HW4: Network Flows (Deadline: 1400/11/12)

Part 1. Evacuating People

In this problem, you will apply an algorithm for finding maximum flow in a network to determine how fast people can be evacuated from the given city.

Problem Description. A tornado is approaching the city, and we need to evacuate the people quickly. There are several roads outgoing from the city to the nearest cities and other roads going further. The goal is to evacuate everybody from the city to the capital, as it is the only other city which is able to accommodate that many newcomers. We need to evacuate everybody as fast as possible, and your task is to find out what is the maximum number of people that can be evacuated each hour given the capacities of all the roads.

Input Format. The first line of the input contains integers n and m — the number of cities and the number of roads respectively. Each of the next m lines contains three integers u, v and c describing a particular road — start of the road, end of the road and the number of people that can be transported through this road in one hour. u and v are the 1-based indices of the corresponding cities.

The city from which people are evacuating is the city number 1, and the capital city is the city number n. Note that all the roads are given as one-directional, that is, you cannot transport people from v to u using a road that connects u to v. Also note that there can be several roads connecting the same city u to the same city v, there can be both roads from u to v and from v to u, or there can be only roads in one direction, or there can be no roads between a pair of cities. Also note that there can be roads going from a city u to itself in the input.

When evacuating people, they cannot stop in the middle of the road or in any city other than the capital. The number of people per hour entering any city other than the evacuating city 1 and the capital city n must be equal to the number of people per hour exiting from this city. People who left a city u through some road (u, v, c) are assumed to come immediately after that to the city v. We are interested in the maximum possible number of people per hour leaving the city 1 under the above restrictions.

Constraints. $1 \leq n \leq 100$; $0 \leq m \leq 10{,}000$; $1 \leq u, v \leq n$; $1 \leq c \leq 10{,}000$. It is guaranteed that $m \times \text{EvacuatePerHour} \leq 2\times108$, where EvacuatePerHour is the maximum number of people that can be evacuated from the city each hour, the number which you need to output.

Output Format. Output a single integer, the maximum number of people that can be evacuated from the city number 1 each hour, and a visual representation of the solution on the graph.

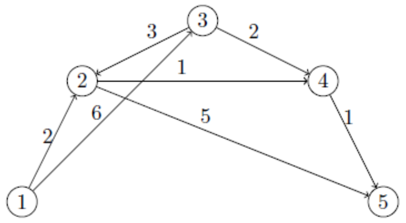
Time Limits.

Language	*C*	*C++*	*Java*	*Python, Ruby*
Time(s)	**1**	**1**	**5**	**45**

Sample.

Input:
```
5 7
1 2 2
2 5 5
1 3 6
3 4 2
4 5 1
3 2 3
2 4 1
```
Output:
```
6
```

In this sample, the road graph with capacities looks like this:



We can evacuate 2 people through the route 1−2−5, additional 3 people through the route 1−3−2−5 and 1 more person through the route 1−3−4−5 — for a total of 6 people. It is impossible to evacuate more people each hour, as the total capacity of all roads incoming to the capital city 5 is 6 people per hour.

What to Do. Implement an algorithm for finding maximum flow described in the lectures, but be careful with the choice of the algorithm.

Part 2. Stock Charts

In this problem you will need to guess how to apply the network algorithms to find the most compact way of visualizing stock price data using charts.

Problem Description. You are in the middle of writing your newspaper's end-of-year economics summary, and you've decided that you want to show a number of charts to demonstrate how different stocks have performed over the course of the last year.

You've already decided that you want to show the price of n different stocks, all at the same k points of the year. A simple chart of one stock's price would draw lines between the points (0, $price_0$), (1, $price_1$), . . ., (k−1, $price_{k-1}$), where $pricei$ is the price of the stock at the ith point in time.

In order to save space, you have invented the concept of an overlaid chart. An overlaid chart is the combination of one or more simple charts, and shows the prices of multiple stocks (simply drawing a line for each one). In order to avoid confusion between the stocks shown in a chart, the lines in an overlaid chart may not cross or touch.

Given a list of n stocks' prices at each of k time points, determine the minimum number of overlaid charts you need to show all of the stocks' prices.

Input Format. The first line of the input contains two integers n and k — the number of stocks and the number of points in the year which are common for all of them. Each of the next n lines contains k integers. The ith of those n lines contains the prices of the ith stock at the corresponding k points in the year.

Constraints. $1 \le n \le 100$; $1 \le k \le 25$. All the stock prices are between 0 and 1,000,000.

Output Format. Output a single integer, the minimum number of overlaid charts to visualize all the stock price data you have, and a visual representation of the solution.

Time Limits.

Language	*C*	*C++*	*Java*	*Python, Ruby*
Time(s)	**2**	**2**	**3**	**10**

Sample.

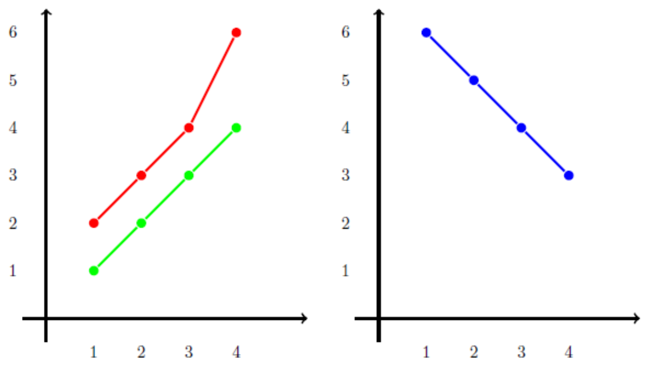
Input:

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

Output:

```
2
```

This data can be put into two following overlaid charts.



However, we cannot put all the data in one overlaid chart, as the lines corresponding to the third stock would touch the lines corresponding to the second stock, because they have the same price value at the third point.

What to Do. Try to reduce the problem to the maximum matching in a bipartite graph problem and use a proper algorithm to tackle it.

General Instructions

Your codes must contain proper comments and be well-designed. You have to report the result obtained by your algorithm for each available instance. For each problem, the *instances* folder includes the instances as well as their answers.

Good Luck!