15-295 Spring 2019 #4 DFS in Graphs

A. Party

3 seconds, 256 megabytes

A company has n employees numbered from 1 to n. Each employee either has no immediate manager or exactly one immediate manager, who is another employee with a different number. An employee A is said to be the superior of another employee B if at least one of the following is true:

- Employee A is the immediate manager of employee B
- Employee B has an immediate manager employee C such that employee A is the superior of employee C.

The company will not have a managerial cycle. That is, there will not exist an employee who is the superior of his/her own immediate manager.

Today the company is going to arrange a party. This involves dividing all n employees into several groups: every employee must belong to exactly one group. Furthermore, within any single group, there must not be two employees A and B such that A is the superior of B.

What is the minimum number of groups that must be formed?

Input

The first line contains integer n ($1 \le n \le 2000$) — the number of employees.

The next n lines contain the integers p_i ($1 \le p_i \le n$ or $p_i = -1$). Every p_i denotes the immediate manager for the i-th employee. If p_i is -1, that means that the i-th employee does not have an immediate manager.

It is guaranteed, that no employee will be the immediate manager of him/herself ($p_i \neq i$). Also, there will be no managerial cycles.

Output

Print a single integer denoting the minimum number of groups that will be formed in the party.

input
5 -1 1 2 1 -1
output
3

For the first example, three groups are sufficient, for example:

- Employee 1
- Employees 2 and 4
- Employees 3 and 5

B. Badge

1 second, 256 megabytes

In Summer Informatics School, if a student doesn't behave well, teachers make a hole in his badge. And today one of the teachers caught a group of n students doing yet another trick.

Let's assume that all these students are numbered from 1 to n. The teacher came to student a and put a hole in his badge. The student, however, claimed that the main culprit is some other student p_a.

After that, the teacher came to student p_a and made a hole in his badge as well. The student in reply said that the main culprit was student p_{p_a}.

This process went on for a while, but, since the number of students was finite, eventually the teacher came to the student, who already had a hole in his badge.

After that, the teacher put a second hole in the student's badge and decided that he is done with this process, and went to the sauna.

You don't know the first student who was caught by the teacher. However, you know all the numbers p_i. Your task is to find out for every student a, who would be the student with two holes in the badge if the first caught student was a.

Input

The first line of the input contains the only integer n ($1 \le n \le 1000$) — the number of the naughty students.

The second line contains n integers p_1, ..., p_n ($1 \le p_i \le n$), where p_i indicates the student who was reported to the teacher by student i.

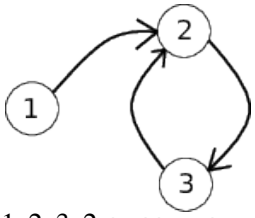
Output

For every student a from 1 to n print which student would receive two holes in the badge, if a was the first student caught by the teacher.

input
3 2 3 2
output
2 2 3

input
3 1 2 3
output
1 2 3

The picture corresponds to the first example test case.



When $a = 1$, the teacher comes to students $1, 2, 3, 2$, in this order, and the student 2 is the one who receives a second hole in his badge.

When $a = 2$, the teacher comes to students $2, 3, 2$, and the student 2 gets a second hole in his badge. When $a = 3$, the teacher will visit students $3, 2, 3$ with student 3 getting a second hole in his badge.

For the second example test case it's clear that no matter with whom the teacher starts, that student would be the one who gets the second hole in his badge.

C - Battle for Silver

Piet Hein was a Dutch naval officer during the Eighty Years' War between the United Provinces of The Netherlands and Spain. His most famous victory was the capture of the *Zilvervloot* ('Silver Fleet') near Cuba in 1628, where he intercepted a number of Spanish vessels that were carrying silver from the Spanish colonies in the Americas to Spain. Details about this famous naval battle are sketchy, so the description below may contain some historical inaccuracies.

The Silver Fleet consisted of vessels containing silver coins. Piet Hein's basic strategy was simple: tow away a number of vessels from the fleet, in order to capture their contents.

In an attempt to prevent the Dutch from carrying out this plan, the Spanish tied all the ships in their fleet together using huge iron chains. Each vessel in their fleet was fixed to at least one other vessel; any two vessels were connected by at most one chain; and the Spanish made sure that the chains did not cross each other, otherwise they could get tied up into a knot. As an end result, the vessels and the chains connecting them formed a connected, planar graph.

However, the Spanish preventive measures only made their situation worse. As an experienced naval officer, Piet Hein knew that towing away a group of ships was easiest if, for every two ships in the group, the ships were connected by a chain. He called such groups *chaingroups*.

Piet Hein ordered his men to tow away all the ships in the chaingroup that contained the largest amount of booty, after severing the links with the remaining ships in the Spanish fleet with a few highly accurate canon shots. The total booty in a chaingroup is the total number of silver coins in the vessels that make up the chaingroup.

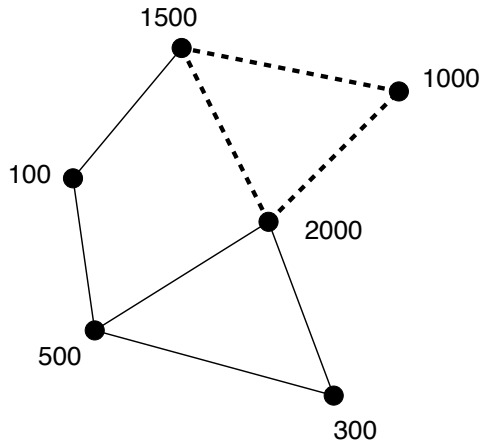


Figure 1 – The Silver Fleet represented as a graph: each dot denotes a vessel in the fleet, while each line denotes a chain that connects two vessels. The vessels that are connected in the figure by the dashed lines correspond to the chaingroup that provides the highest total value of silver coins. In this case, Piet Hein loots 4500 silver coins from the fleet.

Task

Given a description of the Silver Fleet, find the value of the chaingroup with the highest amount of booty (i.e., total number of silver coins in the ships that make up the chaingroup).

Input

For each test-case:

- A line containing two integers v ($2 \leq v \leq 450$) and e ($1 \leq e \leq 900$), the number of vessels in the fleet and the number of chains, respectively.
- Then, v lines specifying $S_1, S_2, \ldots, S_v$, the amount of silver coins carried by vessel i ($1 \leq i \leq v$). The S_i will be positive integers, where $100 \leq S_i \leq 6000$.
- Then, for each chain, a line containing two integers c_{start} and c_{end}, the two vessels connected by the chain, where ($1 \leq c_{start} < c_{end} \leq v$).

Each fleet forms a connected, planar graph.

Output

For each test case, one line containing a single positive integer: the number of silver coins that is captured by Piet Hein's fleet.

Example

input	output
4 6 100 5000 1000 2000 1 2 1 3 1 4 2 3 2 4 3 4 6 8 1500 1000 100 2000 500 300 1 2 1 3 1 4 2 4 3 5 4 5 4 6 5 6	8100 4500

D. Ring Road 2

2 seconds, 256 megabytes

It is well known that Berland has n cities, which form the Silver ring — cities i and $i+1$ ($1 \le i < n$) are connected by a road, as well as the cities n and 1. The goverment have decided to build m new roads. The list of the roads to build was prepared. Each road will connect two cities. Each road should be a curve which lies inside or outside the ring. New roads will have no common points with the ring (except the endpoints of the road).

Now the designers of the constructing plan wonder if it is possible to build the roads in such a way that no two roads intersect (note that the roads may intersect at their endpoints). If it is possible to do, which roads should be inside the ring, and which should be outside?

Input

The first line contains two integers n and m ($4 \le n \le 100$, $1 \le m \le 100$). Each of the following m lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$). No two cities will be connected by more than one road in the list. The list will not contain the roads which exist in the Silver ring.

Output

If it is impossible to build the roads in such a way that no two roads intersect, output `Impossible`. Otherwise print m characters. i-th character should be `i`, if the road should be inside the ring, and `o` if the road should be outside the ring. If there are several solutions, output any of them.

input
4 2 1 3 2 4
output
io

input
6 3 1 3 3 5 5 1
output
ooo

Problem E : Digraphs

A *digraph* is a graph with orientation... oh, sorry, not this time. Let's stop being nerds for a minute and talk about languages (*human* languages, not PHP).

Digraphs are pairs of characters that represent one phoneme (sound). For example, "ch" in English (as in "church") is a single consonant sound. The languages of Central Europe are fond of digraphs: Hungarian "sz", Czech "ch" and Polish "rz" are fine examples of them.

Digraphs are very annoying for people who do not use them natively. We will make up a letter-puzzle specifically for those people. Given a list of digraphs, construct a biggest possible square of lower case English letters such that its rows and columns *do not* contain any of these digraphs. This means that no two consecutive letters (read from top to bottom or from left to right) can form a digraph.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with an integer n, $0 \leqslant n \leqslant 676$, denoting the number of forbidden digraphs. The n following lines contain the digraphs.

Output

For each test case print a square of the largest possible size which does not contain any of the digraphs. If it is possible to construct a square of size 20×20 or bigger, print only a 20×20 square.

Example

Part of the example test data below was omitted for clarity. You can access full sample tests at your workstation.

For an example input	a possible correct answer is:
2 628 aa az ba bb bc ... by ca cb cc ... cy da ... dy ... wa ... wy ya ... yy za zb ... zy zz 2 aa bb	aw wz abababababababababab bababababababababaa abababababababababab bababababababababaa abababababababababab bababababababababaa abababababababababab bababababababababaa abababababababababab bababababababababaa abababababababababab bababababababababaa abababababababababab bababababababababaa abababababababababab bababababababababaa abababababababababab bababababababababaa

F - Absurdistan Roads

The people of Absurdistan discovered how to build roads only last year. After the discovery, every city decided to build their own road connecting their city with another city. Each newly built road can be used in both directions.

Absurdistan is full of surprising coincidences. It took all N cities precisely one year to build their roads. And even more surprisingly, in the end it was possible to travel from every city to every other city using the newly built roads.

You bought a tourist guide which does not have a map of the country with the new roads. It only contains a huge table with the shortest distances between all pairs of cities using the newly built roads. You would like to know between which pairs of cities there are roads and how long they are, because you want to reconstruct the map of the N newly built roads from the table of shortest distances.

Task

You get a table of shortest distances between all pairs of cities in Absurdistan using the N roads built last year. From this table, you must reconstruct the road network of Absurdistan. There might be multiple road networks with N roads with that same table of shortest distances, but you are happy with any one of those networks.

Input

For each test case:

- A line containing an integer N ($2 \leq N \leq 2000$) – the number of cities and roads.
- N lines with N numbers each. The j-th number of the i-th line is the shortest distance from city i to city j. All distances between two distinct cities will be positive and at most 1 000 000. The distance from i to i will always be 0 and the distance from i to j will be the same as the distance from j to i.

Output

For each test case:

- Print N lines with three integers 'a b c' denoting that there is a road between cities $1 \leq a \leq N$ and $1 \leq b \leq N$ of length $1 \leq c \leq 1\,000\,000$, where $a \neq b$. If there are multiple solutions, you can print any one and you can print the roads in any order. At least one solution is guaranteed to exist.

Print a blank line between every two test cases.

Example

input	output
4 0 1 2 1 1 0 2 1 2 2 0 1 1 1 1 0 4 0 1 1 1 1 0 2 2 1 2 0 2 1 2 2 0 3 0 4 1 4 0 3 1 3 0	2 1 1 4 1 1 4 2 1 4 3 1 2 1 1 3 1 1 4 1 1 2 1 1 3 1 1 2 1 4 3 2 3

G - Guards

The royal castles in Molvania follow the design of king Sane, first of his dynasty. He ruled by divide and conquer. Therefore, all castles are built according to a hierarchical pattern based on interconnected buildings. A building consists of *halls* and *corridors* that connect halls.

Initially, a castle consists of only one building (the *main building*). When its population grows, the castle is extended as follows: A new peripheral building is constructed, attached to one of the existing buildings. Like any other building, the new building also consists of halls and corridors. An additional corridor is created to connect a hall in the existing building to a hall in the new building. That corridor is the only way to access the new building.

The number of halls in a building is at most 10.

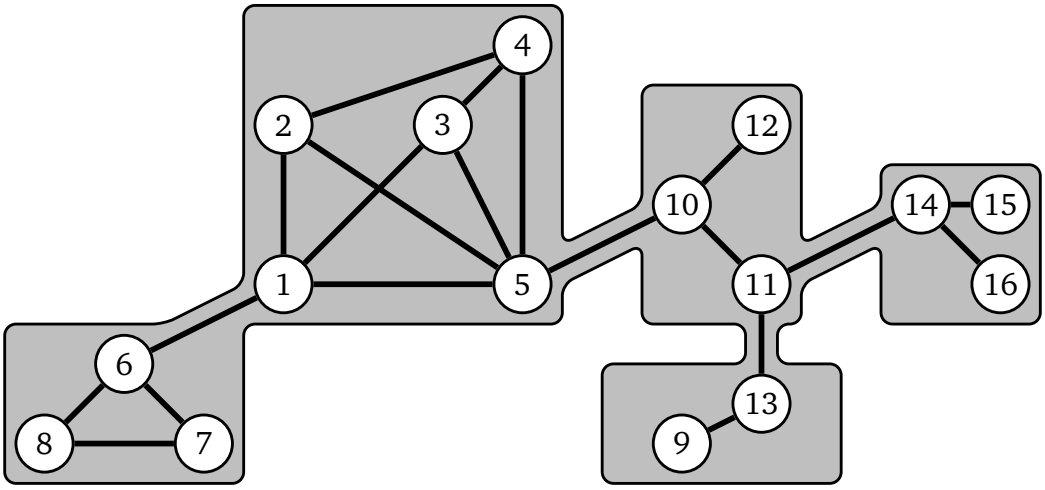


Figure 1: The castle layout of the example provided below.

In times of turmoil, the king monitors all corridors by strategically placing guards in halls. He asks you to determine the least number of guards required to monitor all corridors in the castle (as he wants to keep his personal guard as large as possible). Note that since the last fire, there are no doors in the castle, so we can safely assume that a guard placed in a hall can monitor all connecting corridors.

Input

The input contains a number of castle descriptions. Within a castle, each hall is identified by a unique number between 1 and 10000. Each castle is recursively defined, starting with a description of the main building:

1. A line containing three integers, representing the number of halls in this building ($2 \leq n \leq 10$), the number of corridors in this building ($1 \leq m \leq 45$), and the number of peripheral buildings that were later attached to this building ($0 \leq w \leq 10$).
2. For each of the m corridors:
    - A line containing two integers (each $\leq$ 10000), representing the two halls connected by this corridor. Both halls are located inside the current building.

3. For each of the w peripheral buildings:
    - A line containing two integers, describing the corridor that leads to this peripheral building. The first integer represents a hall in the current building, while the second integer represents a hall in the peripheral building.
    - The structure of the peripheral building and any newer buildings that were later attached to it, described by repeating rules 1 to 3.

The castle is fully connected: any hall is directly or indirectly reachable from any other hall. Corridors with the same start and end hall do not exist, and for every two halls there is at most one corridor between them.

Output

For each castle, print a single line containing a positive integer: the minimum number of guards to place in halls such that all corridors in the castle are monitored.

Example

input	output
5 8 2 1 2 2 4 3 4 1 3 1 5 2 5 3 5 4 5 1 6 3 3 0 6 7 7 8 8 6 5 10 3 2 2 10 11 10 12 11 13 2 1 0 13 9 11 14 3 2 0 14 15 14 16	8