LightOJ

Forming the Council

1 seconds 64 MB Medium Hard LOJ-1251 Debug

Statement Submissions Statistics Tutorial

English

In a city there are **n** voters, and **m** people formed the Govt. council. The council members are numbered from **1** to **m**. Now everyone is complaining that the council is biased. So, they made a plan. The plan is that the voters are given a chance to vote again to form the new council. A vote will be like **±i ±j**. **'+'** means the voter wants that member to be in the council, **'-'** means the voter doesn't want the member to be in the council. For example, there are 4 voters, they voted like

- **+1 -3 the voter wants member 1 to be kept in the council or member 3 to be thrown out**
- **+2 +3 the voter wants member 2 to be kept in the council or member 3 to be kept in the council**
- **-1 -2 the voter wants member 1 to be thrown out or member 2 to be thrown out**
- **-4 +1 the voter wants member 4 to be thrown out or member 1 to be kept in the council**

A voter will be satisfied if at least one of his wishes becomes true. Now your task is to form the council such that all the voters are happy.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing two integers **n (1 ≤ n ≤ 20000)** and **m (1 ≤ m ≤ 8000)**. Each of the next **n** lines contains a vote in the form **±i ±j (1 ≤ i, j ≤ m)**.

Output

For each case, print the case number and **'Yes'** if a solution exists, or **'No'** if there is no solution. Then if the result is yes, print another line containing the number of members in the

Sample

Input	Output
3 4 3 +1 +3 +2 -1 +2 -3 -1 -2 4 2 +1 -2 +1 +2 -1 -2 -1 +2 1 3 +1 -3	Case 1: Yes 2 2 3 Case 2: No Case 3: Yes 0

Click to copy

Notes

This is a special judge problem. Wrong output format may cause wrong answer.

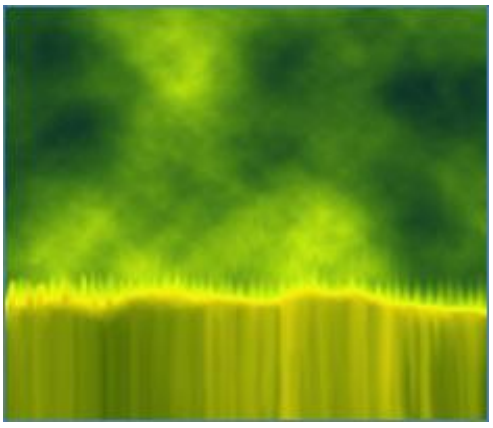
LightOJ

Explosion

2 seconds 64 MB Hard LOJ-1407 Debug

Statement Submissions Statistics Tutorial

English



Planet Krypton is about to explode. The inhabitants of this planet have to leave the planet immediately. But the problem is that, still some decisions have to be made - where to go, how to go etc. So, the council of Krypton has invited some of the people to meet in a large hall.There are **n** people in planet Krypton, for simplicity they are given ids from **1** to **n**. The council uses a super computer named Oracle to call them in the meeting. Oracle has four types of messages for invitation.

The message format is **type x y**, where **x** and **y** are two different person's ids and **type** is an integer as follows:

1. **1 x y** means that either **x** or **y** should be present in the meeting.
2. **2 x y** means that if **x** is present, then no condition on **y**, but if **x** is absent **y** should be absent.
3. **3 x y** means that either **x** or **y** must be absent.
4. **4 x y** means that either **x** or **y** must be present but not both.

Each member of the council has an opinion too. The message format is **type x y z**, where **x, y** and **z** are three different person's ids and **type** is an integer as follows:

1. **1 x y z** means that at least one of **x, y** or **z** should be present.
2. **2 x y z** means that at least one of **x, y** or **z** should be absent.

Now you have to find whether the members can be invited such that every message by oracle and the council members are satisfied.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains three integers **n, m** and **k (3 ≤ n ≤ 1000, 0 ≤ m ≤ 2000, 0 ≤ k ≤ 5)** where **m** means the number of messages by oracle, **k** means the total members in the council. Each of the next **m** lines will contain a message of Oracle in the format given above.

Output

For each case, print the case number and whether it's possible to invite the people such that all the messages are satisfied. If it's not possible, then print `Impossible.` in a single line. Otherwise, print `Possible` and the number of invited people and the ids of the invited people in ascending order. Print the line leaving a single space between fields. Terminate this line with a `.`. See the samples for more details. There can be multiple answers; print any valid one.

Sample

Input	Output
3 3 2 1 3 2 1 1 2 3 1 1 2 3 4 4 1 2 2 1 4 1 2 4 1 3 4 1 4 2 2 3 4 4 5 0 3 1 2 2 2 3 2 2 4 2 1 2 2 2 1	Case 1: Possible 2 1 3. Case 2: Impossible. Case 3: Possible 0.

Notes

This is a special judge problem; wrong output format may cause 'Wrong Answer'.

LightOJ

Critical Links

1 seconds 64 MB Medium LOJ-1026 Debug

Statement Submissions Statistics Tutorial

English

In a computer network a link **L**, which interconnects two servers, is considered critical if there are at least two servers **A** and **B** such that all network interconnection paths between **A** and **B** pass through **L**. Removing a critical link generates two disjoint sub-networks such that any two servers of a sub-network are interconnected.

For example, the network shown in figure **1** has three critical links that are marked red: **0 - 1**, **3 - 4** and **6 - 7** in figure **2**.

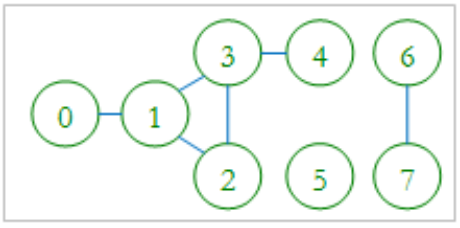


Figure 1: Original Graph

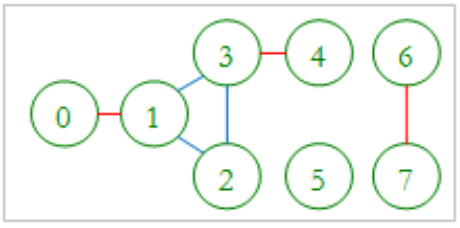


Figure 2: The Critical Links

It is known that:

1. The connection links are bi-directional.
2. A server is not directly connected to itself.
3. Two servers are interconnected if they are directly connected or if they are interconnected with the same server.
4. The network can have stand-alone sub-networks.

Write a program that finds all critical links of a given computer network.

Input

Input starts with an integer **T (≤ 15)**, denoting the number of test cases.

Each case starts with a blank line. The next line will contain **n (0 ≤ n ≤ 10000)** denoting the number of nodes. Each of the next n lines will contain some integers in the following format:

`u (k) v1 v2 ... vk`

Output

For each case, print the case number first. Then you should print the number of critical links and the critical links, one link per line, starting from the beginning of the line, as shown in the sample output below. The links are listed in ascending order according to their first element and then second element. Since the graph is bidirectional, print a link **u v** if **u** < **v**.

Sample

Input	Output
3 8 0 (1) 1 1 (3) 2 0 3 2 (2) 1 3 3 (3) 1 2 4 4 (1) 3 7 (1) 6 6 (1) 7 5 (0) 0 2 0 (1) 1 1 (1) 0	Case 1: 3 critical links 0 - 1 3 - 4 6 - 7 Case 2: 0 critical links Case 3: 1 critical links 0 - 1

Notes

Dataset is huge, use faster I/O methods.

Ant Hills

1 seconds 64 MB Medium LOJ-1063 Debug

English

After many years of peace, an ant-war has broken out.

In the days leading up to the outbreak of war, the ant-government devoted a large number of resources towards gathering intelligence on ant hills. It discovered the following:

1. The ant empire has a large network of ant-hills connected by bidirectional tracks.
2. It is possible to send a message from any ant hill to any other ant hill.

Now you want to stop the war. Since they sometimes attack your house and disturb you quite a lot. So, you have made a plan. You have a gun which can destroy exactly one ant-hill. So, you want to hit an ant hill if it can stop at least two other ant hills passing messages between them. Now you want the total number of ant hills you may choose to fire.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each test case contains a blank line and two integers **n (1 ≤ n ≤ 10000), m (1 ≤ m ≤ 20000)**. **n** denotes the number of ant hills and **m** denotes the number of bi-directional tracks. Each of the next **m** lines will contain two different integers **a b (1 ≤ a, b ≤ n)** denoting that there is a track between **a** and **b**.

Output

For each case, print the case number and the total number of ant hills you may choose to fire.

Input	Output
2 5 4 2 1 1 3 5 4 4 1 3 3 1 2 2 3 1 3	Case 1: 2 Case 2: 0

LightOJ

Real Life Traffic

2 seconds 64 MB Medium Hard LOJ-1291 Debug

Statement Submissions Statistics Tutorial

English

Dhaka city is full of traffic jam and when it rains, some of the roads become unusable. So, you are asked to redesign the traffic system of the city such that if exactly one of the roads becomes unusable, it's still possible to move from any place to another using other roads.

You can assume that Dhaka is a city containing some places and bi directional roads connecting the places and it's possible to go from any place to another using the roads. There can be at most one road between two places. And of course there is no road that connects a place to itself. To be more specific there are **n** places in Dhaka city and for simplicity, assume that they are numbered from **0** to **n-1** and there are **m** roads connecting the places.

Your plan is to build some new roads, but you don't want to build a road between two places where a road already exists. You want to build the roads such that if any road becomes unusable, there should be an alternate way to go from any place to another using other roads except that damaged road. As you are a programmer, you want to find the minimum number of roads that you have to build to make the traffic system as stated above.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers: **n (3 ≤ n ≤ 10000)** and **m (≤ 20000)**. Each of the next **m** lines contains two integers **u v (0 ≤ u, v < n, u ≠ v)** meaning that there is a bidirectional road between place **u** and **v**. The input follows the above constraints.

Output

For each case, print the case number and the minimum number of roads you have to build such that if one road goes down, it's still possible to go from any place to another.

Input	Output
2 4 3 1 2 2 3 2 0 3 3 1 2 2 0 0 1	Case 1: 2 Case 2: 0

Notes

1. Dataset is huge, use faster I/O methods.
2. For case 1, one of the solutions is to construct two roads in (0, 1) and (1, 3).

LightOJ

Odd Personality

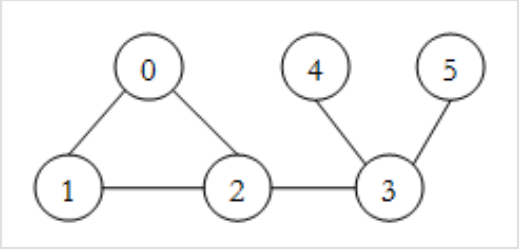
1 seconds 64 MB Medium Hard LOJ-1300 Debug

Statement Submissions Statistics Tutorial

English

Being an odd person, Jim always liked odd numbers. One day he was visiting his home town which consists of **n** places and **m** bidirectional roads. A road always connects two different places and there is at most one road between two places. The places are numbered from **0** to **n-1**.

Jim wants to find a tour which starts from a place **p** and each time it goes to a new **road** and finally at the last step it returns back to **p**. As Jim likes odd numbers, he wants the length of the tour to be odd. And the length of a tour is defined by the number of roads used in the tour.



For the city map given above, **0 - 1 - 2 - 0** is such a tour, so, **0** is one of the results, since from **0**, a tour of odd length is present. Similarly, **1 - 2 - 0 - 1** is also a valid tour. But **3 - 2 - 0 - 1 - 2 - 3** is not. since the road **2 - 3** is used twice.

Now given the city map, Jim wants to find the number of places where he can start his journey for such a tour. As you are the best programmer in town, he asks you for help. Jim can use a place more than once, but a road can be visited at most once in the tour.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers: **n (3 ≤ n ≤ 10000)** and **m (0 ≤ m ≤ 20000)**. Each of the next **m** lines contains two integers **u v (0 ≤ u, v < n, u ≠ v)** meaning that there is a bidirectional road between place **u** and **v**. The input follows the above constraints. And no road is reported more than once.

Output

For each case, print the case number and the total number places where Jim can start his journey.

Input	Output
1 6 6 0 1 1 2 2 0 3 2 3 4 3 5	Case 1: 3

Notes

Dataset is huge, user faster I/O methods.

Ant Network

2 seconds 64 MB Medium Hard LOJ-1308 Debug

English

Ants made a large underground network for their communication, consisting of **n** junctions, and junctions are connected by **m** underground tunnels. As there are animals/insects that can attack them outside the ground, they made the full network under the ground such that it becomes a safe hideout for them.

The junctions and tunnels are strong but if there is any kind of natural disasters like earthquakes or tornadoes, there is a chance that a junction may collapse. That's why they want to built some escape shafts in junctions (at most one shaft in one junction) that lead them to the surface. Now they want to build minimum number of shafts such that if any of the junctions (only one) collapses, ants that survive the collapse, still have a path to the surface. Now your task is to find the minimum number of shafts, and the number of ways the minimum shafts can be built.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers: **n (2 ≤ n ≤ 10000)** and **m (0 ≤ m ≤ 20000)**. Each of the next **m** lines contains two integers **u v (0 ≤ u, v < n, u ≠ v)** meaning that there is a bidirectional tunnel between junction **u** and **v**. The input follows the above constraints. And no tunnel is reported more than once. All junctions are connected.

Output

For each case, print the case number, the minimum number of escape shafts and the number of ways they can build the minimum shafts modulo 2^{64}.

Input	Output
2 6 6 0 3 0 1 1 4 4 2 2 5 5 0 5 4 2 1 1 3 0 4 4 1	Case 1: 2 4 Case 2: 3 1

Notes

Dataset is huge, use faster I/O methods.

Reduce the Maintenance Cost

3 seconds 64 MB Hard LOJ-1439 Debug

English

There are **n** towns (numbered from **1** to **n**) in a strange city and the maintenance costs of the towns are not necessarily same. There are bi-directional roads in the city and a road connects two towns. The merchants use the roads to trade amongst towns. If a road becomes unusable for some reason, it may affect the trading between some towns.

So, the mayor of the city has planned that every road will be maintained by one of its connecting towns. That means if a road connects town **a** and **b**, then either **a** or **b** (not both) will be given the responsibility to maintain the road.

The cost of maintaining a road is = **P** * **L**. Here:

- **P** = The number of town pairs that cannot trade if this road is damaged,
- **L** = the length of the road

If a town is given the responsibility to maintain some roads then the maintenance cost of the town will be increased by the summation of maintenance costs of the roads it'd be maintaining. Now, you are given the information of the towns, your task is to assign the roads to towns such that the maximum maintenance cost of a town is as small as possible.

Input

Input starts with an integer **T (T ≤ 30)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n (2 ≤ n ≤ 10000)** and **m (0 ≤ m ≤ 20000)**, where **n** denotes the number of towns and **m** denotes the number of roads respectively.

The next line contains **n** space separated integers between **1** and **10000** (inclusive), where the i-th integer denotes the maintenance cost of the i-th town.

Each of the next **m** lines contains three integers **u v w (1 ≤ u, v ≤ n, 1 ≤ w ≤ 10000, u != v)**, denoting that there is a road between town **u** and **v** whose length is **w**. There is at most one road between two towns.

Output

For each case, print the case number and the result.

Input	Output
3 2 1 5 10 1 2 10 6 6 10 20 30 40 50 60 1 2 1 2 3 1 1 3 1 1 4 6 1 5 6 4 6 2 3 1 10 20 30 2 3 10	Case 1: 15 Case 2: 80 Case 3: 30

Notes

- Dataset is huge, use faster I/O methods.
- For case 2, if the road between 1 and 4 goes down the town pairs that will be affected are: (1, 4), (1, 6), (2, 4), (2, 6), (3, 4), (3, 6), (4, 5), (5, 6).

Back to Underworld

2 seconds 64 MB Easy LOJ-1009 Debug

Statement Submissions Statistics Tutorial

English

The Vampires and Lykans are fighting each other to death. The war has become so fierce that, none knows who will win. The humans want to know who will survive finally. But humans are afraid of going to the battlefield.

So, they made a plan. They collected the information from the newspapers of Vampires and Lykans. They found the information about all the dual fights. Dual fight means a fight between a Lykan and a Vampire. They know the name of the dual fighters, but don't know which one of them is a Vampire or a Lykan.

So, the humans listed all the rivals. They want to find the maximum possible number of Vampires or Lykans.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case contains an integer **n (1 ≤ n ≤ 10^5)**, denoting the number of dual fights. Each of the next **n** lines will contain two different integers **u v (1 ≤ u, v ≤ 20000)** denoting there was a fight between **u** and **v**. No rival will be reported more than once.

Output

For each case, print the case number and the maximum possible members of any race.

Sample

Input	Output
2 2 1 2 2 3 3 1 2 2 3 4 2	Case 1: 2 Case 2: 3

Notes

Dataset is huge, use faster I/O methods.

Guilty Prince

1 seconds 64 MB Easy LOJ-1012 Debug

English

Once there was an Emperor named Akbar. He had a son named Jahangir. For an unforgivable reason, the king wanted him to leave the kingdom. Since he loved his son, he decided his son would be banished to a new place. The prince became sad, but he followed his father's will. On the way, he found that the place was a combination of land and water. Since he didn't know how to swim, he was only able to move on the land. He didn't know how many places might be his destination. So, he asked for your help.

For simplicity, you can consider the place as a rectangular grid consisting of some cells. A cell can be a land or can contain water. Each time the prince can move to a new cell from his current position if they share a side.

Now write a program to find the number of cells (unit land) he could reach including the cell he was initially in.

Input

Input starts with an integer **T (≤ 500)**, denoting the number of test cases.

Each case starts with a line containing two positive integers **W** and **H**; **W** and **H** are the numbers of cells in the **x** and **y** directions, respectively. **W** and **H** will not be more than 20.

There will be **H** more lines in the data set, each of which includes **W** characters. Each character represents the status of a cell as follows.

1. `.` - land.
2. `#` - water.
3. `@` - initial position of the prince (appears exactly once in a dataset).

Output

For each case, print the case number and the number of cells he can reach from the initial position (including self).

Input	Output
4 6 9 ....#. .....# .... #@...# .#..#. 11 9 .#......... .#.#######. .#.#.....#. .#.#.###.#. .#.#..@#.#. .#.#####.#. .#.......#. .#########. .... 11 6 ..#..#..#.. ..#..#..#.. ..#..#..### ..#..#..#@. ..#..#..#.. ..#..#..#.. 7 7 ..#.#.. ..#.#.. ###.### ...@... ###.### ..#.#.. ..#.#..	Case 1: 45 Case 2: 59 Case 3: 6 Case 4: 13

LightOJ

A Toy Company

1 seconds 64 MB Medium LOJ-1039 Debug

Statement Submissions Statistics Tutorial

English

The toy company "Babies Toys" has hired you to help develop educational toys. The current project is a word toy that displays three letters at all times. Below each letter are two buttons that cause the letter above to change to the previous or next letter in alphabetical order. So, with one click of a button the letter 'c' can be changed to a 'b' or a 'd'. The alphabet is circular, so for example an 'a' can become a 'z' or a 'b' with one click.

In order to test the toy, you would like to know if a word can be reached from some starting word, given one or more constraints. A constraint defines a set of forbidden words that can never be displayed by the toy. Each constraint is formatted like "X X X", where each X is a string of lowercase letters. A word is defined by a constraint if the i^{th} letter of the word is contained in the ith X of the constraint. For example, the constraint "lf a tc" defines the words "lat", "fat", "lac" and "fac".

You will be given a string start, a string finish, and some forbidden strings. Calculate and return the minimum number of button presses required for the toy to show the word finish if the toy was originally showing the word start. Remember, the toy must never show a forbidden word. If it is impossible for the toy to ever show the desired word, return **-1**.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case begins with a blank line and two strings in two lines, start and finish both having exactly three characters each.

The next line contains an integer **n (1 ≤ n ≤ 50)** denoting the number of forbidden words. Each of the next **n** lines will contain three strings each, separated by a single space. Each string (all the three strings) in a line will contain only distinct letters. Remember that start or finish can be forbidden. You may assume that all the characters are lowercase.

Output

For each case of input you have to print the case number and desired result.

Input	Output
3 aab zna 8 a a a a a z a z a z a a a z z z a z z z a z z z aaa aaa 0 aab nnn 1 a a ab	Case 1: 15 Case 2: 0 Case 3: -1

LightOJ

Rider

1 seconds 64 MB Medium LOJ-1046 Debug

Statement Submissions Statistics Tutorial

English

A rider is a fantasy chess piece that can jump like a knight several times in a single move. A rider that can perform a maximum of **K** jumps during a single move is denoted as a **K**-rider. For example, a 2-rider can jump once or twice during a single move, and a 1-rider is a traditional knight.

There are some riders of different types on a chessboard. You are given a 2D board representing the layout of the pieces. The $\mathbf{j}^{th}$ character of the $\mathbf{i}^{th}$ element of board is the content of the square at row **i**, column **j**. If the character is a digit **K** between **'1'** and **'9'**, the square contains a **K**-rider. Otherwise, if the character is a **'.'**, the square is empty. Find the minimal total number of moves necessary to move all the riders to the same square. Only one piece can move during each move. Multiple riders can share the same squares all times during the process. Print **-1** if it is impossible.

A traditional knight has up to **8** moves from a square with coordinates **(x, y)** to squares **(x+1, y+2)**, **(x+1, y-2)**, **(x+2, y+1)**, **(x+2, y-1)**, **(x-1, y+2)**, **(x-1, y-2)**, **(x-2, y+1)**, **(x-2, y-1)**, and can't move outside the chessboard.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case begins with a blank line and two integers **m, n (1 ≤ m, n ≤ 10)** denoting the rows and the columns of the board respectively. Each of the next **m** lines will contain **n** integers each denoting the board.

Output

For each case of input you have to print the case number the desired result.

Input	Output
4 3 2 .. 2. .. 3 3 1.1 ... ..1 10 10 .... .2....2... ......2... 1......... ...2.1.... ...1...... ....21. .... 1 4 1..1	Case 1: 0 Case 2: 4 Case 3: 14 Case 4: -1

LightOJ

One Way Roads

1 seconds 64 MB Medium LOJ-1049 Debug

Statement Submissions Statistics Tutorial

English

Now-a-days the one-way traffic is introduced all over the world in order to improve driving safety and reduce traffic jams. The government of Dhaka Division decided to keep up with new trends. Formerly all **n** cities of Dhaka were connected by **n** two-way roads in the ring, i.e. each city was connected directly to exactly two other cities, and from each city it was possible to get to any other city. Government of Dhaka introduced one-way traffic on all **n** roads, but it soon became clear that it's impossible to get from some of the cities to some others. Now for each road is known in which direction the traffic is directed at it, and the cost of redirecting the traffic. What is the smallest amount of money the government should spend on the redirecting of roads so that from every city you can get to any other?

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a blank line and an integer **n (3 ≤ n ≤ 100)** denoting the number of cities (and roads). Next **n** lines contain description of roads. Each road is described by three integers a_i, b_i, c_i **(1 ≤** a_i, b_i **≤ n,** $a_i \neq b_i$, **1 ≤** c_i **≤ 100)** - road is directed from city a_i to city b_i, redirecting the traffic costs c_i.

Output

For each case of input you have to print the case number and the smallest amount of money the government should spend on the redirecting of roads so that from every city you can get to any other.

LightOJ

Input	Output
4 3 1 3 1 1 2 1 3 2 1 3 1 3 1 1 2 5 3 2 1 6 1 5 4 5 3 8 2 4 15 1 6 16 2 3 23 4 6 42 4 1 2 9 2 3 8 3 4 7 4 1 5	Case 1: 1 Case 2: 2 Case 3: 39 Case 4: 0

Going Together

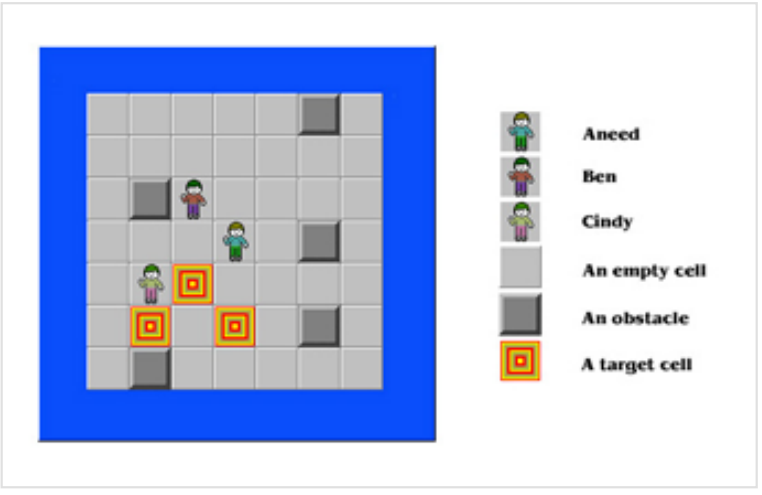
1 seconds 64 MB Medium Hard LOJ-1055 Debug

Statement Submissions Statistics Tutorial

English

You are playing a computer game in which three robots (Aneed, Ben and Cindy) are trapped in a labyrinth. Initially all three are situated in three different locations in the maze. There are three outlets through which the robots have to exit. As expected, there are several obstacles in the maze and the robots can't go through them.

The maze can be modeled as a square grid with **N x N** cells. The robots are placed on three different cells into the maze. You can command them to move. A single command will be activated for the three robots simultaneously. A robot will move to a new position (whatever its current situation is) if it is an empty cell within the maze or it is one of the free target cells, otherwise the command will be ignored for that robot. Your task is to command them such a way that all of them are on three exit cells (in any order).



A move consists of one of the following (Each move takes 1 unit of time):

Move North	The robots move one cell to the north
Move East	The robots move one cell to the east
Move South	The robots move one cell to the south
Move West	The robots move one cell to the west

Each cell consists of one of the following characters:

1. `A` - Initial position of Aneed.
2. `B` - Initial position of Ben.
3. `C` - Initial position of Cindy.
4. `.` - An empty cell.
5. `#` - An obstacle.
6. `X` - A target cell.

You can assume that for every maze each of the letters **["A", "B", "C"]** will appear exactly once and the letter **X** will appear exactly three times.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with an integer **N (2 < N < 10)**. Each of the next **N** lines contains **N** characters that fill up the maze.

Output

For each case, output the case number followed by the minimum time required. If it is impossible to move them as described, print `trapped` instead of the time.

Sample

Input	Output

3 7 .....#. ....#B.... ...A.#. .CX.... .X.X.#. .#..... 3 ABC ... XXX 3 ABC ### XXX	Case 1: 2 Case 2: 2 Case 3: trapped

LightOJ

Gathering Food

1 seconds 64 MB Medium LOJ-1066 Debug

Statement Submissions Statistics Tutorial

English

Winter is approaching! The weather is getting colder and days are becoming shorter. The animals take different measures to adjust themselves during this season.

- Some of them "migrate." This means they travel to other places where the weather is warmer.
- Few animals remain and stay active in the winter.
- Some animals "hibernate" for all of the winter. This is very deep sleep. The animal's body temperature drops, and its heartbeat and breathing slow down. In the fall, these animals get ready for winter by eating extra food and storing it as body fat.

For this problem, we are interested in the 3rd example and we will be focusing on 'Yogi Bear'.

Yogi Bear is in the middle of some forest. The forest can be modeled as a square grid of size **N x N**. Each cell of the grid consists of one of the following.

- `.` represents an empty space
- `#` represents an obstacle
- `[A-Z]` represents an English alphabet

There will be at least 1 alphabet and all the letters in the grid will be distinct. If there are **k** letters, then it will be from the first **k** alphabets. Suppose **k = 3**, that means there will be exactly one **A**, one **B** and one **C**.

The letters actually represent foods lying on the ground. Yogi starts from position **'A'** and sets off with a basket in the hope of collecting all other foods. Yogi can move to a cell if it shares an edge with the current one. For some superstitious reason, Yogi decides to collect all the foods in order. That is, he first collects **A**, then **B**, then **C**, and so on until he reaches the food with the highest alphabet value. Another philosophy he follows is that if he lands on a particular food he must collect it.

Help Yogi to collect all the foods in a minimum number of moves so that he can have a long sleep in the winter.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case contains a blank line and an integer **N (0 < N < 11)**, the size of the grid. Each of the next **N** lines contains **N** characters each.

Output

For each case, output the case number first. If it's impossible to collect all the food, output `Impossible`. Otherwise, print the shortest distance.

Input	Output
4 5 A.... ####. ..B.. .#### C.DE. 2 AC .B 2 A# #B 3 A.C ##. B..	Case 1: 15 Case 2: 3 Case 3: Impossible Case 4: Impossible

LightOJ

Farthest Nodes in a Tree

2 seconds 64 MB Easy LOJ-1094 Debug

Statement Submissions Statistics Tutorial

English

Given a tree (a connected graph with no cycles), you have to find the farthest nodes in the tree. The edges of the tree are weighted and undirected. You have to find two nodes in the tree whose distance is maximum amongst all nodes.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with an integer **n (2 ≤ n ≤ 30000)** denoting the total number of nodes in the tree. The nodes are numbered from **0** to **n-1**. Each of the next **n-1** lines will contain three integers **u v w (0 ≤ u, v < n, u ≠ v, 1 ≤ w ≤ 10000)** denoting that node **u** and **v** are connected by an edge whose weight is **w**. You can assume that the input will form a valid tree.

Output

For each case, print the case number and the maximum distance.

Sample

Input	Output
2 4 0 1 20 1 2 30 2 3 50 5 0 2 20 2 1 10 0 3 29 0 4 50	Case 1: 100 Case 2: 80

Notes

Dataset is huge, use faster I/O methods.

Best Picnic Ever

1 seconds 64 MB Medium LOJ-1111 Debug

Statement Submissions Statistics Tutorial

English

K people are having a picnic. They are initially in **N** cities, conveniently numbered from **1** to **N**. The roads between cities are connected by **M** one-way roads (no road connects a city to itself).

Now they want to gather in the same city for their picnic, but (because of the one-way roads) some people may only be able to get to some cities. Help them by figuring out how many cities are reachable by all of them, and hence are possible picnic locations.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with three integers **K (1 ≤ K ≤ 100), N (1 ≤ N ≤ 1000), M (1 ≤ M ≤ 10000)**. Each of the next **K** lines will contain an integer **(1 to N)** denoting the city where the i^{th} person lives. Each of the next **M** lines will contain two integers **u v (1 ≤ u, v ≤ N, u ≠ v)** denoting there is a road from **u** to **v**.

Output

For each case, print the case number and the number of cities that are reachable by all of them via the one-way roads.

Sample

Input	Output
1 2 4 4 2 3 1 2 1 4 2 3 3 4	Case 1: 2

LightOJ

Filling the Regions

1 seconds 64 MB Hard LOJ-1115 Debug

Statement Submissions Statistics Tutorial

English

Given a model of a map in a 2D grid, you have to color the map satisfying the following constraints:

1. A map contains one or more regions. Each region is identified by an uppercase English letter L_i. In the grid, there will be some cells containing L_i to form the region. From any cell of that region, it's possible to go to all cells (in that region) using adjacent moves. Adjacent move means going to a cell which shares a side with the current cell and belongs to the same region.
2. A region **Q** may be surrounded by another region **P**. In such case, **Q** is called a sub-region of **P** and **Q** may be erased from the map. Mark all the cells that are surrounded by P with P's identifier letter.
3. If two cells of a region share a corner, then there will always be at least one cell which shares sides with both the cells.

Now your job is to report the updated map after filling the regions.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with two integers **m** and **n (5 ≤ m, n ≤ 50)** denoting the number of rows and columns respectively. Each of the next **m** lines contains **n** characters each. Each character will be either a `.` or any uppercase English letter. `.` means empty place, and letters represent regions as described above. You can assume that the input data satisfies the above constraints.

Output

For each case, print the case number first. Print **m** lines, each line with **n** characters showing the final grid after erasing sub-regions and filling the surrounding cells.

Input	Output
2 5 5 AAAEE ABAHE A.AHF AAAHF .G... 20 20 ....B................ ..BBBB.............. ..B..BBB............ .BB....B............ BBB..RRBBBBBBBBB... BBB..RR....SS.BBB... .BB..RR....SSBB..... ..B..RR.DD.BBB...... .BB..KMDDD.B........ ..BB.KDD...B..BBB... ...B.KK....BBBBBB... ..BB.KK....BBBCBB... ...B.KK...BB.BCB.... ...B.K..BBB..BCBB... ...BBBBBB....BBBB... ...BBBBBBB....BBB... ....BB... ....BB.. ....B..	Case 1: AAAEE AAAHE AAAHF AAAHF .G... Case 2: ....B................ ..BBBB.............. ..BBBBBB............ .BBBBBBB............ BBBBBBBBBBBBBBBB... BBBBBBBBBBBBBBBB... .BBBBBBBBBBBBB..... ..BBBBBBBBBBB...... .BBBBBBBBBB........ ..BBBBBBBBB..BBB... ...BBBBBBBBBBBBB... ..BBBBBBBBBBBBBB... ...BBBBBBBBB.BBB.... ...BBBBBBBB..BBBB... ...BBBBBB....BBBB... ...BBBBBBB....BBB... ....BB... ....BB.. ....B..

LightOJ

Number Transformation

1 seconds 64 MB Medium LOJ-1141 Debug

Statement Submissions Statistics Tutorial

English

In this problem, you are given an integer number **s**. You can transform any integer number **A** to another integer number **B** by adding **x** to **A**. This **x** is an integer number which is a prime factor of **A** (please note that 1 and **A** are not being considered as a factor of **A**). Now, your task is to find the minimum number of transformations required to transform **s** to another integer number **t**.

Input

Input starts with an integer **T (≤ 500)**, denoting the number of test cases.

Each case contains two integers: **s (1 ≤ s ≤ 100)** and **t (1 ≤ t ≤ 1000)**.

Output

For each case, print the case number and the minimum number of transformations needed. If it's impossible, then print `-1`.

Sample

Input	Output
2 6 12 6 13	Case 1: 2 Case 2: -1

LightOJ

Digit Dancing

4 seconds 64 MB Hard LOJ-1165 Debug

Statement Submissions Statistics Tutorial

English

Digits like to dance. One day, 1, 2, 3, 4, 5, 6, 7 and 8 stand in a line to have a wonderful party. Each time, a male digit can ask a female digit to dance with him, or a female digit can ask a male digit to dance with her, as long as their sum is a prime. Before every dance, exactly one digit goes to who he/she wants to dance with - either to its immediate left or immediate right.

For simplicity, we denote a male digit x by itself x, and denote a female digit x by -x. Suppose the digits are in order {1, 2, 4, 5, 6, -7, -3, 8}. If -3 wants to dance with 4, she must go either to 4's left, resulting {1, 2, -3, 4, 5, 6, -7, 8} or his right, resulting {1, 2, 4, -3, 5, 6, -7, 8}.

Note that -3 cannot dance with 5, since their sum 3+5 = 8 is not a prime; 2 cannot dance with 5, since they're both male.

Given the initial ordering of the digits, find the minimal number of dances needed for them to sort in increasing order (ignoring signs of course).

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains exactly eight integers in a single line. The absolute values of these integers form a permutation of **{1, 2, 3, 4, 5, 6, 7, 8}**.

Output

For each test case, print the case number and the minimal number of dances needed. If they can never be sorted in increasing order, print **-1**.

Sample

Input	Output
5 1 2 4 5 6 -7 -3 8 1 2 3 4 5 6 7 8 1 2 3 5 -4 6 7 8 1 2 3 5 4 6 7 8 2 -8 -4 5 6 7 3 -1	Case 1: 1 Case 2: 0 Case 3: 1 Case 4: -1 Case 5: 3

Commandos

1 seconds 64 MB Medium LOJ-1174 Debug

Statement Submissions Statistics Tutorial

English

A group of commandos were assigned a critical task. They are to destroy an enemy headquarter. The enemy head quarter consists of several buildings and the buildings are connected by roads. The commandos must visit each building and place a bomb at the base of each building. They start their mission at the base of a particular building and from there they disseminate to reach each building. The commandos must use the available roads to travel between buildings. Any of them can visit one building after another, but they must all gather at a common place when their task in done. In this problem, you will be given the description of different enemy headquarters. Your job is to determine the minimum time needed to complete the mission. Each commando takes exactly one unit of time to move between buildings. You may assume that the time required to place a bomb is negligible. Each commando can carry unlimited number of bombs and there is an unlimited supply of commando troops for the mission.

Input

Input starts with an integer **T (≤50)**, denoting the number of test cases.

The first line of each case starts with a positive integer **N (1 ≤ N ≤ 100)**, where **N** denotes the number of buildings in the headquarter. The next line contains a positive integer **R**, where **R** is the number of roads connecting two buildings. Each of the next **R** lines contain two distinct numbers **u v (0 ≤ u, v < N)**, this means there is a road connecting building **u** to building **v**. The buildings are numbered from **0** to **N-1**. The last line of each case contains two integers **s d (0 ≤ s, d < N)**. Where **s** denotes the building from where the mission starts and **d** denotes the building where they must meet. You may assume that two buildings will be directly connected by at most one road. The input will be given such that, it will be possible to go from any building to another by using one or more roads.

Output

For each case, print the case number and the minimum time required to complete the mission.

Input	Output
2 4 3 0 1 2 1 1 3 0 3 2 1 0 1 1 0	Case 1: 4 Case 2: 1

LightOJ

Jane and the Frost Giants

1 seconds 64 MB Medium LOJ-1175 Debug

Statement Submissions Statistics Tutorial

English

Jane is one of the most talented young programmers as well as an astrophysicist. Recently she discovered a planet and named it Jotunheim - the world of giants. As you already guessed that the inhabitants are all giants. Among them the Frost Giants are the most evil ones. Before Jane could publicly announce her great discovery, the Frost Giants came and captured her in a maze. Since the Giants would be discovered to the universe because of her, that's why they lit fires on some positions in the maze to kill her.

You are given Jane's location in the maze and the positions of the fires lit by the Frost Giants whom are always keeping an eye on her; you must find out whether Jane can escape from the maze before fire catches her, and how fast she can do it.

The Maze is defined as a 2D grid and the locations are defined as squares. The cost of each move is one square per minute. In each move, Jane can move vertically or horizontally but not diagonally. She cannot move to a square which is blocked by an obstacle, or which is already burning. If a square has fire in it, in the next minute, fires spread to its adjacent **non-obstacle** squares (vertically or horizontally). Jane can escape from the maze from any squares that borders the edge of the maze.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

The first line of each test case contains the two integers **R** and **C**, separated by spaces, with **1 ≤ R, C ≤ 200** The following **R** lines of the test case each contain one row of the maze. Each of these lines contains exactly **C** characters, and each of these characters is one of:

1. `#`, an obstacle.
2. `.`, a free location.
3. `J`, Jane's initial position in the maze (there will be exactly one `J` in the maze).
4. `F`, position of a fire.

Output

For each case, print the case number and `IMPOSSIBLE` if Jane cannot escape from the maze before fire reaches her, or the earliest time for Jane to safely escape from the maze, in minutes.

Input	Output
2 4 5 ##.## #JF.# #...# #...# 3 3 ### #J. #.F	Case 1: 3 Case 2: IMPOSSIBLE

Escape

1 seconds 64 MB Medium Hard LOJ-1185 Debug

English

Hanzo Hattori rescued Princess Nakururu and now they are planning to escape from the castle, because it's going to explode. Now the problem is that Nakururu is not quite well that's why she can't run all the way. And Hattori is also injured. That's why he can't carry her all the way.

So, they made a plan. From their current position, Hanzo will carry Nakururu to the next place. Then they both will run to the next place. Then again Hanzo will carry Nakururu to the next place. And they will continue this procedure to go from one place to another.

Each place can be treated as a node. Nodes are connected by bi-directional roads. All the nodes are numbered as integers. Initially they are at node **1**. And both of them will run to the next node.

Now given the description of the nodes and roads you have to count the number of possible nodes which can be entered by Hanzo carrying Nakururu with him. Initially Hattori doesn't carry Nakururu.

Input

Input starts with an integer **T (≤ 210)**, denoting the number of test cases.

Each case starts with a blank line. The next line will contain two integers: **n (1 ≤ n ≤ 100)** and **m (0 ≤ m ≤ n*(n-1)/2)**, indicating the number of nodes and the number of roads respectively. The next **m** lines, each will contain two integers **a b (1 ≤ a, b ≤ n, a ≠ b)** indicating that there is a road between node **a** and **b**. All the given roads will be distinct.

Output

For each case, print the case number and number of nodes that can be entered by Hanzo, carrying Nakururu with him.

LightOJ

Input	Output
2 4 3 1 2 2 3 3 4 5 4 1 2 2 3 1 3 3 4	Case 1: 2 Case 2: 4

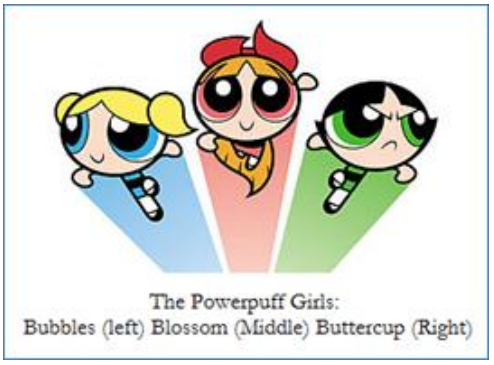
LightOJ

Power Puff Girls

1 seconds 64 MB Medium LOJ-1238 Debug

Statement Submissions Statistics Tutorial

English


The Powerpuff Girls:
Bubbles (left) Blossom (Middle) Buttercup (Right)

The city of Townsville! This nice city is the home for the power puff girls - Blossom, Bubbles, and Buttercup. To introduce their personality we can sing a song:

Blossom, commander and the leader;
Bubbles, she is the joy and the laughter;
Buttercup, she is the toughest fighter.

These super girls defend Townsville from monsters and supervillains using their superpowers, intelligence. They live in a home in Townsville with the professor who is like their father.

The girls are young now and they don't like to fight the monsters anymore. So, if they are outside their home, they are found shopping. And when they get back home, they simply watch TV shows. It's such a horrible fact that the super-intelligent girls are wasting their time watching TV serials that consist of the rivalries between Wives and their Mother in Laws. And when they use computers they are usually found using the face note (a dangerous[1] social networking site).

So, such wonderful girls just became lazy and useless. Often they are seen fighting each other, the comments that can be heard, are like, 'Tulsi is the best.', 'Gopi is better than Rashee.' The professor is quite upset with the girls, and he can't even watch any science show or sports because of these irritating serials.

So, the professor made a plan and asked the girls to go shopping such that he can watch an important science show on the TV. The girls became very excited and they went out shopping. But soon they realize that one of their favorite serials will start soon, and they need to get back home for that serial.

To be more specific let's consider the city as a 2D grid. In the grid there are some symbols, the meaning of them are:

- `.` means an empty place.
- `a` denotes the position of Blossom.
- `b` denotes the position of Bubbles.
- `c` denotes the position of Buttercup.
- `m` denotes that there is a monster.
- `h` denotes their home.
- `#` denotes a wall and the girls cannot pass through it.

The three girls move simultaneously. And in each minute, from their current cells, they can move to any four adjacent cells (North, East, West, South) if the destination cell is neither a wall nor the cell contains a monster. Because they want to get home as soon as possible, they want to avoid the monsters. You can assume that they can move to a common cell if necessary.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers: **m** and **n (4 ≤ m, n ≤ 20)**, **m** denotes the number of rows and **n** denotes the number of columns of the modeled grid. Each of the next **m** lines contains **n** characters representing the city.

You can assume that `a`, `b`, `c`, `h` will occur exactly once in the grid. The outer boundaries will always be marked by `#`.

Output

For each case, print the case number and the minimum time needed when they all are in their home. You can assume that a solution will always exist.

Sample

Input	Output
2 6 8 ######## #...c..# #......# #.a.h.b# #......# ######## 5 9 ######### #mmm...c# #ma.h#### #m....b.# #########	Case 1: 2 Case 2: 4

Farthest Nodes in a Tree (II)

2 seconds 64 MB Medium LOJ-1257 Debug

English

Given a tree (a connected graph with no cycles), you have to find the cost to go to the farthest node from each node. The edges of the tree are weighted and undirected.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with an integer **n (2 ≤ n ≤ 30000)** denoting the total number of nodes in the tree. The nodes are numbered from **0** to **n-1**. Each of the next **n-1** lines will contain three integers **u v w (0 ≤ u, v < n, u ≠ v, 1 ≤ w ≤ 10000)** denoting that node **u** and **v** are connected by an edge whose weight is **w**. You can assume that the input will form a valid tree.

Output

For each case, print the case number in a line first. Then for each node (from **0** to **n - 1**) print the cost to go to the farthest node in a separate line.

Sample

Input	Output
2 4 0 1 20 1 2 30 2 3 50 5 0 2 20 2 1 10 0 3 29 0 4 50	Case 1: 100 80 50 100 Case 2: 50 80 70 79 80

Notes

Dataset is huge, use faster I/O methods.

Equalizing Money

1 seconds 64 MB Medium LOJ-1263 Debug

English

There are **n** people in a certain village. Each of them contains some amount of money. One day a wise person told them to distribute the money such that everyone has equal amount of money. If they can do so, they will be favored by their fortunes.

You are given the information about the money of each person and some relations. Each relation is of the form **u v**. That means person **u** and **v** are capable of making money transactions. They are allowed to use transactions any number of times but they have to do integer transactions only.

Now your task is to answer whether they can redistribute the money such that each of them contains exactly same of money.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing an integer **n (2 ≤ n ≤ 1000)** and **m (0 ≤ m ≤ 10000)** where **m** denotes the number of relations. The next line contains **n** space separated integers ranging from **0** to **1000**. The i^{th} integer of this line denotes the money for the i^{th} person. Each of the next **m** lines contains two integers **u v (1 ≤ u, v ≤ n, u ≠ v)** meaning that person **u** and **v** can make transactions. No relation is reported more than once.

Output

For each case, print the case number and `Yes` if they can equalize their money or `No` otherwise.

Input	Output
3 5 4 1 0 1 1 2 1 2 2 3 3 4 4 5 2 1 5 10 1 2 4 2 1 1 0 2 1 2 2 3	Case 1: Yes Case 2: No Case 3: No

Notes

Dataset is huge, use faster I/O methods.

LightOJ

Better Tour

2 seconds 64 MB Medium Hard LOJ-1271 Debug

Statement Submissions Statistics Tutorial

English

Alice and Bob like to travel. One day Alice went for a tour, and after coming back she described his tour to Bob. Bob got quite excited hearing the adventures Alice made. So, he planned to make a tour like Alice. As Alice told him that in Alice's path the two most interesting places were the first city and the last city she visited. So, Bob tries to make a tour that contains minimum number of cities and which starts from the city where Alice started her journey, and which finishes in the city where Alice finished her journey.

But the problem is that Bob doesn't have the map, so, he doesn't know the information of the roads between cities. But he has Alice's diary, so he figured out the cities Alice visited in order. It's known that there are bidirectional roads connecting the cities. And each city is identified by an integer between **1** and **50000**.

Now Bob has given you the name of the cities visited by Alice in order, your task is to make a tour plan for Bob that visits **least** number of cities. Since there can be many such solutions, you have to find the tour which is lexicographically smallest. For example, let a tour be $c_1, c_2 \dots c_m$ and another tour be $d_1, d_2 \dots d_m$, (c_i, d_i denote cities), then the first tour is lexicographically smaller if

- $c_1 < d_1$, or
- $c_1 = d_1$ and $c_2 < d_2$, or
- $c_1 = d_1$ and $c_2 = d_2$ and $c_3 < d_3$, or
- ...

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing an integer **n (2 ≤ n ≤ 50000)**. The next line contains **n** space separated integers denoting the tour made by Alice. Each of these integers will lie in the range **[1, 50000]**. Two adjacent integers: **u v** means that Alice moved to city **v** from city **u**. The starting city and the ending city will always be different. And two adjacent integers will also be different.

Output

For each case, print the case number in a single line. The next line should contain the tour plan for Bob. Two integers in this line should be separated by a single space.

Input	Output
2 6 1 2 3 4 1 3 5 4 2 6 3 1	Case 1: 1 3 Case 2: 4 2 6 3 1

Notes

Dataset is huge, use faster I/O methods.

LightOJ

The Crystal Maze

1 seconds 64 MB Medium LOJ-1337 Debug

Statement Submissions Statistics Tutorial

English

You are in a plane and you are about to be dropped with a parasuit in a crystal maze. As the name suggests, the maze is full of crystals. Your task is to collect as many crystals as possible.

To be more exact, the maze can be modeled as an **M x N** 2D grid where **M** denotes the number of rows and **N** denotes the number of columns. There are three types of cells in the grid:

1. A **'#'** denotes a wall, you may not pass through it.
2. A **'C'** denotes a crystal. You may move through the cell.
3. A **'.'** denotes an empty cell. You may move through the cell.

Now you are given the map of the maze, you want to find where to land such that you can collect maximum number of crystals. So, you are spotting some position **x, y** and you want to find the maximum number of crystals you may get if you land to cell **(x, y)**. And you can only move vertically or horizontally, but you cannot pass through walls, or you cannot get outside the maze.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with a line containing three integers **M**, **N** and **Q (2 ≤ M, N ≤ 500, 1 ≤ Q ≤ 1000)**. Each of the next **M** lines contains **N** characters denoting the maze. You can assume that the maze follows the above restrictions.

Each of the next **Q** lines contains two integers x_i and y_i **($1 \le x_i \le M$, $1 \le y_i \le N$)** denoting the cell where you want to land. You can assume that cell **(x_i, y_i)** is empty i.e. the cell contains **'.'**.

Output

For each case, print the case number in a single line. Then print **Q** lines, where each line should contain the maximum number of crystals you may collect if you land on cell **(x_i, y_i)**.

Input	Output
1 4 5 2 ..#.. .C#C. ##..# ..C#C 1 1 4 1	Case 1: 1 2

Notes

Dataset is huge, use faster I/O methods.

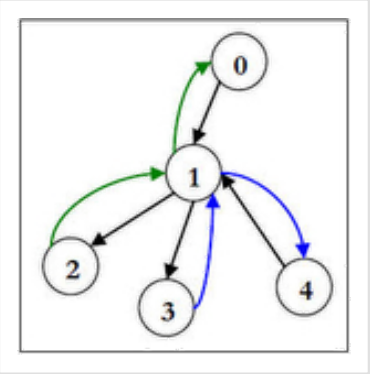
LightOJ

Paths in a Tree

1 seconds 64 MB Hard LOJ-1353 Debug

Statement Submissions Statistics Tutorial

English



You are given a tree (a connected graph with no cycles), and then the edges in the tree are made directional; your task is to add **minimum** number of special paths in the tree such that it's possible to go from one node to another. The rules for the special paths are noted below:

1. A special path starts in a node, consists of some continuous edges and nodes and also ends in a node.
2. In a special path, the edges should be in opposite directions as they are in the tree.
3. A node or an edge can be visited at most once in a special path.
4. Multiple special paths may have common nodes or edges.

For example, in the given picture, a tree is drawn, the black arrows represent the edges and their directions, circles represent nodes. Then we need two special paths. One path is **2-1-0** (green arrow), another is **3-1-4** (blue arrow). Another valid option is to use paths **3-1-4** and **2-1-4**. You cannot add a path like **1-3** or **0-1-2** because of rule 2. You cannot add **0-2** or **2-3-0** because of rule 1.

Input

Input starts with a positive integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing an integer **N (2 ≤ N ≤ 20000)**, where **N** denotes the number of nodes. The nodes are numbered from **0** to **N-1**. Each of the next **N-1** lines contains two integers **u v (0 ≤ u, v < N, u ≠ v)** meaning that there is an edge from **u** to **v**.

Output

For each case, print the case number and the minimum number of special paths required such that it's possible to go from any node to another.

Input	Output
2 4 0 1 1 2 1 3 5 0 1 1 2 1 3 0 4	Case 1: 2 Case 2: 3

Notes

Dataset is huge, use faster I/O methods.

Corrupted Friendship

1 seconds 64 MB Medium Hard LOJ-1357 Debug

English

Many of us know a person named Mr. Haba Kom Khan. He is very corrupted and lazy. He maintains links with a lot of (corrupted) persons. In order to corrupt more lazy persons, he invites his friends in a recursive fashion in a meeting. The idea is:

If **X** has some cards, he keeps one card and he invites one of his friends **Y** who has no invitation cards yet and **X** gives all the remaining cards to **Y**. Then we say that **Y** is invited by **X**. **Y** then has the responsibility to distribute as many cards as he can; **Y** keeps one card and continues the same procedure as **X**. When **Y** cannot find any more friends to give the invitation cards, he gives the remaining cards back to **X**. **X** then checks for his friends who have no cards yet. If any such friend is found, **X** continues the same procedure. Otherwise, if **X** was invited by **Z** then **X** gives the remaining cards back to **Z**. **X** keeps the cards if there is no such person.

Initially Mr. Haba has **N** invitations cards, and he starts the invitation process as stated. There are **N** persons, and they are numbered from **1** to **N**. Mr. Haba is the person numbered **1**. And the strange fact is that, after the invitation process, each person gets exactly **1** card.

Given all the information of persons being invited by others, Mr. Haba wants you to find the total number of invitations that was made. He also asks you to find the number of different pairs of persons who are certainly not friends. Help Mr. Haba Kom Khan to succeed in his corrupted life!

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with an integer **N (1 ≤ N ≤ 10^5)**. Each of the next **N-1** lines will contain two integers, **X** and **Y (1 ≤ X, Y ≤ N, X ≠ Y)** denoting that person **Y** received his invitation card from person **X**.

Output

For each case, print the case number, total number of invitations made, and the number of different pairs of persons who are surely not friends. See samples for detailed formatting.

Input	Output
2 2 1 2 3 1 2 1 3	Case 1: 1 0 Case 2: 2 1

Notes

Dataset is huge, use faster I/O methods.

LightOJ

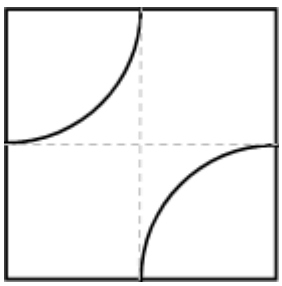
Truchet Tiling

3 seconds 64 MB Hard LOJ-1368 Debug

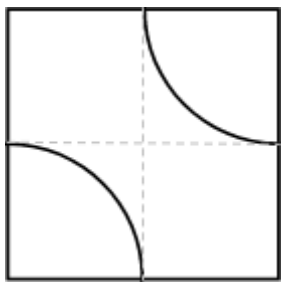
Statement Submissions Statistics Tutorial

English

In 1704, French mathematician Sebastien Truchet proposed a tiling system that started with simple motifs and used its rotations and random placement to create quite interesting tiling patterns. In this problem we will calculate the area enclosed by the curves in a special case of his tiling system.



Motif 0

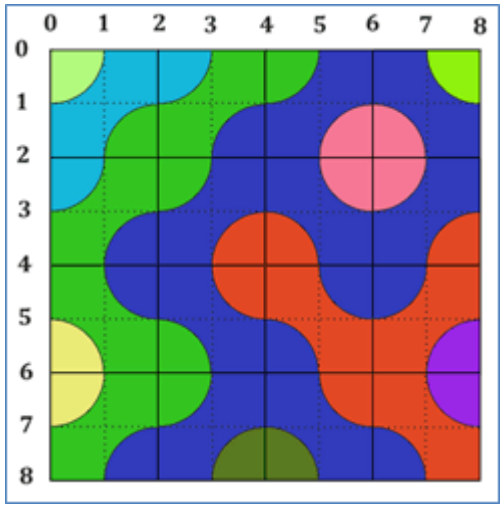


Motif 1

Observe the two motifs here. Motif 0 is a square of length 2 units on each side. There are two circles with radius 1 unit drawn at two opposite corners of the square. Motif 1 is just a 90 degrees rotation of the first one (or you can think of it as drawing the circles on the other two corners).

Using just two basic motifs as shown here, we can tile an area to create rather interesting artistic patterns. They do not have to be symmetrically placed; we can lay out these motifs randomly to cover an area. If we then use a tool like paint bucket (found in most paint programs) at a particular point in the pattern, we can color a contiguous region with one color.

In this problem you'll be given a description of such a tiling pattern and then be asked how much area would it color if we use paint bucket starting at specific points.



Here is one illustration: We have the motifs randomly placed on a **9 x 9** grid. Then we used a blue color at position (2, 4) (here 2 denotes the distance from the top and 4 denotes the distance from left). It colored the entire contiguous region it found bounded by the grid's boundary and the curves themselves. We could achieve the same effect if we used the color at position (4, 6), (0, 6), (1, 7) etc. However, using the paint bucket on the perimeter of the curves (such as (3, 4)) will only color the perimeter line of the curves. They will not fill up a region with any color.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers **R** and **C (1 ≤ R, C ≤ 100)**. Each of the next **R** lines contains the row description given in the binary form. The basic motifs are numbered **0** and **1**. For example, the accompanying picture's first row can be described as `0001` .The next line contains an integer **Q (1 ≤ Q ≤ 100)** denoting the number of queries. Each query will be of the form **x y (0 ≤ x ≤ 2R, 0 ≤ y ≤ 2C)**, where **x** and **y** are integers denoting the row and column number where the paint bucket tool will be used.

Output

For each case, print the case number in a single line. Then for each query, print one number giving the area enclosed by the boundary and the curves that contain that point. If the point in question itself lies on a curve, you can assume the enclosed area to be zero. Errors less than 10^{-6} will be ignored.

Sample

Input	Output
3 1 2 01 4 0 0 2 0 0 1 0 2 2 2 01 00 1 2 2 3 1 1 0 1 2 3 1 4 2	Case 1: 0.7853981634 4.8584073464 0 4.8584073464 Case 2: 4.7853981634 Case 3: 7.2876110196 1.5707963268

Notes

This is a special judge problem; wrong output format may cause `wrong answer` .

Blade and Sword

2 seconds 64 MB Medium Hard LOJ-1377 Debug

Statement Submissions Statistics Tutorial

English

You may have played the game 'Blade and Sword', it's an action game. However, in this problem we are actually solving one stage of the game.

For simplicity, assume that the game stage can be modeled as a **2D** grid which has **m** rows and **n** columns. The cells are categorized as follows:

- `.` represents an empty space. The player can move through it.
- `#` represents wall, and the player cannot move through it. You can assume that the boundaries of the grid will be walls.
- `*` means a teleporting cell, once the player moves into this cell, he must choose **any other** teleporting cell where he will be taken to. But if he cannot find a desired teleporting cell, he will die. However, after moving to the desired teleporting cell, he can either move to an adjacent cell, or he can teleport again using the same procedure.
- `P` means the position of the player and there will be exactly one cell containing `P`.
- `D` means the destination cell and there will be exactly one cell containing `D`.

Now you are given a stage and the player starts moving. It takes one unit of time for the player to move to any adjacent cell from his current position. Two cells are adjacent if they share a side. One unit of time is needed for the teleporting service; that means taking the player from one teleporting cell to any other teleporting cell. Your task is to find the minimum possible time unit required for the player to reach the destination cell.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers: **m** and **n (3 ≤ m, n ≤ 200)**. Each of the next **m** lines contains **n** characters denoting the stage. The given stage follows the restrictions stated above.

Output

For each case, print the case number and the minimum required time. If it's impossible for the player to reach the destination cell, print `impossible`. See the samples for details.

Input	Output
2 4 10 ########## #.P..#*..# #*......D# ########## 3 9 ######### #P.#..D.# #########	Case 1: 6 Case 2: impossible

Visiting Islands

2 seconds 64 MB Medium Hard LOJ-1412 Debug

English

There are **N** islands and **M** bridges. All the bridges are setup between two islands and to pass a bridge you have to give a toll of **$1**. The bridges are built in such a way that there is no more than one path among two islands. Now, you want to visit at least **K** different islands. You may choose the starting island of your choice, but you want to visit at least **K** different islands in minimum cost (starting island is considered to be already visited).

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with two integers **N, M (1 ≤ N ≤ 10^5, 0 ≤ M < N)**. Each of the next **M** lines contains two integers **u v (1 ≤ u, v ≤ N, u ≠ v)** meaning that there is a bridge between island **u** and **v**. No bridge will be reported more than once.The next line contains an integer **q (1 ≤ q ≤ 50000)** denoting the number of queries. Each of the next **q** lines contains one integer **K (1 ≤ K ≤ 10^5)**.

Output

For each case, print the case number first. Then for each query, print the minimum amount of toll you need to pay to visit at least **K** different islands. If it is not possible, print `impossible` .

Input	Output
2 2 1 1 2 3 1 2 3 5 4 1 2 2 3 2 4 2 5 2 3 2	Case 1: 0 1 impossible Case 2: 2 1

Notes

1. Dataset is huge, use faster I/O methods.
2. For the first case, for **K = 1**, which ever island we start with, we visit this. So without giving any toll we can visit one island. For **K = 2**, we choose island 1 to start. So we visit island 2 using the only bridge. So it costs $1. For **K = 3**, as there are only 2 islands in total so we cannot visit 3 islands.

LightOJ

Blind Escape

2 seconds 64 MB Hard LOJ-1426 Debug

Statement Submissions Statistics Tutorial

English

Illidan Stormage, the blind Warcraft hero is thrown to a maze for punishment. Being a blind hero, he cannot see anything but he can sense which direction is north. He memorized the map of the maze when he was a child. But he doesn't know his exact location.

Assume that the maze is laid out on a grid, and each grid location is either blocked or free. He has four possible moves: **North, South, East** and **West**. He wants to find the shortest sequence of moves that will guarantee his escape (his initial location can be any free cell in the grid). He is said to be escaped if he is outside the maze and once he is out of the maze, further moves are irrelevant. And of course if he tries to walk into a wall, he will simply stay in the same spot.

As he cannot see anything, for any move, he will only stop if he bumps into a wall or he is out of the maze. Now, your task is to help him by finding the shortest sequence of moves.

Input

Input starts with an integer **T (≤ 40)**, denoting the number of test cases.

Each case starts with a line containing two integers **M** and **N (1 ≤ M, N ≤ 12)** denoting the number of rows and columns of the grid respectively. Each of the next **M** lines contains **N** characters (either `.` or `#`) denoting the maze. `.` means free location and `#` means a wall. Assume that there is at least one free location in the grid.

Output

For each case, print the case number first. If it's impossible to do so, print `Impossible`. Otherwise, print the shortest possible sequence of moves that guarantees his escape. Show the sequence by the first letters of the moves. As there can be many solutions, print the one that comes lexicographically earliest. See the samples for details.

Input	Output
2 4 8 ######## #...#..# ##....## ##.##### 3 4 #### #..# ####	Case 1: ESWSWS Case 2: Impossible

LightOJ

Patch Quilt

1 seconds 64 MB Hard LOJ-1434 Debug

Statement Submissions Statistics Tutorial

English

We will now try another problem inspired by one of Sam Loyd's puzzles.

A group of children made a lovely patch quilt that they gave as a gift to their teacher. Knowing their teacher is a puzzle aficionado, the kids created the quilt with a very particular feature: the names of everyone who contributed to the gift are hidden in the patchwork.

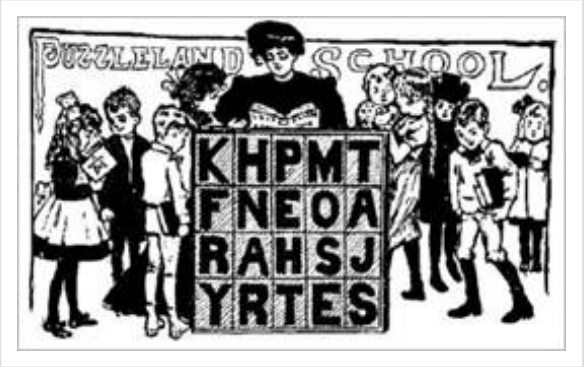


Figure 1: A present from her students

The teacher was happily surprised by the thoughtful gift and kept it in the classroom, where it is displayed and continues to pose a challenge to anyone who gazes upon it. The rules of the puzzle are simple:

1. To find any name, you can start at any position, and at each step you may move in any direction, including diagonals.
2. Any position in the puzzle can be used more than once. You can even choose not to move at some steps, to use the same letter in succession.
3. For every name that is hidden in the puzzle, there is only one valid way to find it (there are no multiple solutions for any of the names).

You know the list of names of all the students. Your task is to identify their locations inside the quilt.

Input

Input starts with an integer **T (≤ 200)**, denoting the number of test cases.

Each case starts with a line containing two integers **R** and **C (1 ≤ R, C ≤ 30)** denoting the number of rows and columns of the quild respectively. The following **R** lines contain **C** letters each, representing

the contents of the puzzle. All letters are uppercase.

The next line contains an integer **N (1 ≤ N ≤ 20)**, denoting the number of students. Each of the next **N** lines contains a name, containing uppercase letters. The length of each name will be at least **2**, and at most **15**. You can safely assume that if a certain name is found in the puzzle, there will only be one possible way to discover it.

Output

For each case, print the case number in a single line. Then, for each name given in the input, print if it is found in the puzzle or not.

If a name is found, print the message **"NAME found:"** followed by a description of its location, given in the following way:

1. First print the coordinates of the first letter in the form **(r,c)** where **1 ≤ r ≤ R** and **1 ≤ c ≤ C**.
2. Then, for each additional letter, print its location relative to the previous letter. Use the letters **U, D, L, R** for up, down, left and right respectively. For diagonals, use the following combinations: **UL** for up-left, **UR** for up-right, **DL** for down-left, and **DR** for down-right. To indicate no movement at all, use an asterisk **(*)**.
3. Make sure to use a comma and a single space between elements in the list. Notice, however, that there must be no spaces after the last move.

If a name is not found, simply print `NAME not found`. For more details on the format of the output, please refer to the sample below.

Sample

Input	Output
1 4 5 KHPMT FNEOA RAHSJ YRTES 4 JAMES JOHN HANNAH MARIE	Case 1: JAMES found: (3,5), U, UL, DL, DR JOHN found: (3,5), UL, DL, UL HANNAH found: (3,3), L, U, *, D, R MARIE not found

Beehives

4 seconds 64 MB Medium LOJ-1437 Debug

English

Bees are one of the most industrious insects. They collect nectar and pollen from the flowers and thus they rely on the trees in the forest. For better navigability, they numbered the **n** trees from **0** to **n - 1**. Instead of roaming around all over the forest, they use a predefined map consisting of some paths. A path is based on two trees, and the bees fly from one tree to another in a straight line. They don't use paths that are not on their map.

As technology has been improved a lot, they also changed their working strategy. Instead of hovering over all the trees in the forest, they are targeting particular trees, mainly trees with lots of flowers. To make things even better, they are planning to build some new hives on some targeted trees. Once done, they will only collect pollens from those trees and the unnecessary paths will be removed from their map.

Now, they want to build the hives such that if **one** of the paths in their new map become unavailable (some birds or animals interrupting them in that path), it's still possible to go from any hive to another using the other paths in their new map.

The bees don't want to choose less than two trees and they also want to build minimum number of hives. Now you are given the trees with the current map, your task is to propose a new bee hive colony for them.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n (2 ≤ n ≤ 500)** and **m (0 ≤ m ≤ 20000)**, where **n** denotes the number of trees and **m** denotes the number of paths. Each of the next **m** lines contains two integers **u v (0 ≤ u, v < n, u ≠ v)** meaning that there is a path between tree **u** and **v**. Assume that there can be at most one path between any pair of trees.

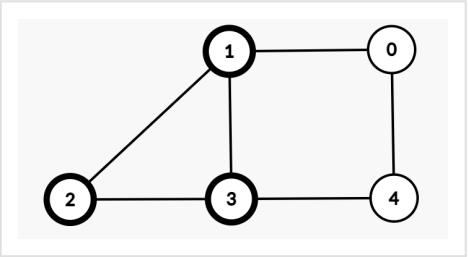
Output

For each case, print the case number and the number of beehives in the proposed colony or `impossible` if it's impossible to find such a colony.

Input	Output
3 3 3 0 1 1 2 2 0 2 1 0 1 5 6 0 1 1 2 1 3 2 3 0 4 3 4	Case 1: 3 Case 2: impossible Case 3: 3

Notes

- Dataset is huge, use faster I/O methods.
- For case 3, the bees will build 3 hives in node {1, 2, 3} and they will only maintain three roads in the colony - {(1, 2), (1, 3), (2, 3)}.



Drunk

2 seconds 128 MB Medium LOJ-1003 Debug

Statement Submissions Statistics Tutorial

English

One of my friends is always drunk. So, sometimes I get a bit confused whether he is drunk or not. So, one day I was talking to him, about his drinks! He began to describe his way of drinking. So, let me share his ideas a bit. I am expressing in my words.

There are many kinds of drinks, which he used to take. But there are some rules; there are some drinks that have some pre requisites. Suppose if you want to take wine, you should have taken soda, water before it. That's why to get real drunk is not that easy.

Now given the name of some drinks! And the prerequisites of the drinks, you have to say that whether it's possible to get drunk or not. To get drunk, a person should take all the drinks.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with an integer **m (1 ≤ m ≤ 10000)**. Each of the next **m** lines will contain two names each in the format **a b**, denoting that you must have **a** before having **b**. The names will be non-empty and contain at most **10** characters.

Output

For each case, print the case number and `Yes` or `No`, depending on whether it's possible to get drunk or not.

Sample

Input	Output
2 2 soda wine water wine 3 soda wine water wine wine water	Case 1: Yes Case 2: No

LightOJ

Hit the Light Switches

1 seconds 64 MB Medium LOJ-1034 Debug

Statement Submissions Statistics Tutorial

English

Ronju is a night-guard at the "Lavish office buildings Ltd." headquarters. The office has a large grass field in front of the building. Every day, when Ronju comes to duty in the evening, it is his duty to turn on all the lights in the field. However, given the large size of the field and a large number of lights, it is very tiring for him to walk to each light to turn it on.

After a lot of thinking, he has devised an ingenious plan - he will swap the switches for light-sensitive triggers. A local electronic store nearby sells these funny trigger switches at a very cheap price. Once installed at a light-post, it will automatically turn that light on whenever it can sense some other light lighting up nearby. So, from now on, Ronju can just manually flip a few switches, and the light from those will trigger nearby sensors, which will in turn light up some more lights nearby, and so on, gradually lighting up the whole field.

Now Ronju wonders: how many switches does he have to flip manually for this?

Input

Input starts with an integer **T (≤ 10)**,denoting the number of test cases.

Each case contains a blank line two integers **N (1 ≤ N ≤ 10000)** and **M (0 ≤ M ≤ 50000)**, where **N** is the number of lights in the field, and **M** more lines of input follows in this input case. Each of these extra **M** lines will have two different integers **a** and **b** separated by a space, where **1 ≤ a, b≤ N**, indicating that if the light **a** lights up, it will trigger the light **b** to turn on as well (according to their distance, brightness, sensor sensitivity, orientation and other complicated factors).

Output

For each case, print the case number and the minimum number of lights that Ronju must turn on manually before all the lights in the whole field gets lit up.

Input	Output
2 5 4 1 2 1 3 3 4 5 3 4 4 1 2 1 3 4 2 4 3	Case 1: 2 Case 2: 2

Wishing Snake

1 seconds 64 MB Medium Hard LOJ-1168 Debug

Statement Submissions Statistics Tutorial

English

'Wishing Snake' is a computer game for children. In this game, there is a board and a snake. The board contains 1000 checkpoints numbered from 0 to 999. Initially, the snake sits at checkpoint 0.

Now **n** children come in front of the board and start wishing. Each wish looks like - "I want to see the snake walking from checkpoint **u** to checkpoint **v**." Each child can have multiple wishes. At first, a child comes in front of the board and makes his/her wishes. After that, a new child comes and makes his new wishes (that means not wished by any children yet). And it continues until the n^{th} children.

After that, the snake starts walking from one checkpoint to another. It can only walk from one checkpoint to another if any child had wished it. While walking from checkpoint **u** to checkpoint **v**, the snake cannot cross any other checkpoints along the path.

The snake wants to fulfill all the wishes done by all the children in one single path. Since you are the lead designer of the game, you want to find whether it's possible or not.

Input

Input starts with an integer **T (≤ 65)**, denoting the number of test cases.

Each case starts with an integer **n (1 ≤ n ≤ 100)**. Then for each child an integer **k (k ≥ 0)** is given. Each of the next **k** lines contains two integers **u v (0 ≤ u, v < 1000, u ≠ v)** denoting that the child wants to see the snake going from checkpoint **u** to checkpoint **v**. You may assume that all the wishes are distinct and correct. And the total number of wishes in any case is between **1** and **10000** (inclusive).

Output

For each case, print the case number and `YES` if it's possible, otherwise print `NO`.

Input	Output
2 2 2 0 9 9 10 1 10 15 1 2 0 9 0 11	Case 1: YES Case 2: NO

LightOJ

Efficient Traffic System

1 seconds 64 MB Medium Hard LOJ-1210 Debug

Statement Submissions Statistics Tutorial

English

I was given the task to make all the major two way roads in Bangladesh into one way roads. And I have done that easily with some great pruning. And I asked the Govt. Traffic Management System to change the direction of all the roads.

But after some days, I realized that I should have thought of the fact that all cities should be reachable from other cities using the existing one way roads. Since the traffic system is already designed, so it may not be changed. But I can ask the govt. to build new roads between any pair of cities.

Now since the task looks quite hard for me, I am asking you to do it for me. I will give you the current roads configuration. You have to find the minimum number of roads that have to be built such that it's possible to go from any city to any other city.

Input

Input starts with an integer **T (≤ 25)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n (1 ≤ n ≤ 20000)** and **m (0 ≤ m ≤ 50000)**, where **n** denotes the number of cities and **m** denotes the number of one way roads. Each of the next **m** lines contains two integers **u v (1 ≤ u, v ≤ n, u ≠ v)** meaning that there is a road from **u** to **v**. Assume that there can be at most one road from a city **u** to **v**.

Output

For each case, print the case number and the minimum number of roads that have to be built.

Sample

Input	Output
2 3 0 3 2 1 2 1 3	Case 1: 3 Case 2: 2

Notes

Dataset is huge. Use faster I/O methods.

LightOJ

Weight Comparison

2 seconds 64 MB Hard LOJ-1390 Debug

Statement Submissions Statistics Tutorial

English

Rimi has **n** different objects having distinct weights and identified by $x_1, x_2, ..., x_n$. All she wanted to do was to sort them in ascending order according to their weights. That's why she asked her robot-helper to do all the weight comparisons. She went for another task, and the robot continued doing the comparisons. This robot was not that intelligent, that's why it was picking two objects randomly, compared them and wrote down the object names as x_i x_j meaning that x_i is heavier than x_j. After a while, Rimi came back and found out this foolishness of the robot. So, she wants to find the comparisons that are really necessary. For example, the robot compared and found the following:

1. $x_1 > x_2$
2. $x_2 > x_5$
3. $x_1 > x_5$

Clearly the third one is unnecessary, as from the first two relations, it's clear that x_1 is heavier than x_5. But the second one is necessary since from first and third relation, it's impossible to determine the relation between x_2 and x_5. The first relation is also necessary.

Now Rimi wants to keep minimum number of necessary relations such that they are enough to find all the comparisons.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n, m (1 ≤ n ≤ 5000, 0 ≤ m ≤ 10^5)**, where **m** denotes the number of comparisons done by the robot. Each of the next **m** lines contains two integers **i j (1 ≤ i, j ≤ n, i ≠ j)** meaning that x_i is heavier than x_j. There is no duplicate transition. And assume that the data is valid.

Output

For each case, print the case number and the number of necessary comparisons. Then print the comparisons in **i j** (x_i is heavier than x_j) form, first sorted by **i** then by **j** in ascending order.

Input	Output
2 4 4 1 2 3 4 2 3 1 4 4 1 4 2	Case 1: 3 1 2 2 3 3 4 Case 2: 1 4 2

Notes

Dataset is huge; use faster I/O methods.

Beehives

4 seconds 64 MB Medium LOJ-1437 Debug

English

Bees are one of the most industrious insects. They collect nectar and pollen from the flowers and thus they rely on the trees in the forest. For better navigability, they numbered the **n** trees from **0** to **n - 1**. Instead of roaming around all over the forest, they use a predefined map consisting of some paths. A path is based on two trees, and the bees fly from one tree to another in a straight line. They don't use paths that are not on their map.

As technology has been improved a lot, they also changed their working strategy. Instead of hovering over all the trees in the forest, they are targeting particular trees, mainly trees with lots of flowers. To make things even better, they are planning to build some new hives on some targeted trees. Once done, they will only collect pollens from those trees and the unnecessary paths will be removed from their map.

Now, they want to build the hives such that if **one** of the paths in their new map become unavailable (some birds or animals interrupting them in that path), it's still possible to go from any hive to another using the other paths in their new map.

The bees don't want to choose less than two trees and they also want to build minimum number of hives. Now you are given the trees with the current map, your task is to propose a new bee hive colony for them.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n (2 ≤ n ≤ 500)** and **m (0 ≤ m ≤ 20000)**, where **n** denotes the number of trees and **m** denotes the number of paths. Each of the next **m** lines contains two integers **u v (0 ≤ u, v < n, u ≠ v)** meaning that there is a path between tree **u** and **v**. Assume that there can be at most one path between any pair of trees.

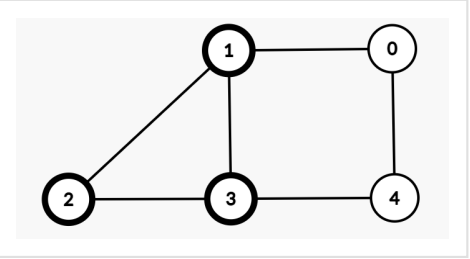
Output

For each case, print the case number and the number of beehives in the proposed colony or `impossible` if it's impossible to find such a colony.

Input	Output
3 3 3 0 1 1 2 2 0 2 1 0 1 5 6 0 1 1 2 1 3 2 3 0 4 3 4	Case 1: 3 Case 2: impossible Case 3: 3

Notes

- Dataset is huge, use faster I/O methods.
- For case 3, the bees will build 3 hives in node {1, 2, 3} and they will only maintain three roads in the colony - {(1, 2), (1, 3), (2, 3)}.



Assassin's Creed (II)

2 seconds 64 MB Hard LOJ-1429 Debug

Statement Submissions Statistics Tutorial

English

Ezio needs to kill **N** targets located in **N** different cities. The cities are connected by some one way roads. As time is short, Ezio can send a massage along with the map to the assassin's bureau to send some assassins who will start visiting cities and killing the targets. An assassin can start from any city, he/she may visit any city multiple times even if the cities that are already visited by other assassins.

Ezio wants to find the minimum number of assassins needed to do the job.

Input

Input starts with an integer **T (≤ 70)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **N (1 ≤ N ≤ 1000)** and **M(0 ≤ M ≤ 10000)**, where **N** denotes the number of cities and **M** denotes the number of one way roads.

Each of the next **M** lines contains two integers **u v (1 ≤ u, v≤ N, u ≠ v)** meaning that there is a road from **u** to **v**. Assume that there can be at most one road from a city **u** to **v**.

Output

For each case, print the case number and the minimum number of assassins needed to kill all targets.

Input	Output
3 5 4 1 2 1 3 4 1 5 1 7 0 8 8 1 2 2 3 3 4 4 1 1 6 6 7 7 8 8 6	Case 1: 2 Case 2: 7 Case 3: 2

Notes

Dataset is huge, use faster I/O methods.

LightOJ

Forwarding Emails

3 seconds 64 MB Medium Hard LOJ-1417 Debug

Statement Submissions Statistics Tutorial

English

"... so forward this to ten other people, to prove that you believe the emperor has new clothes." Aren't those sorts of emails annoying?

Martians get those sorts of emails too, but they have an innovative way of dealing with them. Instead of just forwarding them willy-nilly, or not at all, they each pick one other person they know to email those things to every time - exactly one, no less, no more (and never themselves). Now, the Martian clan chieftain wants to get an email to start going around, but he stubbornly only wants to send one email. Being the chieftain, he managed to find out who forwards emails to whom, and he wants to know: which Martian should he send it to maximize the number of Martians that see it?

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing an integer **N (2 ≤ N ≤ 50000)** denoting the number of Martians in the community. Each of the next **N** lines contains two integers: **u v (1 ≤ u, v ≤ N, u ≠ v)** meaning that Martian **u** forwards email to Martian **v**.

Output

For each case, print the case number and an integer **m**, where **m** is the Martian that the chieftain should send the initial email to. If there is more than one correct answer, output the smallest number.

Input	Output
3 3 1 2 2 3 3 1 4 1 2 2 1 4 3 3 2 5 1 2 2 1 5 3 3 4 4 5	Case 1: 1 Case 2: 4 Case 3: 3

LightOJ

Assassin's Creed

4 seconds 64 MB Medium Hard LOJ-1406 Debug

Statement Submissions Statistics Tutorial

English

Altair is in great danger as he broke the three tenets of the assassin's creed. The three tenets are: 1) never kill innocent people, 2) always be discrete and 3) never compromise the brotherhood. As a result, Altair is given another chance to prove that he is still a true assassin. Altair has to kill **n** targets located in **n** different cities. Now as time is short, Altair can send a massage along with the map to the assassin's bureau to send some assassins who will start visiting cities and killing the targets. An assassin can start from any city, but cannot visit a city which is already visited by any other assassin except him (because they do not like each other's work). He can visit a city multiple times though. Now Altair wants to find the minimum number of assassins needed to kill all the targets. That's why he is seeking your help.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains two integers **n (1 ≤ n ≤ 15)** and **m (0 ≤ m ≤ 50)**, where **n** denotes the number of cities and **m** denotes the number of one way roads. Each of the next **m** lines contains two integers **u v (1 ≤ u, v ≤ n, u ≠ v)** meaning that there is a road from **u v**. Assume that there can be at most one road from a city **u** to **v**.

Output

For each case, print the case number and the minimum number of assassins needed to kill all the targets.

Input	Output
2 3 2 1 2 2 3 6 6 1 2 2 3 2 4 5 4 4 6 4 2	Case 1: 1 Case 2: 2

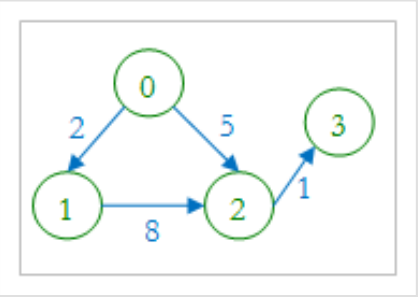
LightOJ

Teleport

1 seconds 64 MB Hard LOJ-1380 Debug

Statement Submissions Statistics Tutorial

English



You are in a tour and want to visit all the **n** cities in a country which are numbered from **0** to **n - 1**. Initially, you are in city **K**. You have a car and can drive to one city to another. And you have a teleporting device, too. It can take you to any place to another place in no time. But the problem is that, the teleporting machine only works if the place you want to go is already visited by you. It can't detect the co-ordinates of a new city. But when you visit a city, the device locates the co-ordinates carefully, and thus helps you to teleport to this city from any other place. And the device is so strong that you can even teleport using the device along with your car! Since the traffic of the country is quite high, the Govt. has planned to use only **m** unidirectional roads.

You are given the map of the city and the estimated driving duration for each the roads, you have to find the minimum possible time needed for you to visit all the cities.

For example, for the city in the picture, you are in city 0. At first you go to city 1, and it will take 2 units of time, then you can go to city 2 from city 1, but it will take 8 units of time. You can teleport to city 0 and then go to city 2, it will cost you 5. So, you can visit city 1 and 2 in 7 units of time. And you can visit all the cities in time unit 8.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains three integers: **n m K (1 ≤ n ≤ 1000, 0 ≤ m ≤ 10000, 0 ≤ K < n)**. Each of the next **m** lines contains three integers **u v w (0 ≤ u, v < n, u ≠ v, 1 ≤ w ≤ 10000)** meaning that there is a road from city **u** to city **v** and takes **w** units of time to drive that road. There can be at most one road from a city **u** to another city **v**.

Output

For each case, print the case number and the minimum time needed to visit all the cities. If it's impossible to do so, print `impossible`.

Input	Output
2 4 4 0 0 1 2 1 2 8 0 2 5 2 3 1 2 1 1 0 1 10	Case 1: 8 Case 2: impossible

Notes

Dataset is Huge, use faster I/O Methods.

LightOJ

Stream My Contest

1 seconds 64 MB Hard LOJ-1384 Debug

Statement Submissions Statistics Tutorial

English

During 2009 and 2010 ICPC world finals, the contests were webcasted via World Wide Web. Seeing this, some contest organizers from Ajobdesh decided that, they will provide a live stream of their contests to every university in Ajobdesh. The organizers have decided that, they will provide best possible service to them. But there are two problems:

1. There is no existing network between universities. So, they need to build a new network. However, the maximum amount they can spend on building the network is **C**.
2. Each link in the network has a bandwidth. If, the stream's bandwidth exceeds any of the link's available bandwidth, the viewers, connected through that link can't view the stream.

Due to the protocols used for streaming, a viewer can receive stream from exactly one other user (or the server, where the contest is organized). That is, if you have two 128kbps links, you won't get 256kbps bandwidth, although, if you have a stream of 128kbps, you can stream to any number of users at that bandwidth.

Given **C**, find the maximum possible bandwidth they can stream.

Input

Input starts with an integer **T (≤ 35)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains three integers **N, M, C (1 ≤ N ≤ 60, 1 ≤ M ≤ 10^4, 1 ≤ C ≤ 10^9)**, the number of universities and the number of possible links, and the budget for setting up the network respectively. Each university is identified by an integer between **0** and **N-1**, where **0** denotes the server.

Each of the next **M** lines contains four integers **u, v, b, c (0 ≤ u, v < N, 1 ≤ b, c ≤ 10^6)**, describing a possible link from university **u** to university **v**, that has the bandwidth of **b** kbps and of cost **c**. All links are unidirectional. There can be multiple links between two universities.

Output

For each case, print the case number and the maximum possible bandwidth to stream. If it's not possible, print `impossible` . See the samples for details.

Input	Output
3 3 4 300 0 1 128 100 1 2 256 200 2 1 256 200 0 2 512 300 3 4 500 0 1 128 100 1 2 256 200 2 1 256 200 0 2 512 300 3 4 100 0 1 128 100 1 2 256 200 2 1 256 200 0 2 512 300	Case 1: 128 kbps Case 2: 256 kbps Case 3: impossible

LightOJ

Jogging Trails

1 seconds 64 MB Medium Hard LOJ-1086 Debug

Statement Submissions Statistics Tutorial

English

Robin is training for a marathon. Behind his house is a park with a large network of jogging trails connecting water stations. Robin wants to find the shortest jogging route that travels along every trail at least once.

For each case, there should be one line of output giving the length of Robin's jogging route.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains two positive integers **n (2 ≤ n ≤ 15)**, the number of water stations, and **m (0 ≤ 1000)**, the number of trails. For each trail, there is one subsequent line of input containing three positive integers: the first two, between **1** and **n**, indicating the water stations at the endpoints of the trail; the third indicates the length of the trail, in cubits. There may be more than one trail between any two stations; each different trail is given only once in the input; each trail can be traveled in either direction. It is possible to reach any trail from any other trail by visiting a sequence of water stations connected by trails. Robin's route may start at any water station and must end at the same station.

Output

For each case, print the case number and the minimum possible length of Robin's jogging route.

Sample

Input	Output
1 4 5 1 2 3 2 3 4 3 4 5 1 4 10 1 3 12	Case 1: 41

Village Postman

1 seconds 64 MB Hard LOJ-1250 Debug

English

I postman has to deliver post to villagers who live in several villages and in the roads that connect the villages. Actually there are houses in villages as well as along the sides of the roads. There are **n** villages and **m** roads. Villages are numbered from **1** to **n**, and all the villages are connected by some paths. A path consists of some connected roads. The postman starts his journey from village 1 and he must finish his journey at village 1. So, he wants to find a tour that visits each village, each road at least once. Since all the villagers want the postman to come in their house as early as possible; they designed a cost service.

If an unvisited village **i** is visited as the **k^{th}** (1 indexed) **different** village on the tour and **k ≤ w(i)**, the village pays **w(i) - k** taka to the post. However, if **k > w(i)**, the post agrees to pay **k - w(i)** taka to the village. Moreover, the post has to spend one taka for each road the postman visits. Even if the postman travels a road **p** times, then the post has to spend **p** taka. The villages are established in a way that such a tour is always possible. And there are always 2, 4 or 8 roads going out from each village. Now your task is to help the post to find a route that maximizes the earning (or minimizes the loss).

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing two integers **n (1 ≤ n ≤ 200)** and **m**. The next line contains **n** space separated integers. The **i^{th}** integer denotes the **w(i)** for the **i^{th}** village **(0 ≤ w(i) ≤ 1000)**. Each of the next **m** lines contains two integers **u v (1 ≤ u, v ≤ n)** denoting that there is a road between village **u** and **v**. You can assume that the given input follows the constraints given above.

Output

For each case, print the case number and the maximum profit. Then in the next line print the route. So, this line should contain numbers of consecutive villages on the route $v_1\ v_2\ \ldots\ v_x$ separated by single spaces, with $v_1 = v_x = 1$. There can be several solutions, any valid one will do.

Input	Output
1 6 7 0 7 4 10 20 5 2 4 1 5 2 1 4 5 3 6 1 6 1 3	Case 1: 18 1 5 4 2 1 6 3 1

Notes

This is a special judge problem, wrong output format may cause **'wrong answer'**.

Word Puzzle

1 seconds 64 MB Hard LOJ-1256 Debug

English

You are given a puzzle board where there are **n** words. Initially they are scattered. Your task is to find a order of the words such that the first character of **i^{th}** word is same as the last character of the **$(i-1)^{th}$** word **(1 < i ≤ n)**.

For example, you are given **{"abef", "pqrs", "fzzp", "zama", "pxrp"}**, the solution is - **{"zama", "abcf", "fzzp", "pxrp", "pqrs"}**.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a line containing two an integer **n (1 ≤ n ≤ 1000)**. Each of the next **n** lines contains a word whose length is between **1** and **20** (inclusive). You can assume that the words contain lowercase letters only.

Output

For each case, print the case number and `Yes` if such a ordering can be possible, or `No` if there is no such ordering.

If the result is yes, then in the next line you should print the **n** words in a valid order. Print a single space between two consecutive words. There can be multiple solutions, any valid one will do.

Sample

Input	Output
2 5 abcf pqrs fzzp zama pxrp 3 yes no notsolvable	Case 1: Yes zama abcf fzzp pxrp pqrs Case 2: No

Notes

This is a special judge problem. Wrong output format maycause wrong answer.

Aladdin and the Return Journey

3 seconds 64 MB Medium LOJ-1348 Debug

Statement Submissions Statistics Tutorial

English

Finally the Great Magical Lamp was in Aladdin's hand. Now he wanted to return home. But he didn't want to take any help from the Genie because he thought that it might be another adventure for him. All he remembered was the paths he had taken to reach there. But since he took the lamp, all the genies in the cave became angry and they were planning to attack. As Aladdin was not afraid, he wondered how many genies were there. He summoned the Genie from the lamp and asked this.

Now you are given a similar problem. For simplicity assume that, you are given a tree (a connected graph with no cycles) with **n** nodes, nodes represent places, edges represent roads. In each node, initially there are an arbitrary number of genies. But the numbers of genies change in time. So, you are given a tree, the number of genies in each node and several queries of two types. They are:

1. `0 i j`, it means that you have to find the total number of genies in the **nodes** that occur in path from node **i** to **j** **(0 ≤ i, j < n)**
2. `1 i v`, it means that number of genies in node **i** is changed to **v** **(0 ≤ i < n, 0 ≤ v ≤ 1000)**.

Input

Input starts with an integer **T** **(≤ 5)**, denoting the number of test cases.

Each case starts with a blank line. Next line contains an integer **n** **(2 ≤ n ≤ 30000)**. The next line contains **n** space separated integers between **0** and **1000**, denoting the number of genies in the nodes respectively. Then there are **n-1** lines each containing two integers: **u v** **(0 ≤ u, v < n, u ≠ v)** meaning that there is an edge from node **u** and **v**. Assume that the edges form a valid tree. Next line contains an integer **q** **(1 ≤ q ≤ 10^5)** followed by **q** lines each containing a query as described above.

Output

For each case, print the case number in a single line. Then for each query **0 i j**, print the total number of genies in the nodes that occur in path **i** to **j**.

Input	Output
1 4 10 20 30 40 0 1 1 2 1 3 3 0 2 3 1 1 100 0 2 3	Case 1: 90 170

Notes

Dataset is huge, use faster I/O methods.

LightOJ

Trux it Up

3 seconds 128 MB Hard LOJ-1445 Debug

Statement Submissions Statistics Tutorial

English

The **Dinotrux** are facing the biggest and baddest challenge of all time. D-Structs and his gang are coming to attack the colony so Ty and Revvit are planning to fight back. Since they have a big colony to maintain; they are planning to shrink it such that they have better focus on the rest.

They live in a colony where there are **n** rest stops, which are connected by **m** bidirectional roads. Since they built the rest stops and also strengthened the roads time to time, they know which roads are strong and which are not.

Building rest stops are hard but roads are easy. So, Ty and Revvit are planning to do the following two things:

1. Destroy as many roads as possible but all the rest stops should still be reachable (directly or indirectly) from one another.
2. They also want to destroy in such a way that the summation of the strengths of the remaning roads is as strong as possible.

Since there could be multiple solutions, they are seeking your help to figure out how many ways they could do the destruction.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a blank line followed by two integers **n** and **m (2 ≤ n ≤ 200, m ≥ 0)**. Each of the next **m** lines will contain three integers **u v w (1 ≤ u, v ≤ n, u ≠ v, 0 < w ≤ 15000)** denoting a road between rest stop **u** and **v** with strength **w**. A stronger road means higher **w**. There will be at most one road between any two rest stops.

Output

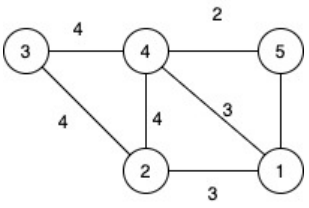
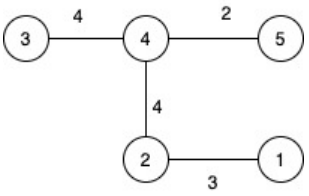
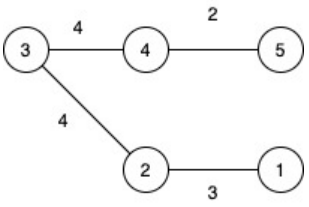
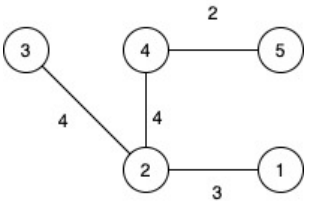
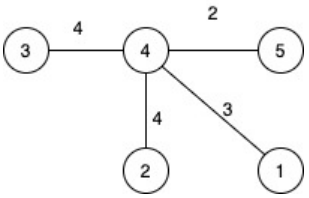
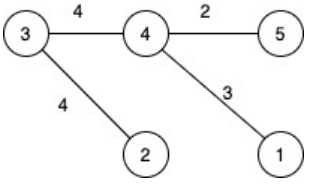
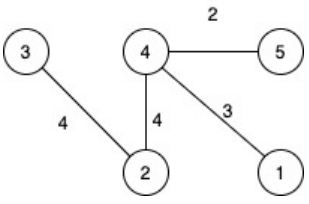
For each case, print the case number and the number of ways Ty and Revvit can destroy the roads. Since the result could be very large, print the result modulo **1000210433** (which is a prime).

Input	Output
3 4 3 1 4 1 2 3 1 3 4 1 5 7 4 5 2 2 4 4 5 1 1 1 2 3 2 3 4 3 4 4 4 1 3 3 0	Case 1: 1 Case 2: 6 Case 3: 0

Notes

1. For case 1, no road can be destroyed. So, we have one solution.

2. For case 2, the 6 solutions are below:

	Roads Destroyed	View
Initial map	-	
Solution 1	(5 1) (4 1) (2 3)	
Solution 2	(5 1) (4 1) (2 4)	
Solution 3	(5 1) (4 1) (3 4)	
Solution 4	(5 1) (1 2) (2 3)	
Solution 5	(5 1) (1 2) (2 4)	
Solution 6	(5 1) (1 2) (3 4)	

3. For case 3, the rest stops are not reachable from one another. Thus, we don't have any solution.

LightOJ

Air Ports

1 seconds 64 MB Medium LOJ-1059 Debug

Statement Submissions Statistics Tutorial

English

The government of a certain developing nation wants to improve transportation in one of its most inaccessible areas, in an attempt to attract investment. The region consists of several important locations that must have access to an airport.

Of course, one option is to build an airport in each of these places, but it may turn out to be cheaper to build fewer airports and have roads link them to all of the other locations. Since these are long distance roads connecting major locations in the country (e.g. cities, large villages, industrial areas), all roads are two-way. Also, there may be more than one direct road possible between two areas. This is because there may be several ways to link two areas (e.g. one road tunnels through a mountain while the other goes around it etc.) with possibly differing costs.

A location is considered to have access to an airport either if it contains an airport or if it is possible to travel by road to another location from there that has an airport.

You are given the cost of building an airport and a list of possible roads between pairs of locations and their corresponding costs. The government now needs your help to decide on the cheapest way of ensuring that every location has access to an airport. The aim is to make airport access as easy as possible, so if there are several ways of getting the minimal cost, choose the one that has the most airports.

Input

Input starts with an integer **T (≤ 15)**, denoting the number of test cases.

Each case starts with three integers **N, M** and **A (0 < N ≤ 10000, 0 ≤ M ≤ 100000, 0 < A ≤ 10000)** separated by white space. **N** is the number of locations, **M** is the number of possible roads that can be built, and **A** is the cost of building an airport.

The following **M** lines each contain three integers **X, Y** and **C (1 ≤ X, Y ≤ N, 0 < C ≤ 10000)**, separated by white space. **X** and **Y** are two locations, and **C** is the cost of building a road between **X** and **Y**.

Output

For each case, print the case number and **Y Z**, where **Y** is the minimum cost of making roads and airports so that all locations have access to at least one airport, and **Z** is the number of airports to be built. As mentioned earlier, if there are several answers with minimal cost, choose the one that maximizes the number of airports.

Input	Output
2 4 4 100 1 2 10 4 3 12 4 1 41 2 3 23 5 3 1000 1 2 20 4 5 40 3 2 30	Case 1: 145 1 Case 2: 2090 2

Notes

The input file is large; make sure your I/O code is fast.

Civil and Evil Engineer

1 seconds 64 MB Medium LOJ-1029 Debug

English

A Civil Engineer is given a task to connect **n** houses with the main electric power station directly or indirectly. The Govt has given him permission to connect exactly **n** wires to connect all of them. Each of the wires connects either two houses, or a house and the power station. The costs for connecting each of the wires are given.

Since the Civil Engineer is clever enough and tries to make some profit, he made a plan. His plan is to find the best possible connection scheme and the worst possible connection scheme. Then he will report the average of the costs.

Now you are given the task to check whether the Civil Engineer is evil or not. That's why you want to calculate the average before he reports to the Govt.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case contains a blank line and an integer **n (1 ≤ n ≤ 100)** denoting the number of houses. You can assume that the houses are numbered from **1** to **n** and the power station is numbered **0**. Each of the next lines will contain three integers in the form **u v w (0 ≤ u, v ≤ n, 0 < w ≤ 10000, u ≠ v)** meaning that you can connect **u** and **v** with a wire and the cost will be **w**. A line containing three zeroes denotes the end of the case. You may safely assume that the data is given such that it will always be possible to connect all of them. You may also assume that there will not be more than **12000** lines for a case.

Output

For each case, print the case number and the average as described. If the average is not an integer then print it in **p/q** form. Where **p** is the numerator of the result and **q** is the denominator of the result; **p** and **q** are relatively-prime. Otherwise print the integer average.

Input	Output
3 1 0 1 10 0 1 20 0 0 0 3 0 1 99 0 2 10 1 2 30 2 3 30 0 0 0 2 0 1 10 0 2 5 0 0 0	Case 1: 15 Case 2: 229/2 Case 3: 15

LightOJ

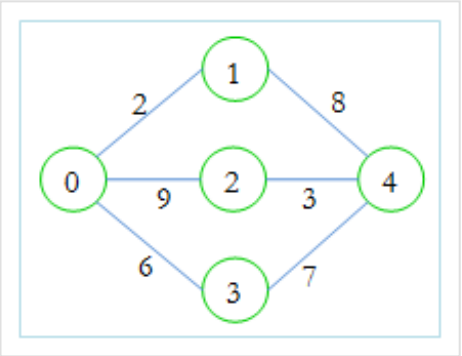
Country Roads

1 seconds 64 MB Easy LOJ-1002 Debug

Statement Submissions Statistics Tutorial

English

I am going to my home. There are many cities and many bi-directional roads between them. The cities are numbered from **0** to **n-1** and each road has a cost. There are **m** roads. You are given the number of my city **t** where I belong. Now from each city you have to find the minimum cost to go to my city. The cost is defined by the cost of the maximum road you have used to go to my city.



For example, in the above picture, if we want to go from 0 to 4, then we can choose

1. 0 - 1 - 4 which costs 8, as 8 (1 - 4) is the maximum road we used
2. 0 - 2 - 4 which costs 9, as 9 (0 - 2) is the maximum road we used
3. 0 - 3 - 4 which costs 7, as 7 (3 - 4) is the maximum road we used

So, our result is 7, as we can use 0 - 3 - 4.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a blank line and two integers **n (1 ≤ n ≤ 500)** and **m (0 ≤ m ≤ 16000)**. The next m lines, each will contain three integers **u, v, w (0 ≤ u, v < n, u ≠ v, 1 ≤ w ≤ 20000)** indicating that there is a road between **u** and **v** with cost **w**. Then there will be a single integer **t (0 ≤ t < n)**. There can be multiple roads between two cities.

Output

For each case, print the case number first. Then for all the cities (from **0** to **n-1**) you have to print the cost. If there is no such path, print `Impossible`.

Input	Output
2 5 6 0 1 5 0 1 4 2 1 3 3 0 7 3 4 6 3 1 8 1 5 4 0 1 5 0 1 4 2 1 3 3 4 7 1	Case 1: 4 0 3 7 7 Case 2: 4 0 3 Impossible Impossible

Notes

Dataset is huge, use faster I/O methods.

LightOJ

Donation

1 seconds 64 MB Medium LOJ-1040 Debug

Statement Submissions Statistics Tutorial

English

A local charity is trying to gather donations of Ethernet cable. You realize that you probably have a lot of extra cable in your house, and make the decision that you will donate as much cable as you can spare.

You will be given the lengths (in meters) of cables between each pair of rooms in your house. You wish to keep only enough cable so that every pair of rooms in your house is connected by some chain of cables, of any length. The lengths are given in **n** lines, each having **n** integers, where **n** is the number of rooms in your house. The j^{th}integer of i^{th} line gives the length of the cable between rooms **i** and **j** in your house.

If both the j^{th} integer of i^{th}line and the i^{th} integer of j^{th} line are greater than **0**, this means that you have two cables connecting rooms **i**and **j**, and you can certainly donate at least one of them. If the i^{th}integer of i^{th} line is greater than **0**, this indicates unused cable in room **i**, which you can donate without affecting your home network in any way. **0** means no cable.

You are not to rearrange any cables in your house; you are only to remove unnecessary ones. Return the maximum total length of cables (in meters) that you can donate. If any pair of rooms is not initially connected by some path, return **-1**.

Input

Input starts with an integer **T (≤ 100)**,denoting the number of test cases.

Each case begins with a blank line and an integer **n (1≤ n ≤ 50)** denoting the number of rooms in your house. Then there will be **n** lines, each having **n** space separated integers,denoting the lengths as described. Each length will be between **0** and **100**.

Output

For each case of input you have to print the case number and the desired result.

Input	Output
3 2 27 26 1 52 4 0 10 10 0 0 0 1 1 0 0 0 2 0 0 0 0 4 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0	Case 1: 105 Case 2: 12 Case 3: -1

Road Construction

1 seconds 64 MB Medium LOJ-1041 Debug

English

There are several cities in the country - Zen, and some bidirectional roads are connecting them. Unfortunately, some of the roads are damaged and are unusable right now. Your goal is to rebuild enough of the damaged roads so that there is a functional path between every pair of cities.

You are given the description of roads. Damaged roads are formatted as **$city_1$ $city_2$ cost"** and non-damaged roads are formatted as **"$city_1$ $city_2$ 0"**. In this notation **$city_1$** and **$city_2$** are the case-sensitive names of the two cities directly connected by that road. If the road is damaged, there is a cost of rebuilding that road. Every city in Zen will appear at least once in the given input. And there can be multiple roads between same pair of cities.

Your task is to find the minimum cost of the roads that must be rebuilt to achieve the given goal. If it is impossible to do so, print `Impossible` .

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case begins with a blank line and an integer **m (1≤ m ≤ 50)** denoting the number of roads. Then there will be **m**lines, each containing the description of a road. No names will contain morethan **50** characters. The road costs will lie in the range **[0, 1000]**.

Output

For each case of input you have to print the case number andthe desired result.

Input	Output
2 12 Dhaka Sylhet 0 Ctg Dhaka 0 Sylhet Chandpur 9 Ctg Barisal 9 Ctg Rajshahi 9 Dhaka Sylhet 9 Ctg Rajshahi 3 Sylhet Chandpur 5 Khulna Rangpur 7 Chandpur Rangpur 7 Dhaka Rajshahi 6 Dhaka Rajshahi 7 2 Rajshahi Khulna 4 Kushtia Bhola 1	Case 1: 31 Case 2: Impossible

LightOJ

Trail Maintenance

2 seconds 64 MB Medium LOJ-1123 Debug

Statement Submissions Statistics Tutorial

English

Tigers in the Sunderbans wish to travel freely among the **N** fields (numbered from **1** to **N**), even though they are separated by trees. The tigers wish to maintain trails between pairs of fields so that they can travel from any field to any other field using the maintained trails. Tigers may travel along a maintained trail in either direction.

The tigers do not build trails. Instead, they maintain deer trails that they have discovered. On any week, they can choose to maintain any or all of the deer animal trails they know about. Always curious, the tigers discover one new deer trail at the beginning of each week. They must then decide the set of trails to maintain for that week so that they can travel from any field to any other field. Tigers can only use trails which they are currently maintaining.

The tigers always want to minimize the total length of trail they must maintain. The tigers can choose to maintain any subset of the deer trails they know about, regardless of which trails were maintained the previous week. Deer trails (even when maintained) are never straight. Two trails that connect the same two fields might have different lengths. While two trails might cross, tigers are so focused; they refuse to switch trails except when they are in a field. At the beginning of each week, the tigers will describe the deer trail they discovered. Your program must then output the minimum total length of trail the tigers must maintain that week so that they can travel from any field to any other field, if there is such a set of trails.

Input

Input starts with an integer **T (≤ 25)**, denoting the number of test cases.

Each case starts with two integers **N (1 ≤ N ≤ 200)** and **W**. **W** is the number of weeks the program will cover **(1 ≤ W ≤ 6000)**.

Each of the next **W** lines will contain three integers describing the trail the tigers found that week. The first two numbers denote the end points (filed numbers) and the third number denotes the length of the trail **(1 to 10000)**. No trail has the same field as both of its endpoints.

Output

For each case, print the case number in a line. Then for every week, output a single line with the minimum total length of trail the tigers must maintain so that they can travel from any field to any other field. If no set of trails allows the tigers to travel from any field to any other field, output `-1`.

Input	Output
1 4 6 1 2 10 1 3 8 3 2 3 1 4 3 1 3 6 2 1 2	Case 1: -1 -1 -1 14 12 8

Sending Packets

2 seconds 64 MB Medium Hard LOJ-1321 Debug

English

Alice and Bob are trying to communicate through the internet. Just assume that there are **N** routers in the internet and they are numbered from **0** to **N-1**. Alice is directly connected to router **0** and Bob is directly connected to router **N-1**. Alice initiates the connection and she wants to send **S** KB of data to Bob. Data can go to the $(N-1)^{th}$ router from the 0^{th} router either directly or via some intermediate routers. There are some bidirectional links between some routers.

The links between the routers are not necessarily 100% perfect. So, for each link, a probability p_i is given. If **u** and **v** are two routers and if their underlying link has probability p_i, it means that if data is sent from **u** to **v**, the probability of successfully getting the data in **v** is p_i and vice versa. If multiple links are used, the probability of getting the data in destination is the multiplication of the probabilities of the links that have been used.

Assume that it takes exactly **K** seconds for a packet to reach Bob's router from Alice's router (independent of the number of links) if it's successful. And when the data is successfully received in Bob's router, it immediately sends an acknowledgement to Alice's router and the acknowledgement always reaches her router exactly in **K** seconds (it never disappears).

Alice's router used the following algorithm for the data communication:

1. At time 0, the first KB of data is chosen to be sent.
2. It establishes a path (it takes no time) to the destination router and sends the data in this route.
3. It waits for exactly **2K** seconds.
    i. If it gets the acknowledgement of the current data in this interval:
        a. If **S** KB of data are sent, then step 4 is followed.
        b. Otherwise, it takes 1 KB of the next data, and then step 2 is followed.
    ii. Otherwise it resends the current 1 KB of data and then step 2 is followed.
4. All the data are sent, so it reports Alice.

Assume that the probabilities of the links are static and independent. That means it doesn't depend on the result of the previously sent data. Now your task is to choose some routes through the routers such that data can be sent in these routes and the expected time to send all the data to the destination routes is minimized. You only have to report the minimum expected time.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a line containing four integers **N (2 ≤ N ≤ 100)**, **M (1 ≤ M)**, **S (1 ≤ S ≤ 10^9)** and **K (1 ≤ K ≤ 20)**, where **M** denotes the number of bidirectional links. Each of the next **M** lines contains three integers u_i v_i p_i, meaning that there is a link between router u_i and v_i the the probability of a successful message transfer in this link is p_i% **($0 \le u_i, v_i < N, u_i \ne v_i, 0 < p_i \le 100$)**. There will be at most one link between any two routers.

Output

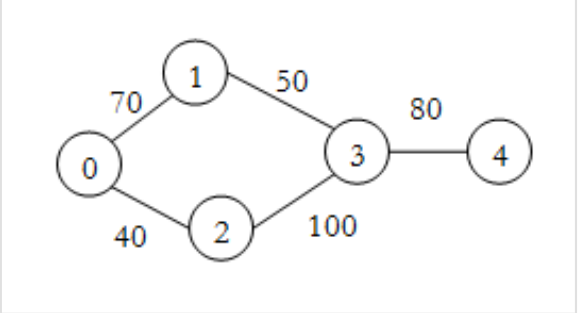
For each case, print the case number and the minimum possible expected time to send all the data. Errors less than 10^{-3} will be ignored. You can assume that at least one valid route between them always exists. And the result will be less than 10^{13}.

Sample

Input	Output
2 5 5 1 10 0 1 70 0 2 40 2 3 100 1 3 50 4 3 80 2 1 30 2 0 1 80	Case 1: 62.5000000000 Case 2: 150.0000000000

Notes

For sample 1, we get the following picture. We send the data through 0 - 2 - 3 - 4.



LightOJ

Ride Sharing Dilemma

2 seconds 64 MB Hard LOJ-1446 Debug

Statement Submissions Statistics Tutorial

English

There was a strange city with **n** junctions connected by **m** bidirectional roads. Even though the rapid growth of ride-sharing companies became a blessing for the general public, it also became a nightmare for the traditional taxi companies. So, they decided to come up with a scheme to stop this immense growth.

For simplicity, assume that a driver earns a static w_i amount passsing a road. Roads form loops, loop is a set of distinct junctions $j_1, j_2, ..., j_k$ except $j_1 = j_k$, $k > 3$ and (j_i, j_{i+1}) $(1 \le i < k)$ are connected by a road.

For each road, they want to set a minimum amount of toll (integer) such that there exists a loop containing the road and if a driver visits the roads in that loop in the same order, he/she needs to pay more tolls than the total earned money in the loop.

Input

Input starts with a positive integer **T (≤ 50)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers **n m (1 ≤ n ≤ 200, 0 ≤ m)**. Assume that the junctions are identified as integers. Each of the next **m** lines will contain three integers **u v w (0 ≤ u, v < n, u ≠ v, 0 < w ≤ 200)**, meaning that a driver earns **w** if he/she visits road connecting junction **u** and **v**. There is at most one road between any two junctions.

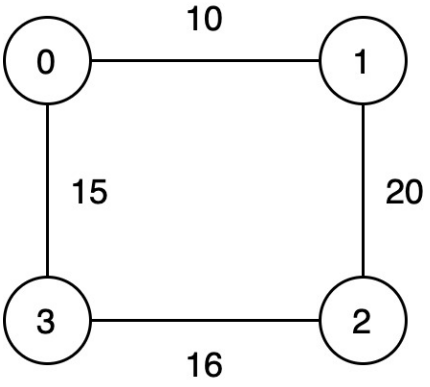
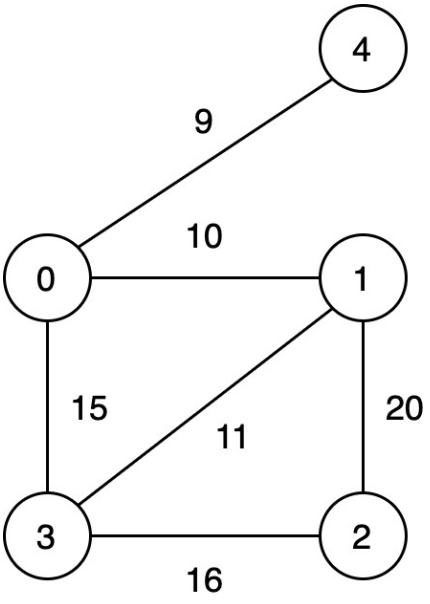
Output

For each case, print the case number in a single line. Then for each road in the city (in the order of input), print the minimum possible toll for that road. If it's not possible, print `impossible`.

Input	Output
2 4 4 0 1 10 1 2 20 0 3 15 3 2 16 5 6 0 1 10 1 2 20 0 3 15 3 2 16 0 4 9 1 3 11	Case 1: 52 42 47 46 Case 2: 27 28 22 32 impossible 26

Notes

- Dataset is huge, use faster I/O methods.

For case 1, for the first road **0 - 1**, if a toll of 52 is set to the road, then if a driver passes `0 - 1 - 2 - 3 - 0` (earning: -52 + 20 + 16 + 15 = -1); he/she is paying more toll than earning.	
For case 2, for the second road **1 - 2**, if a toll of 28 is added to the road, then if a driver passes `3 - 1 - 2 - 3` (earning: 11 - 28 + 16 = -1); he/she is paying more toll than earning.	

LightOJ

A Wedding Party

3 seconds 64 MB Medium Hard LOJ-1316 Debug

Statement Submissions Statistics Tutorial

English

We all know that we have a big and exciting wedding party ahead. We made a plan to buy a gift for the wedding. But just when we were about to busy the gift , we found out that we have a 'Team Practice Contest' ahead. Before going to the contest, we want the gift to be ready.

As time is too short, we will try to buy the gift on the way to the contest and will try to visit as many shops as possible. The city map is represented by a graph with **N** nodes and **M** edges. **N** nodes represent the **N** junctions and **M** edges represent the **M** unidirectional roads connecting the cities. Every road has a cost which represents the required time to use the road. The contest is running at junction **N-1** and we will start our journey at junction **0**. And there are exactly **S** shops located at different junctions.

Given the location of the shops you have to find the route from junction **0** to junction **N-1** which will visit maximum number of shops with minimum time (first maximize the number of shops then minimize the time to visit them). We can visit a junction more than once.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case begins with three non negative integers **N (2 ≤ N ≤ 500)**, **M (1 ≤ M ≤ 10000)** **S (0 ≤ S ≤ 15)**. Next line contains **S** integers denoting the shop locations. Each of the next **M** lines contains three integers **u**, **v, w (0 < u, v < N, u ≠ v, 1 ≤ w ≤ 100)** denoting a road from **u** to **v** with cost **w**.

Output

For each case of input you have to print the case number and two integers representing maximum number of shops we can visit in the way and the minimum time required to reach junction *_N-1_* *after visiting maximum number of shops. If we cannot attend the contest, print `Impossible` . See samples for more details.

Input	Output
2 4 4 4 0 1 2 3 0 1 10 1 3 30 0 2 30 2 3 5 4 4 4 0 1 2 3 0 1 10 3 1 30 0 2 30 3 2 5	Case 1: 3 35 Case 2: Impossible

LightOJ

Brush (V)

1 seconds 64 MB Medium LOJ-1019 Debug

Statement Submissions Statistics Tutorial

English

Tanvir returned home from the contest and got angry after seeing his room dusty. Who likes to see a dusty room after a mind boggling programming contest? After checking a bit he found that there is no brush in him room. So, he called Atiq to get a brush. But as usual Atiq refused to come. So, Tanvir decided to go to Atiq's house.

The city they live in is divided by some junctions. The junctions are connected by two way roads. They live in different junctions. And they can go to one junction to using the given roads only.

Now you are given the map of the city and the distances of the roads. You have to find the minimum distance Tanvir has to travel to reach Atiq's house.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains two integers **N (2 ≤ N ≤ 100)** and **M (0 ≤ M ≤ 1000)**, means that there are **N** junctions and **M** two way roads. Each of the next **M** lines will contain three integers **u v w (1 ≤ u, v ≤ N, w ≤ 1000)**, it means that there is a road between junction **u** and **v** and the distance is **w**.

You can assume that Tanvir lives in the **1st** junction and Atiq lives in the **Nth** junction. There can be multiple roads between same pair of junctions.

Output

For each case print the case number and the minimum distance Tanvir has to travel to reach Atiq's house. If it's impossible, then print `Impossible`.

Input	Output
2 3 2 1 2 50 2 3 10 3 1 1 2 40	Case 1: 60 Case 2: Impossible

Commandos

1 seconds 64 MB Medium LOJ-1174 Debug

Statement Submissions Statistics Tutorial

English

A group of commandos were assigned a critical task. They are to destroy an enemy headquarter. The enemy head quarter consists of several buildings and the buildings are connected by roads. The commandos must visit each building and place a bomb at the base of each building. They start their mission at the base of a particular building and from there they disseminate to reach each building. The commandos must use the available roads to travel between buildings. Any of them can visit one building after another, but they must all gather at a common place when their task in done. In this problem, you will be given the description of different enemy headquarters. Your job is to determine the minimum time needed to complete the mission. Each commando takes exactly one unit of time to move between buildings. You may assume that the time required to place a bomb is negligible. Each commando can carry unlimited number of bombs and there is an unlimited supply of commando troops for the mission.

Input

Input starts with an integer **T (≤50)**, denoting the number of test cases.

The first line of each case starts with a positive integer **N (1 ≤ N ≤ 100)**, where **N** denotes the number of buildings in the headquarter. The next line contains a positive integer **R**, where **R** is the number of roads connecting two buildings. Each of the next **R** lines contain two distinct numbers **u v (0 ≤ u, v < N)**, this means there is a road connecting building **u** to building **v**. The buildings are numbered from **0** to **N-1**. The last line of each case contains two integers **s d (0 ≤ s, d < N)**. Where **s** denotes the building from where the mission starts and **d** denotes the building where they must meet. You may assume that two buildings will be directly connected by at most one road. The input will be given such that, it will be possible to go from any building to another by using one or more roads.

Output

For each case, print the case number and the minimum time required to complete the mission.

Input	Output
2 4 3 0 1 2 1 1 3 0 3 2 1 0 1 1 0	Case 1: 4 Case 2: 1

LightOJ

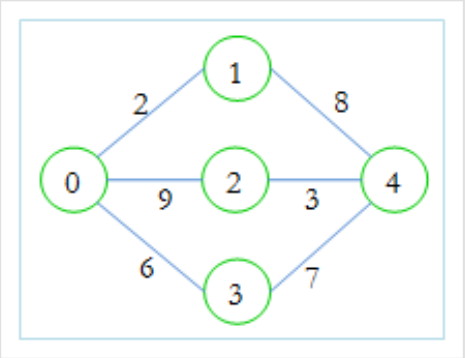
Country Roads

1 seconds 64 MB Easy LOJ-1002 Debug

Statement Submissions Statistics Tutorial

English

I am going to my home. There are many cities and many bi-directional roads between them. The cities are numbered from **0** to **n-1** and each road has a cost. There are **m** roads. You are given the number of my city **t** where I belong. Now from each city you have to find the minimum cost to go to my city. The cost is defined by the cost of the maximum road you have used to go to my city.



For example, in the above picture, if we want to go from 0 to 4, then we can choose

1. 0 - 1 - 4 which costs 8, as 8 (1 - 4) is the maximum road we used
2. 0 - 2 - 4 which costs 9, as 9 (0 - 2) is the maximum road we used
3. 0 - 3 - 4 which costs 7, as 7 (3 - 4) is the maximum road we used

So, our result is 7, as we can use 0 - 3 - 4.

Input

Input starts with an integer **T (≤ 20)**, denoting the number of test cases.

Each case starts with a blank line and two integers **n (1 ≤ n ≤ 500)** and **m (0 ≤ m ≤ 16000)**. The next m lines, each will contain three integers **u, v, w (0 ≤ u, v < n, u ≠ v, 1 ≤ w ≤ 20000)** indicating that there is a road between **u** and **v** with cost **w**. Then there will be a single integer **t (0 ≤ t < n)**. There can be multiple roads between two cities.

Output

For each case, print the case number first. Then for all the cities (from **0** to **n-1**) you have to print the cost. If there is no such path, print `Impossible`.

Input	Output
2 5 6 0 1 5 0 1 4 2 1 3 3 0 7 3 4 6 3 1 8 1 5 4 0 1 5 0 1 4 2 1 3 3 4 7 1	Case 1: 4 0 3 7 7 Case 2: 4 0 3 Impossible Impossible

Notes

Dataset is huge, use faster I/O methods.

LightOJ

Dangerous Bull! Who Wants to Pull?

1 seconds 64 MB Hard LOJ-1208 Debug

Statement Submissions Statistics Tutorial

English

A mad dangerous bull has freed himself from its chain. Now it's attacking all people that come by in its path. So, the people of the village 'Goru Mari' are frightened and at a loss. One of the villagers reported that the bull is sitting in a place (actually after all days of hard work, the bull was resting). So, the villagers made a plan to imprison the cow with a fence.

Assume the village as a 2D grid, where the bull is sitting on coordinate **(x, y)**. And some pairs of coordinates will be given **(x_1, y_1), (x_2, y_2)**, that means the villagers can put bamboos between this two points. Actually there is a tree in **(x_1, y_1)** and also one in **(x_2, y_2)**, and there are branches in the trees such that the villagers can tie **two** bamboos so that the bull won't be able to cross. The villagers are not able to place bamboos between trees which are not listed (may be, there are no suitable branches in those trees such that a bamboo can be tied).

Now the villagers want to imprison the bull, which means they want to use some bamboos to cover an area such that the bull will be inside that area and not able to cross that area. The villagers want the area to be convex. That means each angle is less than or equal to 180 degree. If there is a bamboo position that crosses any other bamboo position, then only one of them can be used.

Since bamboos are not so cheap, so the villagers want to imprison the bull such that the total length of the bamboo is as small as possible. And villagers are not good at math, so they asked your help.

Input

Input starts with an integer **T ($\leq$ 125)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains three integers **n x y** where **n ($1 \leq n \leq 100$)** denotes the number of tree pairs, **(x, y)** means the position of the bull. Each of the next **n** lines contains four integers **x_1 y_1 x_2 y_2**, meaning that you can place bamboos between **(x_1, y_1)** and **(x_2, y_2)**. You can assume that all the co-ordinates given for this problem satisfy **$(-10^4 \leq x_i, y_i \leq 10^4)$** and no two trees and the position of the bull will be collinear.

Output

For each case, print the case number and the minimum length of the bamboos needed. If no such solution is found, print **-1**. Errors less than **10^{-6}** will be ignored.

Input	Output
2 3 2 1 0 0 10 0 10 0 0 10 0 10 0 0 1 5 5 0 0 10 0	Case 1: 68.2842712475 Case 2: -1.000

New Traffic System

2 seconds 64 MB Medium Hard LOJ-1281 Debug

English

The country - Ajobdesh has a lot of problems in traffic system. As the Govt. is very clever (!), they made a plan to use only one way roads. Two cities **s** and **t** are the two most important cities in the country and mostly people travel from **s** to **t**. That's why the Govt. made a new plan to introduce some new one way roads in the traffic system such that the time to travel from **s** to **t** is reduced.

But since their budget is short, they can't construct more than **d** roads. So, they want to construct at most **d** new roads such that it becomes possible to reach **t** from **s** in shorter time. Unluckily you are one living in the country and you are assigned this task. That means you will be given the existing roads and the proposed new roads, you have to find the best path from **s** to **t**, which may allow at most **d** newly proposed roads.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing four integers **n (2 ≤ n ≤ 10000)**, **m (0 ≤ m ≤ 20000)**, **k (0 ≤ k ≤ 10000)**, **d (0 ≤ d ≤ 10)** where **n** denotes the number of cities, **m** denotes the number of existing roads and **k** denotes the number of proposed new roads. The cities are numbered from **0** to **n-1** and city **0** is denoted as **s** and city **(n-1)** is denoted as **t**.

Each of the next **m** lines contains a description of a road, which contains three integers u_i v_i w_i **($0 \le u_i, v_i < n$, $u_i \ne v_i$, $1 \le w_i \le 1000$)** meaning that there is a road from u_i to v_i and it takes w_i minutes to travel in the road. There is at most one road from one city to another city.

Each of the next **k** lines contains a proposed new road with three integers u_i v_i w_i **($0 \le u_i, v_i < n$, $u_i \ne v_i$ $1 \le w_i \le 1000$)** meaning that the road will be from u_i to v_i and it will take w_i minutes to travel in the road. There can be at most one proposed road from one city to another city.

Output

For each case, print the case number and the shortest path cost from **s** to **t** or `Impossible` if there is no path from **s** to **t**.

Input	Output
2 4 2 2 2 0 1 10 1 3 20 0 2 5 2 3 14 2 0 1 0 0 1 100	Case 1: 19 Case 2: Impossible

Notes

Dataset is huge, use faster I/O methods.

LightOJ

Not the Best

1 seconds 64 MB Medium LOJ-1099 Debug

Statement Submissions Statistics Tutorial

English

Robin has moved to a small village and likes visiting one of his best friends. He usually takes a longer route, because he likes the scenery along the way. He has decided to take the second-best rather than the shortest path. He knows that there must be a second-best path.

The countryside consists of **R** bidirectional roads, each linking two of the **N** intersections, conveniently numbered from **1**to **N**. Robin starts at intersection **1**, and his friend is at intersection **N**.

The second-best path may share roads with any of the shortest paths, and it may backtrack i.e. use the same road or intersection more than once. The second-best path is the shortest path whose length is longer than the shortest path(s) (i.e. if two or more shortest paths exist, the second-shortest path is the one whose length is longer than those but no longer than any other path).

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case contains two integers **N (1 ≤ N ≤ 5000)** and **R (1 ≤ R ≤ 10^5)**. Each of the next **R** lines contains three space-separated integers: **u, v** and **w** that describe a road that connects intersections **u** and **v** and has length **w (1 ≤ w ≤ 5000)**.

Output

For each case, print the case number and the second best shortest path as described above.

Input	Output
2 3 3 1 2 100 2 3 200 1 3 50 4 4 1 2 100 2 4 200 2 3 250 3 4 100	Case 1: 150 Case 2: 450

Prison Break

2 seconds 64 MB Medium Hard LOJ-1254 Debug

English

Michael Scofield has just broken out of the prison. He now wants to go to a certain city for his next unfinished job. As you are the only programmer in his gang, he asked for your help.

As you know that the fuel prices vary in the cities, you have to write a code to help Scofield that instructs him where to take the fuel and which path to choose. Assume that his car uses one unit of fuel in one unit of distance. Now he gives you the starting city **s** where he starts his journey with his car, the destination city **t** and the capacity of the fuel tank of his car **c**, the code should find the route that uses the cheapest fuel cost. You can assume that Scofield's car starts with an empty fuel tank.

Input

Input starts with an integer **T (≤ 5)**, denoting the number of test cases.

Each case starts with a line containing two integers **n (2 ≤ n ≤ 100)** and **m (0 ≤ m ≤ 1000)** where **n** denotes the number of cities and **m** denotes the number of roads. The next line contains **n** space separated integers, each lies between **1** and **100**. The i^{th} integer in this line denotes the fuel price (per unit) in the i^{th} city. Each of the next **m** lines contains three integers **u v w (0 ≤ u, v < n, 1 ≤ w ≤ 100, u ≠ v)** denoting that there is a road between city **u** and **v** whose length is **w**.

The next line contains an integer **q (1 ≤ q ≤ 100)** denoting the number of queries by Scofield. Each of the next **q** lines contains the request. Each request contains three integers: **c s t (1 ≤ c ≤ 100, 0 ≤ s, t < n)** where **c** denotes the capacity of the tank, **s** denotes the starting city and **t** denotes the destination city.

Output

For each case, print the case number first. Then for each query print the cheapest trip from **s** to **t** using the car with the given capacity **c** or `impossible` if there is no way of getting from **s** to **t** with the given car.

Input	Output
1 5 5 10 10 20 12 13 0 1 9 0 2 8 1 2 1 1 3 11 2 3 7 2 10 0 3 20 1 4	Case 1: 170 impossible

Toll Management

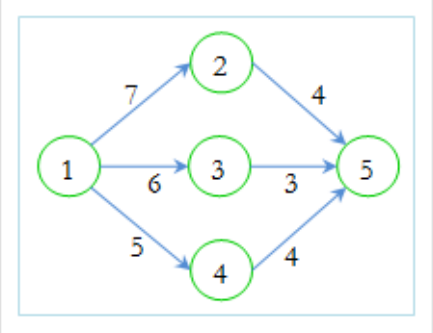
1 seconds 64 MB Medium Hard LOJ-1379 Debug

Statement Submissions Statistics Tutorial

English

In Dhaka, there are too many vehicles. So, the result is well known, yes, traffic jam. So, mostly people have to spend quite a time in the roads to go from one place to another.

Now, the students have finally found a solution to this problem. The idea is to make all the roads one way. That means a vehicle can go through the roads in one way only. And to make the number of vehicles low, each vehicle has to pay a toll to use a road. Now you want to go from a place **s** to another place **t**. And you have a total of **p** taka in your pocket. Now you want to find the path which contains the highest toll road, to go from **s** to **t**. Remember that you can't use more than **p** taka.



For the given picture, **s = 1, t = 5** and **p = 10**. There are three paths from 1 to 5.

1. Path 1: **1 - 2 - 5**, total toll = **11 (> p)**.
2. Path 2: **1 - 3 - 5**, total toll = **9 (≤ p)**, 6 is the maximum toll.
3. Path 3: **1 - 4 - 5**, total toll = **9 (≤ p)**, 5 is the maximum toll.

The maximum toll for a road of all of the paths having total toll not greater than **p** is 6.

Input

Input starts with an integer **T (≤ 10)**, denoting the number of test cases.

Each case starts with five integers **N (2 ≤ N ≤ 10000)**, **M (1 ≤ M ≤ 50000)**, **s (1 ≤ s ≤ N)**, **t (1 ≤ t ≤ N)** and **p (1 ≤ p ≤ 10^6)** where **N** means the number of junctions and **M** means the number of roads connecting the junctions. Then there will be **M** lines each containing three integers **u v c**. **u** and **v** are junctions and there is a road from **u** to **v (1 ≤ u, v ≤ N, u ≠ v)** and **c (0 ≤ c ≤ 10^5)** is the toll needed for that road. There can be multiple roads between two junctions.

Output

For each case, print the case number and the desired result. If no such result is found, print `-1`.

Input	Output
2 5 6 1 5 10 1 2 7 2 5 4 1 3 6 3 5 3 1 4 5 4 5 4 2 1 1 2 10 1 2 20	Case 1: 6 Case 2: -1

Notes

Dataset is huge, use faster I/O methods.

LightOJ

Extended Traffic

1 seconds 64 MB Medium LOJ-1074 Debug

Statement Submissions Statistics Tutorial

English

Dhaka city is getting crowded and noisy everyday. Certain roads always remain blocked for congestion. In order to convince people avoid shortest routes as it's the number one reason for roads crowded; the city authority has come up with a new plan.

Each junction of the city is marked with a positive integer **(≤ 20)** denoting the busy-ness of the junction. Whenever someone goes from one junction (the source junction) to another (the destination junction), the city authority gets an amount of money **(busy-ness of destination - busy-ness of source)**^{3} (that means the cube of the difference) from the traveler.

Now, the authority has appointed you to find the minimum total amount that can be earned when someone goes from a certain junction (the zero point) to several others.

Input

Input starts with an integer **T (≤ 50)**, denoting the number of test cases.

Each case contains a blank line and an integer **n (1 < n ≤ 200)** denoting the number of junctions. The next line contains **n** integers denoting the busyness of the junctions from **1** to **n** respectively.

The next line contains an integer **m**, the number of roads in the city. Each of the next **m** lines (one for each road) contains two junction-numbers (source, destination) that the corresponding road connects (all roads are unidirectional). The next line contains the integer **q**, the number of queries. The next **q** lines each contain a destination junction-number. There can be at most one direct road from a junction to another junction.

Output

For each case, print the case number in a single line. Then print **q** lines, one for each query, each containing the minimum total earning when one travels from junction **1** (the zero point) to the given junction. However, for the queries that gives total earning less than **3**, or if the destination is not reachable from the zero point, then print a `?`.

Input	Output
2 5 6 7 8 9 10 6 1 2 2 3 3 4 1 5 5 4 4 5 2 4 5 2 10 10 1 1 2 1 2	Case 1: 3 4 Case 2: ?

Instant View of Big Bang

1 seconds 64 MB Medium Hard LOJ-1108 Debug

English

Have you forgotten about wormholes? Oh my god! Ok, let me explain again.

A wormhole is a subspace tunnel through space and time connecting two star systems. Wormholes have a few peculiar properties:

1. Wormholes are one-way only.
2. The time it takes to travel through a wormhole is negligible.
3. A wormhole has two end points, each situated in a star system.
4. A star system may have more than one wormhole end point within its boundaries.
5. Between any pair of star systems, there is at most one wormhole in each direction.
6. There are no wormholes with both end points in the same star system.

All wormholes have a constant time difference between their end points. For example, a specific wormhole may cause the person traveling through it to end up 15 years in the future. Another wormhole may cause the person to end up 42 years in the past.

A brilliant physicist wants to use wormholes to study the Big Bang. Since warp drive has not been invented yet, it is not possible for her to travel from a star system to another one directly. This can be done using wormholes, of course.

The scientist can start her journey from any star system. Then she wants to reach a cycle of wormholes somewhere in the universe that causes her to end up in the past. By traveling along this cycle a lot of times, the scientist is able to go back as far in time as necessary to reach the beginning of the universe and see the Big Bang with her own eyes. Write a program to help her to find such star systems where she can start her journey.

Input

Input starts with an integer **T (≤ 125)**, denoting the number of test cases.

Each case starts with a blank line. The next line contains two numbers **n** and **m** . These indicate the number of star systems **(1 ≤ n ≤ 1000)** and the number of wormholes **(0 ≤ m ≤ 2000)**. The star systems are numbered from **0** to **n-1**. For each wormhole a line containing three integer numbers **x**, **y** and **t** is given. These numbers indicate that this wormhole allows someone to travel from the star system numbered **x** to the star system numbered **y**, thereby ending up **t (-1000 ≤ t ≤ 1000)** years in the future or past, a negative integer denotes past, positive integer denotes future.

Output

For each case, print the case number first. Then print the star systems (in ascending order) where she can start her journey. If no such star system is found, print `impossible` .

Input	Output
2 3 3 0 1 1000 1 2 15 2 1 -42 4 4 0 1 10 1 2 20 2 3 30 3 0 -60	Case 1: 0 1 2 Case 2: impossible

Travel Company

2 seconds 64 MB Medium Hard LOJ-1221 Debug

English

A travel company is planning to launch their bus service in a new route. So they conducted a survey and made a list of all possible roads connecting different cities. Each of the roads has a certain amount of income based on current fare. But at the same time, each road has some expenses too (this includes fuel and maintenance cost, staff payments, taxes and tribute to labor union which is recently approved by the Government). The travel company is looking for a cyclic route. That is, the bus will start from any city, then visit one or more cities each exactly once and return to the starting city. The company is also concerned with the profit on the route. In fact the directors of the company have a strict requirement of a profit ratio strictly greater than **P**. Otherwise they will not launch the service. A profit ratio for a route is the ratio between the total incomes to the total expenses for that route.

One of your friends works in that company and he asks for a little help from you. All you have to do is to determine if there exists such route, so that the company has a profit ratio of **P**.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a blank line and three integers **N, R, P (2 ≤ N ≤ 100, 0 ≤ R ≤ 9900, 1 ≤ P ≤ 100)**. **N**, **R** and **P** represents number of cities, number of road links and the expected profit ratio respectively.

Then **R** lines follow. Each line contains four integers A_i, B_i, I_i, E_i **(0 ≤ A_i, B_i < N, 0 ≤ I_i ≤ 5000, 1 ≤ E_i ≤ 5000)**. **(A_i, B_i)** represents directed road link from city A_i to B_i. I_i and E_i are the incomes and expenses of the road link respectively. You may assume that **(A_i, B_i) ≠ (A_j, B_j)**, if **i ≠ j** and $A_i \neq B_i$ for any **i**.

Output

For each case, print the case number and `YES` if there is a cyclic route for which the profit ratio is greater than **P** or `NO`, if there is no such route.

Input	Output
3 5 8 3 0 1 17 8 1 0 10 5 1 2 11 5 1 4 5 3 2 3 13 7 3 1 9 4 4 3 11 1 3 0 11 6 5 8 3 0 1 17 8 1 0 10 5 1 2 11 5 1 4 5 3 2 3 13 7 3 1 9 4 4 3 11 2 3 0 11 6 5 8 2 0 1 17 8 1 0 10 5 1 2 11 5 1 4 5 3 2 3 13 7 3 1 9 4 4 3 11 5 3 0 11 6	Case 1: YES Case 2: NO Case 3: YES

Notes

Dataset is huge. Use faster I/O methods.

LightOJ

Employment

1 seconds 64 MB Medium Hard LOJ-1400 Debug

Statement Submissions Statistics Tutorial

English

Employment is a contract between two parties, one being the employer and the other being the employee. Now assume that there are **n** companies having exactly **one** headcount each and there are **n** candidates applying for the positions. All the candidates interviewed in each of the companies and they have different preferences for the companies and the companies al have preferences for the candidates.

So, you are given the task to assign each candidate to each company such that the employment scheme is stable. A scheme is stable if there is no pair **(candidate$_i$, company$_j$)** and **(candidate$_x$, company$_y$)** where **Candidate$_i$** prefers **company$_y$** more than **company$_j$** and **Company$_y$** prefers **candidate$_i$** more than **candidate$_x$**.

As there can be many solutions, any valid one will do.

Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing an integer **n (1 ≤ n ≤ 100)**. The candidates are numbered from **1** to **n** and the companies are numbered from **n+1** to **2n**.

Each of the next **n** lines contains **n** distinct integers from **n+1** to **2n**, where the **ith** line contains the company preference for the **ith** candidate **(1 ≤ i ≤ n)**.

Each of the next **n** lines contains **n** distinct integers from **1** to **n**, where the **ith** line contains the candidate preference for the company which is denoted by **n+i (1 ≤ i ≤ n)**.

Output

For each case, print the case number and the (candidate, company) pairs. As there can be many solutions any valid one will do. And you can output the pairs in any order but print those as (candidate, company) pair.

Sample

Input	Output
1 3 4 5 6 6 5 4 5 4 6 2 1 3 1 2 3 3 2 1	Case 1: (2 6) (1 4) (3 5)