Algorithms for Programming Contests WS20/21 - Week 03

Chair for Foundations of Software Reliability and Theoretical Computer Science, TU München

Mikhail Raskin, Martin Helfrich, Maxi Weininger, Christoph Welzel

Welcome to our practical course! This problem set is due by

Wednesday, 25.11.2020, 6:00 a.m.

Try to solve all the problems and submit them at

`https://judge.in.tum.de/conpra/`

This week's problems are:

The following amount of points will be awarded for solving the problems.

Problem	A	B	C	D	E
Difficulty	easy	easy	medium	medium	hard
Points	4	4	6	6	8

If the judge does not accept your solution but you are sure you solved it correctly, use the "request clarification" option. In your request, include:

- the name of the problem (by selecting it in the subject field)
- a verbose description of your approach to solve the problem
- the time you submitted the solution we should judge

We will check your submission and award you half the points if there is only a minor flaw in your code.

If you have any questions please ask by using the judge's clarification form.

This page is intentionally left blank.

Problem A
Networking

Sometimes, Lea has to attend loud and boring parties. In the beginning of such events, everybody wants to say hello to all the other people attending. This usually creates a big mess, so this year a new system was introduced: The transitive symmetric stationary greeting system (TSSGS). The system works as follows: Greetings are now transitive, that is, if Lea greets Ralph and Ralph greets Tom, then Lea is considered to have greeted Tom (thus transitive). Greetings are also symmetric, thus if Lea greets Tom (directly or via transitivity), Tom is considered to have greeted Lea. As to further reduce work, people do not move through the room but simply shout out to persons they want to greet (thus stationary). Still everybody wants to greet everybody (possibly indirectly).

This system of course reduced the work, but now everybody was shouting through the room and it soon got very loud.

Lea wants to do one more optimization: The sound level should be as low as possible while satisfying all the constraints above. A greeting between two people is as loud as the distance between them, the sound level is the sum of all greetings that take place. Help Lea with that problem by providing the lowest sound level possible.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with an integer n, the number of people at the party, n lines follow. The i-th line consists of n integers $m_{i,j}$ where $m_{i,j}$ is the distance of person i to person j. It is always the case that $m_{i,i} = 0$ and $m_{i,j} = m_{j,i}$.

Output

For each test case, output one line containing "Case #i:" where i is its number, starting at 1.

Starting in the next line, output the greetings that take place, one per line. For each greeting, output its start person i and end person j, separated by one space, such that $i < j$. Furthermore, order your output lexicographically, that is, greeting ab should appear before greeting ij if $a < i$, or $a = i$ and $b < j$. Each line of the output should end with a line break.

If there are multiple ways the greetings can take place with minimal sound level, any of them will be accepted.

Constraints

- $1 \leq t \leq 20$.
- $1 \leq n \leq 150$.
- $0 \leq m_{i,j} \leq 10000$ for all $1 \leq i, j \leq n$
- $m_{i,i} = 0$ and $m_{i,j} = m_{j,i}$ for all $1 \leq i, j \leq n$

Sample Input 1

```
10
2
0 1
1 0

3
0 1 3
1 0 2
3 2 0

4
0 6 7 4
6 0 5 4
7 5 0 7
4 4 7 0

3
0 3 4
3 0 6
4 6 0

4
0 5 4 3
5 0 7 6
4 7 0 5
3 6 5 0

3
0 3 4
3 0 5
4 5 0

4
0 4 3 5
4 0 5 7
3 5 0 4
5 7 4 0

5
0 6 7 7 3
6 0 5 4 7
7 5 0 7 4
7 4 7 0 7
3 7 4 7 0

3
0 3 3
3 0 5
3 5 0

4
0 4 5 6
4 0 4 7
5 4 0 5
6 7 5 0
```

Sample Output 1

```
Case #1:
1 2
Case #2:
1 2
2 3
Case #3:
1 4
2 3
2 4
Case #4:
1 2
1 3
Case #5:
1 2
1 3
1 4
Case #6:
1 2
1 3
Case #7:
1 2
1 3
3 4
Case #8:
1 5
2 3
2 4
3 5
Case #9:
1 2
1 3
Case #10:
1 2
2 3
3 4
```

Problem B
Library Hell

Lea is running out of storage space on the system. She knows there are some large software packages installed that she never uses (like Qt), but she remembers that removing them can break some packages she needs for filling the order for more storage (like LibreOffice).

Lea has numbered all the packages on the system and exported the complete list of package dependencies. After preparing the list of packages to remove and the list of packages to keep, she needs to check that the plan is safe. Check that all packages to keep continue working, i.e., that they, their dependencies, dependencies of their dependencies etc. are not removed. While Lea herself can never make such a mistake, some of her friends might borrow the tool and specify the same package to keep and to remove, or no packages in one (or both) of the lists.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with a line containing four space-separated integers N, K, R, and D. Here N is the total number of packages, K is the number of packages to keep working, R is the number of packages to remove, and D is the number of package dependency records.

A line containing K space-separated integers $k_1, \ldots, k_K$ follows. The packages with numbers $k_1, \ldots, k_K$ and all their indirect (transitive) dependencies have to be preserved. The next lines contains R space-separated integers $r_1, \ldots, r_R$, denoting the list of packages planned for removal.

Afterwards D lines follow describing the dependencies. Each of these lines contains two space-separated integers u_i and d_i meaning that the package number u dependes on the package number d.

Output

For each test case, output one line containing "Case #i: ok" if the dependency requirements are satisfied, or "Case #i: conflict" otherwise.

Constraints

- $1 \leq t \leq 100$
- $1 \leq N \leq 10000$
- $0 \leq K \leq N$
- $0 \leq R \leq N$
- $0 \leq D \leq 200000$
- $1 \leq k_i \leq N$
- $1 \leq r_i \leq N$
- $1 \leq u_i \leq N$
- $1 \leq d_i \leq N$

Additionally, the total number of lines in a single test input shall not exceed 2 millions.

Sample Input 1	Sample Output 1
2 10 1 1 9 1 2 1 3 3 4 5 2 1 6 7 5 4 8 9 8 9 7 1 10 10 1 1 9 1 2 1 3 3 4 5 2 1 6 7 5 4 8 8 9 9 7 1 10	Case #1: ok Case #2: conflict

Problem C
City Roads

On her tour of Absurdistan, Lea visited Ancientia, an old city with many narrow roads. Some of its roads are one-way roads and can only be passed in one direction, while others are two-way roads and can be passed in two directions.

During the last "Tour d'Absurdistan", many racers went through this city and its roads. However, due to the narrow roads, many racers often blocked each other in the two-way roads, and due to the complex road network, they happened to cycle around needlessly, both of which which lead to traffic congestions and delays. To solve this problem, the city council of Ancientia wants to make the road network acyclic by turning all two-way roads into one-way roads.

Currently, the city council is still unsure if this is even possible to achieve. As Lea's problem solving skills are known throughout Absurdistan, they ask her to help. Given a road network with one-way (directed) and two-way (undirected) roads, they ask for an assignment of directions to the undirected roads such that the resulting road network (consisting only of directed roads) is acyclic. Can you tell Lea how to find such an assignment, if there exists one?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case consists of a line containing three integers $n\ m\ l$. n is the number of intersections (indexed from from 1 to n). m is the number of one-way (directed) roads. l is the number of two-way (undirected) roads. m lines follow. The j-th line contains two integers $a_j\ b_j$, meaning there is a one-way (directed) road from intersection a_j to intersection b_j. l lines follow. The j-th line contains two integers $a_j\ b_j$, meaning there is a two-way (undirected) road between intersections a_j and b_j. Two intersections are connected by at most one road.

Output

For each test case, if there is no way to turn the road network into an acyclic graph by assigning directions to the two-way roads, print a line "Case #i: no". Otherwise, print a line "Case #i: yes", followed by l more lines, describing an assignment. The j-th line should contain two integers $a_j\ b_j$ if the previous two-way road between a_j and b_j (or b_j and a_j) is now a one-way road from a_j to b_j. If there are multiple valid assignments, any one of them is accepted.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 1000$
- $0 \leq m + l \leq 10000$
- $1 \leq a_j, b_j \leq n$ for all $1 \leq j \leq m + l$
- $a_j \neq b_j$ for all $1 \leq j \leq m + l$
- There is at most one entry $a_j\ b_j$ or $b_j\ a_j$ for any $1 \leq a_j, b_j \leq n$.

Sample Input 1

```
3
4 2 3
1 2
4 3
3 1
2 4
2 3

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

8 5 6
2 1
3 2
2 6
4 5
5 8
1 4
2 5
3 7
4 8
6 7
6 8
```

Sample Output 1

```
Case #1: yes
1 3
4 2
2 3
Case #2: no
Case #3: yes
4 1
2 5
3 7
4 8
7 6
6 8
```

Sample Input 2

```
3
7 4 4
6 2
6 5
4 2
6 4
3 7
5 7
4 7
3 5

8 4 2
1 6
6 4
3 1
4 3
7 6
1 2

7 5 4
6 5
5 7
5 4
2 7
5 3
6 4
1 5
1 2
6 1
```

Sample Output 2

```
Case #1: yes
3 7
7 5
7 4
3 5
Case #2: no
Case #3: yes
6 4
1 5
1 2
1 6
```

Problem D
Candy Store

You might now know this about Lea, but besides being a brilliant mathematician she is also an exceptionally gifted confectioner. So far Lea has only been making pralines for her friends, but now she wants to take it a step further and turn her hobby into a career. Lea would like to open a candy store in Lindtown!

The first thing to do is to find a proper location for her store. There are n houses in Lindtown which are connected by $n-1$ bidirectional roads. All roads in Lindtown are of exact same length, and every house in the city can be reached from any other house by traveling along these roads.

Lea believes that she could maximize the turnover if her store was located in a way such that none of her potential customers would have to travel for too long. In other words, she would like to open the store at a location such that the maximal distance between her candy store and any other house in Lindtown is minimized. Lea has already drawn a map of Lindtown but she has trouble finding the perfect location. Maybe you can help her out!

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with one integer n, where n is the number of houses is Lindtown labeled from 0 to $n-1$. The following $n-1$ lines describe the roads in Lindtown. Each line contains two integers x_i and y_i denoting that the houses labeled with x_i and y_i are connected by a road.

Output

For each test case, output one line containing "Case #i: c" with i being the number of the test case starting at 1, and c being the number of the house at which Lea should open her candy store. If there are multiple optimal locations for Lea's candy store, any of them is acceptable for Lea.

Constraints

- $1 \leq t \leq 10$
- $2 \leq n \leq 10^5$
- $1 \leq x_i, y_i \leq n$ for all $1 \leq i \leq n$

Sample Input 1

```
3
4
1 2
2 3
3 4

5
1 2
1 3
1 4
1 5

8
1 2
2 3
3 4
2 5
3 6
5 7
5 8
```

Sample Output 1

```
Case #1: 2
Case #2: 1
Case #3: 2
```

Sample Input 2

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

3
2 3
3 1

6
3 6
3 2
2 1
6 4
6 5

8
3 2
3 8
2 5
8 6
2 7
8 4
6 1

8
3 7
3 4
4 8
8 2
4 5
5 1
7 6
```

Sample Output 2

```
Case #1: 5
Case #2: 3
Case #3: 3
Case #4: 8
Case #5: 4
```

Problem E
Water Temple

As you know by now, Lea has always been a great adventurer, always at the ready to explore the unknown. Again, she booked a trip to Templonia, to find another forgotten temple with relics of the past. After months of searching, she found the legendary "Water Temple" - a temple to honor the water spirits, built by a long forgotten people. As she explores the entrance levels, she notices that the temple continues for many levels below ground. Luckily, she even found a map that shows her how many rooms there are and how to get to each one.

There is just a slight problem - the lower levels are all filled with water. The ancient priests must have tried to preserve the water spirits holy sanctum. So to explore all of the rooms, Lea has to drain some of the water first. Otherwise, she would not get to see the marvelous artifacts hidden in the deeper layers of the temple.

Upon closer inspection of the map, she found that several of the rooms are marked as "control rooms" that can be used to drain some of the water out of the temple. They even carry small numbers that tell Lea just how much water can be drained at this specific room. She also found out the lowest point of every hallway, so she knows how much water to drain until she can go through the hallways and enter the room they lead to. Since the temple seems to be connected by some elaborate pipe system, the water inside the temple has the same level everywhere.

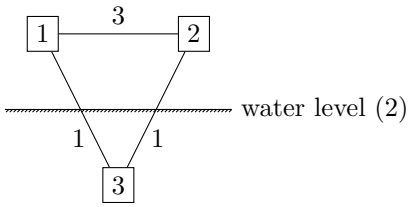

Figure E.1: Sample 1, case 2

Of course Lea wants to explore all the fascinating rooms in the temple, so can you tell her if she can lower the water level enough so that every room is reachable?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with a single line containing four space-separated integers n, m, k and l, where n is the amount of rooms in the water temple, numbered from 1 to n, and m is the amount of hallways connecting the rooms, k is the amount of rooms that are control rooms and l is the initial water level. m lines follow describing the hallways: the i-th line contains three space-separated integers a_i, b_i and l_i, describing a hallway between rooms a_i and b_i. To use this hallway, the water level must have been drained to at least water level l_i. The hallway can be used in either direction. You can assume that a room is drained of water as soon as there is at least one hallway to it that is usable. k lines follow describing the control rooms: the i-th line contains two space-separated integers a_i, d_i specifying that room a_i is a control room. If Lea reaches room a_i and operates the machinery there, she can drain the water throughout the whole temple to any level between the current water level and d_i (inclusively).

Output

For each test case, print a line containing "Case #i: r" where i is its number, starting at 1, and r is the maximum water level that can be left inside the temple so that every room is reachable from the entrance (room 1). Print "Case #i: `impossible`" if Lea cannot lower the water level enough so that every room is reachable. Each line of the output should end with a line break.

Sample explanation

In the first sample, in the first case, the initial water level is 2. This means Lea can use the hallways from room 1 to 2 and from 2 to 3, but not the hallway from 1 to 3 (the water level would have to be lowered to 1 to use it). There are no control rooms, but luckily, Lea can already reach all rooms, so the answer is the inital water level.

In the second case, Lea can only use the hallway from 1 to 2, but has no way to lower the water level any further. Thus she cannot explore the temple fully.

In the third case, Lea can use the hallway from 1 to 2. There, she can lower the water level to 0. However, she only lowers it to 1. Then she can go back to the entrance and use the hallway from 1 to 3 to reach the final room.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 10^4$
- $0 \leq m \leq 10^6$
- $0 \leq k \leq n$
- $0 \leq l \leq 10^4$
- $1 \leq a_i, b_i \leq n$ for all $1 \leq i \leq n$
- $0 \leq d_i \leq 10^4$
- The temple is connected, i.e. there is a water level so that every room is reachable from the entrance.
- The controls rooms given are unique, i.e. every room is mentioned at most once.

Sample Input 1

```
3
3 3 0 2
1 2 3
1 3 1
2 3 3

3 3 0 2
1 2 3
1 3 1
2 3 1

3 3 1 2
1 2 2
1 3 1
2 3 0
2 0
```

Sample Output 1

```
Case #1: 2
Case #2: impossible
Case #3: 1
```

Sample Input 2

```
5
4 5 1 1
1 2 3
3 4 8
1 4 10
4 3 6
2 4 4
2 5

2 3 0 3
2 1 3
2 2 9
1 2 9

4 5 1 8
3 4 8
1 2 9
2 3 0
1 4 10
1 4 6
1 1

5 6 3 10
5 4 6
1 1 3
1 3 6
3 5 8
3 2 7
1 3 10
3 6
1 4
5 0

9 10 0 7
9 8 7
4 3 4
1 3 0
1 4 5
2 9 7
1 7 6
3 2 9
3 2 3
2 5 6
4 6 6
```

Sample Output 2

```
Case #1: 1
Case #2: 3
Case #3: 8
Case #4: 6
Case #5: impossible
```