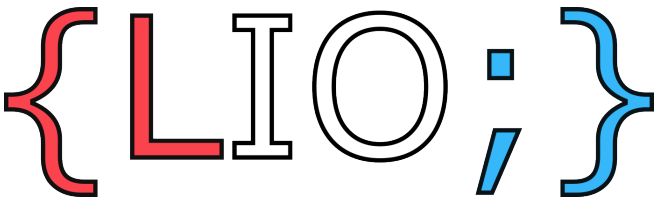
# Lëtzebuerger Informatiksolympiad 2023

Finals

Task descriptions

## Instructions

- The allowed programming language is C/C++.
- All the programs must be console applications. For instructions how to write a console application in the allowed programming languages, please refer to the remarks on the site `www.infosolympiad.lu` under the heading *The tasks*.
- The input of a program can mean either the direct entry of data from the keyboard (referred to as ”standard input”) or input via specific function calls described in the task description (”no reading”). Similarly, the output of a program can mean either printing to the console (referred to as ”standard output”) or output via specific functions described in the task description (”no writing”).
- The formats of the input and output data shown in the execution examples must be respected. Other formats will not be accepted.
- For testing, submitting and evaluating a program, the source file with the correct file extension `c/cpp` must be uploaded to the automated online judge CMS (Contest Management System), accessible via the homepage `www.infosolympiad.lu` or directly via the URL `http://158.64.46.20:81`. Please use your personal login (username and password) to access your account on the CMS. The filename of the single source file should be the same than the task name. Please refer to the CMS for technical details on how to test and submit a program.
- Time limits and memory constraints are described both in the task statements and the CMS. Please refer to the CMS for compilation commands.
- You are allowed to ask questions via the CMS, however no hints concerning the use of programming languages, the implementation of algorithms or the solutions to the tasks will be given. Questions should be about CMS or seek clarification concerning the task descriptions.



# Road Building

## Description

To improve traffic, Luxembourg city decided to make all its roads one-way only. The city consists of $N$ intersections that are connected via $M$ roads. Before the change, the city was connected, which means that one was able to travel from any intersection to any other intersection via the road network. After the change however, this is not necessarily the case anymore. It is possible that intersections that were connected via bilateral roads before the change are not connected anymore after restricting the roads to a single direction. As it is very important for the city to stay traversable after the change, you are tasked with determining if the city is still connected, and to add additional one-way roads to make it connected again, if necessary.

## Task

Given the city's plan of $M$ one-way roads that connect $N$ intersections, find the minimum amount of one-way roads to build to make sure that every intersection is reachable from any other intersection.
Note that the city might not need to build any additional roads.

## Example

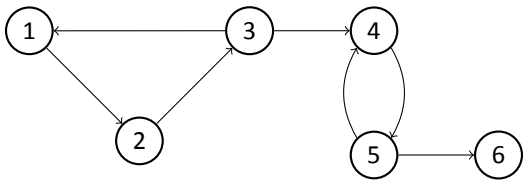



Figure 1: First Example

In example 1, adding a single road is enough to make the city traversable again (e.g. $6 \to 1$). This example corresponds to the execution example.

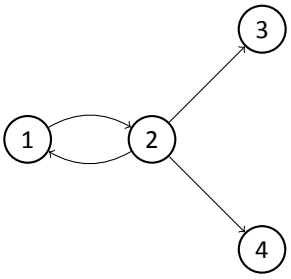



Figure 2: Second Example

In example 2, a minimum two roads are needed. (e.g. $3 \to 1$ and $4 \to 2$)

## Constraints

- $2 \leq N \leq 10^5$
- $N - 1 \leq M \leq 6 \cdot 10^5$
- Before introducing one-way roads, the city is connected. In other words the city is connected if one can traverse any road in both directions.

## Input and output of program

### Input

The first line contains the integers $N$ and $M$, separated by a space.
The following $M$ lines each contain integers $u$ and $v$, separated by a space, such that $1 \leq u, v \leq N$ and the pair $(u, v)$ represents a road from intersection $u$ to intersection $v$.

### Output

A single integer, the minimum amount of roads to build.

## Execution example

### Input

```
6 7
1 2
2 3
3 1
3 4
4 5
5 4
5 6
```

### Output

```
1
```

## Distribution of points

| Subtask | Points | Constraints/Description |
|---|---|---|
| 1 | 10 | $N \leq 10,\ M \leq 100$, the answer is in $\{0, 1, 2\}$ |
| 2 | 10 | $N \leq 2.000,\ M < 20.000$, the answer is only $0$ or $1$ |
| 3 | 10 | $M = N - 1$ |
| 4 | 20 | No additional constraints |

## Technical constraints

| | |
|---|---|
| **Task name** | road |
| **Input file** | standard input |
| **Output file** | standard input |
| **Time limit** | 1 second |
| **Memory limit** | 256 megabytes |

# Traffic Control

## Description

When large planes take off, they create turbulence behind them, that makes it difficult for smaller planes to take off right afterwards on the same runway. Thus, there are waiting requirements in place to make sure all planes, no matter their weight, can take off safely. For instance at Findel Airport, planes are subdivided into three types: light, medium and heavy. If a medium plane takes off after a heavy plane, it needs to wait for at least $3$ minutes before taking off. However, a heavy plane does not need to wait after a light or medium plane has taken off. At busy airports with a multiple runways, where a queue of planes waits to be assigned runways, it can thus make a big difference in efficiency how planes are allocated to runways. It is your task to allocate runways in such a way that all the planes take off in the minimum amount of time.

## Task

Suppose you are at an airport which has either $R = 1$ or 2 runways. This airport has $T$ types of planes, numbered from $1$ to $T$. The time a plane of type $b$ needs to wait after a plane of type $a$ has taken off is $w_{ab}$ minutes. Currently there are $N$ planes queued up before the runway(s) of types $t_i$ where $1 \leq i \leq N$. It is your task to assign each plane to one of the available runways in such a way that the total time taken by all the planes to take off is minimal. It is a requirement that plane $i$ takes off before plane $j$ if $i < j$ and they take off on the same runway.

## Example

### First example

Consider Findel airport with $R = 1$ and three types of planes: light, medium and heavy. We can write the waiting times in matrix form as:

$$w = \begin{pmatrix} 0 & 0 & 0 \\ 3 & 0 & 0 \\ 3 & 3 & 0 \end{pmatrix}.$$

As there is only one runway, all planes must use it in the order they are queued up. For instance, if the queue looks like $3, 2, 3, 2, 1$ then the total take off time will be $9$ minutes, due to the three instances where heavier planes are queued in front of lighter planes.

### Second example

Consider Frankfurt airport with $R = 2$ and again three types of planes. Due to more complicated interactions between planes, we can write the waiting times in matrix form as:

$$w = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 2 \\ 2 & 3 & 5 \end{pmatrix}.$$

Assume the queue looks like $3, 3, 2, 1, 2$. Then the total take off time will be $4$ minutes obtained by sending the queue to runways $1, 2, 1, 2, 1$ respectively. Note that plane number $4$ does not need to wait for plane number $3$ to take off, as they depart from different runways.

## Constraints

- $R = 1$ or $2$
- $1 \leq T \leq 5$
- $1 \leq N \leq 1000$
- $0 \leq w_{ab} \leq 10$
- $1 \leq t_i \leq T$

## Input and output of program

### Input data

The first line contains the integers $R, T$ and $N$, separated by spaces. The next $T$ lines contain $T$ integers each, separated by spaces. Line $i+1$ where ($1 \leq i \leq T$) contains the integers $w_{ij}$, for $1 \leq j \leq T$. The next $N$ lines contain one integer each. Line $i+T+1$ contains the integer $t_i$, where $1 \leq i \leq N$.

### Output data

The output should consist of a single integer, the minimum time it takes for all the planes to take off given the constraints outlined above in the task description.

## Execution examples

The input for the first example is the following:

**Input**

```
1 3 5
0 0 0
3 0 0
3 3 0
3
2
3
2
1
```

**Output**

```
9
```

The input for the second example is the following:

**Input**

```
2 3 5
1 0 0
3 1 2
2 3 5
3
3
2
1
2
```

**Output**

```
4
```

## Distribution of points

| Subtask | Points | Constraints/Description |
|---|---|---|
| 1 | 10 | $R = 1$ |
| 2 | 5 | $R = 2, T = 2$. The only non-zero waiting time is $w_{12}$. |
| 3 | 25 | No additional constraints |

## Technical constraints

| | |
|---|---|
| **Task name** | task |
| **Input file** | standard input |
| **Output file** | standard input |
| **Time limit** | 1 second |
| **Memory limit** | 256 megabytes |