Problem A: Density index

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

We define the density index of a connected undirected graph as the reciprocal of the average pairwise distance between distinct vertices in it, s.t. it is 1 if and only if the graph is complete, $\frac{1}{2}$ if the average distance is 2 etc.

Your job is to write a program that can determine the density index of any connected undirected graph.

Input

The input consists of

- one line containing N and M ($2 \leq N \leq 10^3,\ N - 1 \leq M \leq 10^4$) – the numbers of vertices and edges in the graph
- M lines each containing two numbers u, v ($1 \leq u, v \leq N$) – indicating an edge connecting vertices u and v.

It is guaranteed that the given graph is connected.

Output

Output the graph's density index with a precision of at least 10^{-4}.

Sample input and output

Input	**Output**
5 5 1 2 2 3 3 4 5 3 1 5	0.625000

Problem B: Traffic

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Since your state government is currently planning to renovate a nearby segment of the highway running next to your town, you are worried about what might happen to said town if all the highway traffic was redirected through it. To be able to make a compelling argument to them that an extra detour road outside the town will be needed, you decided to determine how many cars can pass through the town's current road network each minute.

For that purpose, you already created a version of your town's road network where the junctions are numerated $1, ..., N$ and each of the streets is assigned an integer describing how many cars can drive into/out of it in one direction per minute. Of course you left out ring roads.

Since the streets have the same capacity in both directions anyway, you are only interested how many cars could enter/leave the town per minute in one of them.

Input

The input consists of

- one line containing N and M ($2 \leq N \leq 100,\ 0 \leq M \leq \frac{N(N-1)}{2}$) – the numbers of junctions and streets, respectively
- M lines each containing integers a, b and c ($1 \leq a, b \leq N,\ 1 \leq c \leq 10^9$) – the first two being the indices of the junctions the street connects and the third one being its capacity as defined above. Junctions 1 and N are the ones one starts/ends up in when entering/leaving the town.

Output

Output a single integer – the number of cars that can enter/leave the town per minute in one direction.

Sample input and output

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

Problem C: Cancel Culture

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

In the era of cancel culture, social networks are becoming increasingly fragile. Whereas, in the past, the main type of friendship-action in these networks was the forming of a new friendship between two users, now it's the exact opposite: More and more friends are canceling on each other.

As a social media analyst, you were tasked with examining this phenomenon more closely. To be able to do that, you need to assess the stability of given *communities*, a *community* in this context being a network of people that are befriended directly or indirectly through intermediate friends (e.g. a possibility for persons a and b to be befriended indirectly is for there to be persons c and d s.t. a is friends with c, c is friends with d and d is friends with b). The stability of a community is defined to be the minimum number of friendships that need to be canceled to split the community (s.t. there are people a and b that both were previously in the community but are no longer directly or indirectly befriended).

Write a program that can calculate a given community's stability.

Input

The input consists of

- one line containing integers N and M ($2 \leq N \leq 100,\ 0 \leq M \leq \frac{N(N-1)}{2}$) – the numbers of people and friendships in the community
- M lines each containing two integers a and b ($1 \leq a, b \leq N$) denoting a (mutual) friendship between persons a and b.

Output

Output the stability of the given community, i.e. the minimum number of friendships that need to be canceled in order to split it.

Sample input and output

Input	Output
3 3 1 2 2 3 3 1	2

Problem D: Dungeon Explorer 2: Teambuilding Exercise

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

In recent months, you've become an avid player of *Dungeons and Demons*, a very popular new MMORPG. At the moment, your guild is planning to clear a type 2 dungeon, which is an extensive network of huge halls that are filled with monsters and connected by doors and corridors. Through a community effort, the dungeon in question has already been thoroughly analyzed so that, among other things, you know its entire layout.

Clearing a type 2 dungeon generally works as follows: Before entering it, the guild splits up into teams. Then the teams enter the dungeon in distinct halls and start to clear them. After a team has cleared its initial hall, it might be able to use a door or corridor to get to another one and start clearing that too, and so on. However, all doors and corridors only provide one-way access and – in order to avoid friendly fire – the teams' paths should never cross, regardless of how quickly each team clears its halls.

Now the guild's leaders are arguing about the number of teams needed for the dungeon in question. Can you help them?

Input

The input consists of

- one line containing integers N and M ($2 \leq N \leq 500,\ 0 \leq M \leq \frac{N(N-1)}{2}$) – the numbers of halls and doors/corridors in the dungeon. The halls are numerated $1, ..., N$.
- M lines each containing two numbers a and b ($1 \leq a, b \leq N$), denoting a door/corridor from hall a to hall b. No pair (a, b) occurs in the input more than once and it is known that there are no cyclic paths in the dungeon (so once one leaves a hall, there is no possibility to get back into it through a series of doors, corridors and other halls).

Output

Output one number – the minimum amount of teams you will need.

Sample input and output

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

Problem E: Bricks

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Joe's parents have bought him a set of bricks as a birthday present. As he is currently in first grade, they decided to buy bricks that have single letters on each side.

Now Joe wants to demonstrate to his older sister that he already learned how to spell. For that he plans to place the bricks in such a way that they spell her name. But this is not an easy task, because two letters could be on different sides of the same brick – then Joe couldn't use both letters to make a word! However, a letter can appear on different bricks. Joe will require your help to figure this out.

Given a set of bricks and the name of Joe's sister. Find out if it is possible to place the bricks in such a way that they spell her name.

Input

The input consists of

- one line containing N ($1 \leq N \leq 100$) – the number of bricks in Joe's set
- one line containing the name of Joe's sister – a word with only capital letters not longer than 100 characters
- N lines each containing the 6 capital letters that are on the sides of the corresponding brick.

Output

Output `YES` if he can spell her name with the bricks and `NO` otherwise.

Sample input and output

Input	Output
4 ANN ANNNNN BCDEFG HIJKLM NOPQRS	NO
5 HELEN ABCDEF GHIJKL MNOPQL STUVWN EIUOZK	YES

Problem F: Topic Selection

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You are currently creating math learning software for high school students, and in the process of doing so, encountered a rather difficult problem: There are countless topics in mathematics, far more than could be included in your software. Furthermore, a topic can either benefit the students learning about it or cause them brain damage (e.g. geometry) and while studies have shown for each topic how much it helps or hurts students, you can't just take only the most valuable topics, as their contents might depend on other topics too. Instead you need to find a subset of the contemplable topics that maximizes the overall benefit for the students (which is the sum of the values for the chosen topics) and is self-contained in that none of the chosen topics depend on any non-chosen topics.

Input

The input consists of

- one line containing n ($1 \leq n \leq 500$) – the number of topics to be considered
- n lines describing the topics, where the i-th description begins with integers x_i and d_i ($|x| \leq 10^6, 0 \leq d_i < n$) – the first being the benefit or damage associated with topic i and the second being the number of topics that topic i depends on – and continues with d_i integers $b_{i,1}, ..., b_{i,d_i}$ ($1 \leq b_{i,j} \leq n, b_{i,j} \neq i$) – the list of topics that topic i depends on. It is guaranteed that there are no cyclic dependencies.

Output

First output the maximum overall benefit as described above. Then output the indices of the topics that should be chosen to achieve this benefit (in any order, separated by spaces).

Sample input and output

Input	Output
4 -3 0 5 2 1 3 2 1 4 10 0	14 4 3 2 1
7 2 1 4 -3 1 1 5 1 2 -3 0 20 1 4 -16 1 5 14 1 6	21 5 4 3 2 1
1 -100 0	0