Problem A

Troublemakers

Time Limit: 1 second

[after seeing that the room is on fire; Ted has a needle
in his hand while holding the leg of a dead woman; Sara
has a bottle of champagne in her hand, and Juancho is smoking]
Man: *Did they misbehave?*

Robert Rodrigues, *"The Misbehavers."*

Every school class has its troublemakers - those kids who can make the teacher's life miserable. On his own, a troublemaker is manageable, but when you put certain pairs of troublemakers together in the same room, teaching a class becomes very hard. There are **n** kids in Mrs. Shaida's math class, and there are **m** pairs of troublemakers among them. The situation has gotten so bad that Mrs. Shaida has decided to split the class into two classes. Help her do it in such a way that the number of troublemaker pairs is reduced by at least a half.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing **n** (0<=**n**<=100) and **m** (0<**m**<5000). The next **m** lines will contain a pair of integers **u** and **v** meaning that when kids **u** and **v** are in the same room, they make a troublemaker pair. Kids are numbered from 1 to **n**.

Output

For each test case, output one line containing "Case **#x**:" followed by **L** - the number of kids who will be moved to a different class (in a different room). The next line should list those kids. The total number of troublemaker pairs in the two rooms must be at most **m**/2. If that is impossible, print "Impossible." instead of **L** and an empty line afterwards.

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

3 4 4 6 1 2 1 3 1 4 2 3 2 4 3 4	

Problemsetter: Igor Naverniouk

Problem B

Buy one, get the rest free.

Time Limit: 3 seconds

"Whoa! It feels like I'm flying!"

Lrrr

It's year 2258, and the age of airplanes is coming to an end. Everyone is using teleporters now. In an effort to stay competitive, the last remaining air travel company, GetsJo, is offering the following deal to its customers. Instead of buying one plane ticket, you can rent a whole flight from A to B. Each flight can carry a certain number of people and costs a certain amount of money. If you do that, then you can rent all of the other flights of equal or lesser cost for free!

For example, if there are 4 flights with costs $10000, $25000, $30000 and $40000, and you rent the $30000 flight, then you get the $10000 and $25000 flights for free. The total cost to rent these 3 flights is $30000.

You want to organize a large programming competition and would like to invite all of the participants to city **n**, where the competition will be held. Being a nice person, you decide to pay for everyone's airplane tickets. Given the locations of the participants and the list of available flights between now and the day of the competition, what is the cost of renting enough flights to get all of the participants to city **n** in the next **d** days?

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing the number of cities ($1 \le n \le 30$), the number of days ($1 \le d \le 10$) until the competition and the number of flights ($0 \le m \le 1000$). **m** lines follow, each one containing 5 integers: **u**, **v**, **c**, **p** and **e** ($1 \le u, v \le n$, $1 \le c \le 100$, $0 \le e < d$). This means that a flight that can carry **c** passengers and costs **p** dollars leaves city **u** on day **e** in the evening and arrives next day in the morning to city **v**. Day 0 is today, and all of the participants need to be in city **n** in the evening of day **e**. Finally, **n** integers (z_1, z_2, ..., z_n) follow, meaning that there are z_i participants in city **i** on day 0 ($0 \le z_i \le 100$). The maximum cost of a flight is 100000. There will never be two flights with the same **u**, **v** and **e** values.

Output

For each test case, output one line containing "Case #x:" followed by the minimum required cost of flying all of the participants to city n before the end of day d. If no amount of money is enough, print "Impossible" instead.

Sample Input	Sample Output
2 5 4 5 1 5 100 30000 0 2 4 10 10000 0 2 4 10 10000 1 4 5 25 25000 2 2 5 100 40000 3 1 20 0 5 100 2 1 1 1 2 99 10400 0 100 0	Case #1: 30000 Case #2: Impossible

Problemsetter: Igor Navernіouk
Alternate solution: Yury Kholondyrev

Problem C

Double NP-hard

Time Limit: 2 second

"Name and Class Year:
Course to be Covered: (Course Number and Title)
Reason for covering the course independently:"

Hamilton College Application for Independent
Coverage of Course Work

Definitions

In this problem, a *graph* is a set of **n** vertices together with a set of **m** edges, where an *edge* is an unordered pair of different vertices (edges are undirected). The two vertices that comprise an edge are said to be that edge's *endpoints*. A *vertex cover* of a given graph G is a subset C of its vertices, such that each edge of G has at least one of its endpoints in C. An *independent set* of a given graph G is a subset S of its vertices, such that no edge of G has both of its endpoints in S.

The problem of finding a *minimum vertex cover* (that is, a vertex cover of the smallest possible size) for any graph is NP-hard. The problem of finding a *maximum independent set* of any graph is also NP-hard. That is a formal way of saying that no one knows whether there exists an algorithm that runs in time polynomial in **n** and solves any one of the two problems.

We want to define a class of problems that are even harder than the NP-hard problems. We are going to call them "Double NP-hard"! Your job is to solve the first Double NP-hard problem.

Problem

Given a graph G, find a subset C of its vertices that is both a minimum vertex cover and a maximum independent set.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with two lines containing **n** (0<=**n**<=1000) and **m** (0<=**m**<=100000) as above. The next **m** lines will each describe an edge of G as a pair of different vertices, which are numbered from 1 to **n**.

Output

For each test case, output one line containing "Case #x:" followed by either "Impossible" if there is no answer or the size **k** of the set C. In the latter case, on the next line, print the **k** vertices of C in increasing order, separated by spaces. If there are multiple answers, print the lexicographically smallest one.

Sample Input	Sample Output
4 2 1 1 2 0 0 10 0 4 4 1 2 2 3 3 4 4 1	Case #1: 1 1 Case #2: 0 Case #3: Impossible Case #4: 2 1 3

Problemsetter: Igor Naverniouk

Problem D

Rings'n'Ropes

Time Limit: 3 seconds

"Well, that seems to be the situation. But,
I don't want that, and you don't want that,
and Ringo here definitely doesn't want that."

Jules Winnfield

I have **n** tiny rings made of steel. I also have **m** pieces of rope, all of exactly the same length. The two ends of each piece of rope are tied to two different rings.

I am going to take one of the rings, **L**, into my left hand, and another ring, **R** into my right hand. Then I will pull the whole structure apart as hard as I can. Some of the ropes will be streched horizontally because of this. Others will hang down or bend out of shape. If I want the number of horizontally stretched ropes to be as large as possible, which **L** and **R** should I pick?

Assume that the stretching of ropes in negligible, they all have negligible thickness and are free to slide around the rings that they are tied to. The thickness and radius of each ring is negligible, too.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with two lines containing **n** (2<=**n**<=120) and **m** (0<=**m**<=**n**(**n**-1)/2). The next **m** lines will each contain a pair of different rings (integers in the range [0, **n**-1]). Each pair of rings will be connected by at most one rope.

Output

For each test case, output the line containing "Case **#x**:", followed by the largest number of ropes that I can stretch horizontally by picking a pair of rings, **L** and **R**.

Sample Input	Sample Output
4	Case #1: 1

2 1 0 1 3 3 0 1 1 2 2 0 6 6 0 1 0 5 1 3 5 4 3 2 4 2 6 7 0 1 0 5 1 3 1 4 5 4 3 2 4 2	Case #2: 1 Case #3: 6 Case #4: 7

Problemsetter: Igor Naverniouk

Problem E

Sending email

Time Limit: 3 seconds

"A new internet watchdog is creating a stir in Springfield. Mr. X, if that is his real name, has come up with a sensational scoop."

Kent Brockman

There are **n** SMTP servers connected by network cables. Each of the **m** cables connects two computers and has a certain latency measured in milliseconds required to send an email message. What is the shortest time required to send a message from server **S** to server **T** along a sequence of cables? Assume that there is no delay incurred at any of the servers.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing **n** (2<=**n**<20000), **m** (0<=**m**<50000), **S** (0<=**S**<**n**) and **T** (0<=**T**<**n**). **S**!=**T**. The next **m** lines will each contain 3 integers: 2 different servers (in the range [0, **n**-1]) that are connected by a bidirectional cable and the latency, **w**, along this cable (0<=**w**<=10000).

Output

For each test case, output the line "Case **#x**:" followed by the number of milliseconds required to send a message from **S** to **T**. Print "unreachable" if there is no route from **S** to **T**.

Sample Input	Sample Output
3 2 1 0 1 0 1 100 3 3 2 0 0 1 100 0 2 200 1 2 50 2 0 0 1	Case #1: 100 Case #2: 150 Case #3: unreachable

Problemsetter: Igor Navernioukk

Problem F

AntiFloyd

Time Limit: 2 seconds

"There is nothing new under the sun but there are lots of old things we don't know."

Ambrose Bierce

You have been hired as a systems administrator for a large company. The company head office has **n** computers connected by a network of **m** cables. Each cable connects two different computers, and there is at most one cable connecting any given pair of computers. Each cable has a latency, measured in micro-seconds, that determines how long it takes for a message to travel along that cable. The network protocol is set up in a smart way, so that when sending a message from computer A to computer B, the message will travel along the path that has the smallest total latency, so that it arrives at B as soon as possible. The cables are bi-directional and have the same latency in both directions.

As your first order of business, you need to determine which computers are connected to each other, and what the latency is along each of the **m** cables. You soon discover that this is a difficult task because the building has many floors, and the cables are hidden inside walls. So here is what you decide to do. You will send a message from every computer A to every other computer B and measure the latency. This will give you **n**(**n**-1)/2 measurements. From this data, you will determine which computers are connected by cables, and what the latency along each cable is. You would like your model to be simple, so you want to use as few cables as possible.

Input

The first line of input gives the number of cases, **N** (at most 20). **N** test cases follow. Each one starts with a line containing **n** (0<**n**<100). The next **n**-1 lines will contain the message latency measurements. Line **i** will contain **i** integers in the range [1, 10000]. Integer **j** is the amount of time it takes to send a message from computer **i**+1 to computer **j** (or back).

Output

For each test case, output a line containing "Case #**x**:". The next line should contain **m** - the number of cables. The next **m** lines should contain

3 integers each: u, v and w, meaning that there is a cable between computers u and v, and it has latency w. Lines should be sorted first by u, then by v, with u<v. If there are multiple answers, any one will do. If the situation is impossible, print "Need better measurements." Print an empty line after each test case.

Sample Input	Sample Output
2 3 100 200 100 3 100 300 100	Case #1: 2 1 2 100 2 3 100 Case #2: Need better measurements.

Problemsetter: Igor Naverniouk
Alternate solution: Frank Chu

Problem G

From G to H and back

Time Limit: 3 seconds

"All right, I'm stumped... and I think I'm supposed to be."

Dana Scully

There is a funny transformation that you can do with a graph. We start with an undirected graph, G, and build a new graph, H. G has **n** vertices and **m** edges. For each edge in G, we create a vertex in H. Two vertices in H are connected by an edge if and only if their corresponding edges in G share a vertex. H will have **m** vertices and **p** edges.

That's easy. But what about reconstructing G, given H?

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with two lines containing **m** (at most 320) and **p**. **p** lines follow, each containing two different vertices (numbered from 1 to **m**) in H which are connected by an edge.

Output

For each test case, output one line containing "Case **#x**:" followed by either "yes" or "no", depending on whether there exists some graph G that produces the given graph H.

Sample Input	Sample Output
2 3 3 1 2 2 3 3 1 4 3 1 2 1 3 1 4	Case #1: yes Case #2: no

Problemsetter: Igor Naverniouk

Problem H

Bomb, Divide and Conquer

Time Limit: 3 seconds

"War is God's way of teaching Americans geography."

Ambrose Bierce (1842 - 1914)

The enemy has **n** cities connected by **m** roads. We have bombers that can destroy roads. The Bomb, Divide and Conquer strategy dictates that if we separate the enemy's cities from each other, then the two (or more) pieces will be easier to secure. Bombing a road has a cost (fuel, risk factor, etc.) What is the minimum total cost of bombing enough roads to ensure that there is some pair of cities that have no path between them? A path is a sequence of connected roads.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with two lines containing **n** (2<=**n**<=150) and **m** (0<=**m**<=**n**(**n**-1)/2). The next **m** lines contain **m** triples of integers denoting two different cities (between 1 and **n**) that are connected by a road and the cost of destroying that road (between 1 and 1000).

Output

For each test case, output one line containing "Case #**x**:" followed by the total cost of disconnecting a pair of cities.

Sample Input	Sample Output
4 5 4 1 2 100 2 3 299 3 5 400 5 4 99 3 3 1 2 10 2 3 20	Case #1: 99 Case #2: 30 Case #3: 30 Case #4: 0

1 3 40 4 5 1 2 10 2 3 100 3 4 10 4 1 100 1 3 10 3 1 1 2 1000	

Problemsetter: Igor Naverniouk