is optimal:

- (1, 2)
- (1, 6)
- (3, 6)
- (4, 6)
- (5, 6)

The bad edges are $(1, 6)$ and $(3, 6)$.

Problem 2: Techkriti

Guidelines

- Please submit your single file solutions via Gradescope, ensuring that the filename is either `techkriti.c` or `techkriti.cpp`. The solution should consist of a single file and not multiple files.
- Number of test cases (visible and hidden): 15

Statement

You are a part of Techkriti's organizing team and have received information about the initial scheduling of the events. One event is scheduled everyday, each with a distinct *vibe*. For the given schedule, an event can be bad under two scenarios—First, if some event with more vibe has been scheduled before that event. Second, if an event with lower vibe has been scheduled after that event. Your objective is to select a bad day for your assignment under the first scenario.

You can request to change the order of events. With each request, you can swap the dates of two events. You want to keep a day to do the assignment while keeping the bad events to a minimum. In particular, you want to keep the number of bad events to be exactly two. For each query answer, what is the minimum number of requests required to achieve the objective?

Input Format

The first line contains the number of testcases T. For each testcase:

- The first line contains a positive integer N, the number of events.
- Next line contains N numbers that give the vibes v_i ($1 \le i \le N$) of the events in the order they are initially scheduled.

Output Format

Output T lines, each with the minimum number of requests you have to make in the corresponding testcase to fulfill the objective.

Constraints

- $N_i > 1$
- $2 \le \Sigma_{i=1}^{T} N_i \le 2 \cdot 10^5$, where N_i denotes number of events in the i-th test case.
- $1 \le v_i \le N$

Example

Input

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

Output

```
1
0
```

Explanation

Valid schedulings are $1, 3, 2$ and $2, 1, 3$. Both require one request for test 1. Test 2 is already scheduled optimally.