Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen (LEA)
Prof. Dr. Harald Räcke
Moritz Fuchs, Philipp Hoffmann, Christian Müller
Chris Pinkau, Stefan Toman

Winter Term 2015/16
Problem Set 4
04.11.2015

Algorithms for Programming Contests

This problem set is due by

Wednesday, 11.11.2015, 6:00 a.m.

Try to solve all the problems and submit them at

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

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.

A Hiking

Author: Philipp Hoffmann

Lea enjoys nature a lot, therefore she often goes hiking at the weekend. Last Sunday, she got up early, drove to the foot of a mountain and reached the top just at the right time for lunch. During her dessert, “Apfelstrudel”, she suddenly remembered: She had a very important appointment this afternoon which she was about to miss. In a big hurry, she looked at the map to figure out the fastest way to her car. To her amazement, there were hundreds of walking tracks which crossed multiple times forming thousands of possible routes down! Lea was helpless. Luckily she had her satellite phone with her and called... you! After giving you the list of all the tracks she wants to know how far away from her car she currently is. 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 two integers n and m where n is the number of intersections, those are numbered from 1 to n, and m is the number of walking trails, m lines follow. The i-th line contains three integers v_i, w_i and c_i. v_i and w_i each denote an intersection of walking trails, c_i is the length of a walking track connecting those intersections. Walking trails are undirected. Lea is currently at intersection 1, her car at intersection n.

Output

For each test case, output one line containing “Case #i: d” where i is its number, starting at 1, and d is the shortest distance of intersection 1 to intersection n. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 1000$
- $1 \leq m \leq 50000$
- $1 \leq c_i \leq 1000$ for all $1 \leq i \leq m$
- $1 \leq v_i, w_i \leq n$ for all $1 \leq i \leq m$

Sample Data

Input

```
7
3 2
1 2 1
2 3 2

3 3
1 2 1
1 3 3
2 3 1

3 2
1 2 1
2 3 2

3 2
1 3 5
2 3 4

6 11
1 2 2
1 5 7
1 3 3
3 4 6
2 3 6
2 4 3
2 6 3
3 6 1
3 5 1
4 6 2
5 6 2

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

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

Output

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

B Currency Exchange

Author: Chris Pinkau

Summer is coming soon and Lea wants to travel the world. Since she was a kid, she always dreamed of visiting the Great Temples of Templonia, and this summer her dream shall come true. As Templonians pay with their own currency, the Column, Lea has to go to a bank to exchange currencies. The Column is very rarely exchanged, so she decides to go to the National Bank, which has almost all the currencies of the world available. While looking at the current exchange rates, she may just have discovered a loophole in the system: Changing a currency via several exchanges may leave her with more money than just changing to the desired currency in one step.

Soon, Lea realized that the optimal sequence of exchanges is found by multiplying exchange ratios. Luckily, she remembered a grade school course she had taken on calculus, and knows that $\log(a \cdot b) = \log a + \log b$. Moreover, $\log(a \cdot b)$ is minimal if and only if $a \cdot b$ is minimal. This way, Lea can sum the logarithms of the exchange rates and find the optimal sequence. Can you help Lea write a program to find the best way to change her money into Columns?

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 two integers n and m, separated by a space. n denotes the number of currencies (w.l.o.g. the currencies are labelled 1 to n), m denotes the number of possible exchanges between currencies. m lines follow. The i-th line consists of two integers a_i, b_i, and a double c_i, separated by spaces, which means that the i-th exchange gives the rate c_i for changing the currency a_i into the currency b_i, i.e., one can change c_i units of currency a_i into one unit of currency b_i. Note that this does not imply that Lea can change money back from currency b_i to a_i.

The doubles c_i are given with a dot as the decimal symbol.

Lea's current money is given in currency 1, and the Column is represented by currency n.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is one of the three following answers: If Lea can make an infinite amount of money in any currency, then x is "Jackpot". Otherwise x is either the best (smallest) exchange rate achieved via a sequence of exchanges (precise up to 6 decimals); or "impossible" if there is neither a possible exchange between the two currencies nor a way to make infinite money.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 500$
- $0 \leq m \leq 5000$
- $1 \leq a_i, b_i \leq n$ for all $1 \leq i \leq m$
- $0 < c_i < 25$ for all $1 \leq i \leq m$

Sample Data

Input

```
9
4 5
1 2 0.6
1 4 2.0
2 3 0.4
3 1 4.5
4 3 0.4

4 5
1 2 0.6
1 4 2.0
2 3 0.4
3 1 3.0
4 3 0.4

3 4
3 1 18.064478131834562
1 2 2.5613012972458273
1 3 5.744360639529139
3 1 7.151240508976761

2 2
2 1 16.73195652637985
1 2 12.461936251916756

7 5
3 5 11.555567056802557
1 5 24.376215019378815
5 2 24.981765901216963
6 2 24.497367744083956
2 3 18.247482100234066

3 6
2 1 8.089884050564732
1 3 8.331091683144443
2 3 20.614121709340395
1 2 1.9207737413789867
3 1 0.6213988758282524
3 1 20.07488686398781

4 4
4 1 21.0952396315938
3 4 15.48741063715345
4 3 1.676384950685128
2 3 0.37772892327702645

5 8
2 3 13.650585200324855
1 2 24.12830264490565
3 4 18.140786261026435
3 5 20.907615501917792
1 3 0.6161124355635261
1 2 12.176086376957201
3 2 6.644022722972065
2 5 21.984698168506878

6 6
5 3 7.22318569705959
5 6 23.559017709074567
3 5 6.663404710366453
4 1 2.229260413216505
4 3 22.055905464703976
4 1 13.154033492200067
```

Output

```
Case #1: 2.000000
Case #2: Jackpot
Case #3: 5.744361
Case #4: 12.461936
Case #5: impossible
Case #6: 8.331092
Case #7: impossible
Case #8: 12.881442
Case #9: impossible
```

C Snakes and Ladders

Author: Stefan Toman

Lea's favourite board game is "Snakes and Ladders", a simple but funny game. She wins most games when she plays with her family, but when she plays it with her neighbours, Lea always loses. Lea is sure that they are cheating since they always use separate dice for each player. This time, she wants revenge: Lea plans to take a manipulated die with her and win for the first time against the neighbours. She is able to manipulate her die in a way that it always shows the same number. But which number should she choose?

To help her, you will need some more information about the game. "Snakes and Ladders" is played on a board with several fields forming a long queue. All of them are labelled with an integer from 1 to n in order. When the game begins, everybody puts his piece on field 1. In each round every player rolls a six-sided die and moves his piece the according number of fields. It is possible that several pieces are on the same field.

To make it more interesting, there are snakes (most times pointing downwards) and ladders (most times pointing upwards) drawn on the board. If some piece steps on a field with the head of a snake, it will move to the snake's tail. On the other hand, if it steps on a piece with the beginning of a ladder, it may go to the field where the ladder ends. All players are allowed to choose whether they want to use ladders, but everybody must use snakes. Note that snakes and ladders are directed and that a player may use several of them in one turn.

The first one to reach field n wins (even if the move might need to be continued due to snakes). If some player has a higher number than needed to reach field n he wins, too.

Tell Lea which number her die should always show to win the game as fast as possible. Lea will always choose whether to use ladders or not in an optimal way. If there are several numbers with the same speed print all of them.

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 three integers n, s and l. n is the number of fields on the board, s the number of snakes and l the number of ladders. s lines follow: the i-th line contains two integers a_i and b_i describing a snake having its head at field a_i and its tail at field b_i. Similarly, l lines follow them, line j containing two integers c_j and d_j describing a ladder starting at field c_j and ending at field d_j.

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1, and x is the number which lets Lea finish the game as fast as possible. If there are several fastest numbers, print each of them in increasing order separated by spaces. If Lea can't finish the game either way, print "impossible".

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 10000$
- $0 \leq s, l \leq n$
- $1 < a_i, b_i < n$ for all $1 \leq i \leq s$
- $1 < c_j, d_j < n$ for all $1 \leq j \leq l$
- On each field, at most one snake or one ladder starts, but not both.

Sample Data

Input

```
16
100 0 0

13 0 1
2 12

200 1 0
121 61

5 0 0

6 0 1
2 3

10 1 2
9 5
2 4
8 7

9 1 1
5 7
6 4

4 0 0

7 0 1
5 2

2 0 0

6 1 1
3 5
2 4

8 0 0

10 2 1
5 7
2 6
4 9

10 0 2
9 4
5 6

6 1 0
5 4

9 1 0
2 6
```

Output

```
Case #1: 6
Case #2: 1 6
Case #3: impossible
Case #4: 4 5 6
Case #5: 5 6
Case #6: 5 6
Case #7: 4 5 6
Case #8: 3 4 5 6
Case #9: 6
Case #10: 1 2 3 4 5 6
Case #11: 5 6
Case #12: 4 5 6
Case #13: 3 4 5 6
Case #14: 4 5 6
Case #15: 5 6
Case #16: 4 5 6
```

D Supermarkets

Author: Stefan Toman

It was a long time ago that Lea last saw Peter. She got to know him at school, but now she has not seen him for years. One day she met Peter by chance and he invited Lea to visit him at his new home.

A few days later when Lea wants to leave by car, she suddenly remembers that she forgot to buy a gift. Therefore, she decides to buy a bottle of wine at some supermarket on her way to Peter. She wants to be on time, so the extra way and time needed to buy the wine should be as short as possible. Some of the supermarkets are huge malls where she would need a lot of time to get her wine, some are known for long waiting times and others are very small and perfect for getting just one item. Lea knows the lengths of all roads, the locations of all supermarkets and the time she would need to buy the wine in each store. Where should she buy the wine to reach Peter as fast as possible?

Input

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

Each test case starts with a single line containing five integers n, m, s, a and b. n is the number of cities (labelled city 1 to city n), m is the number of roads and s is the number of supermarkets. Lea lives in city a whereas Peter lives in city b.

Next, there are m lines describing the roads. The i-th line contains three integers x_i, y_i and z_i and implies that there is a road between city x_i and city y_i (which may be used in both directions) for which Lea will need z_i minutes. s lines follow describing the supermarkets. The j-th line contains two integers c_j and w_j describing a supermarket in city c_j where Lea will need w_j minutes to buy the wine. Note that there may be multiple roads between cities as well as multiple supermarkets per city.

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1, and x is the time she needs to go to Peter formatted as "hours:minutes", for instance "5:23" (add leading zeros to the number of minutes if needed) or "impossible" if there is no way to Peter's house.

Constraints

- $1 \leq t \leq 20$
- $2 \leq n \leq 10000$
- $0 \leq m \leq n^2$

- $1 \leq a \leq n$
- $1 \leq b \leq n$
- $0 \leq s \leq n$
- $1 \leq x_i, y_i \leq n$ for all $1 \leq i \leq m$
- $1 \leq z_i \leq 100$ for all $1 \leq i \leq m$
- $1 \leq c_j \leq n$ for all $1 \leq j \leq s$
- $1 \leq w_i \leq 1000$ for all $1 \leq j \leq s$

Sample Data

Input

```
10
2 1 2 1 2
1 2 30
1 15
2 20

2 1 0 1 2
1 2 30

5 5 1 4 2
3 5 18
2 5 14
3 1 5
1 2 14
4 3 1
3 100

3 1 2 1 2
1 2 1
2 45
3 72

3 1 0 3 1
2 3 14

5 0 0 3 1

6 0 1 1 3
4 106

7 5 3 7 2
7 7 14
1 1 16
6 3 5
6 2 14
7 6 17
4 119
3 48
2 103

2 0 1 2 1
2 110

4 3 3 4 2
4 2 18
2 4 15
4 4 18
3 60
1 91
4 83
```

Output

```
Case #1: 0:45
Case #2: impossible
Case #3: 2:00
Case #4: 0:46
Case #5: impossible
Case #6: impossible
Case #7: impossible
Case #8: 1:29
Case #9: impossible
Case #10: 1:38
```

E Escher Stairs

Author: Christian Müller

Recently, Lea went to an art exhibition with many interesting pictures. She especially liked one part of the exhibition that dealt with non-euclidean geometry, for example buildings that can not be built in the real world. A famous example of this is “Relativity” by M. C. Escher.

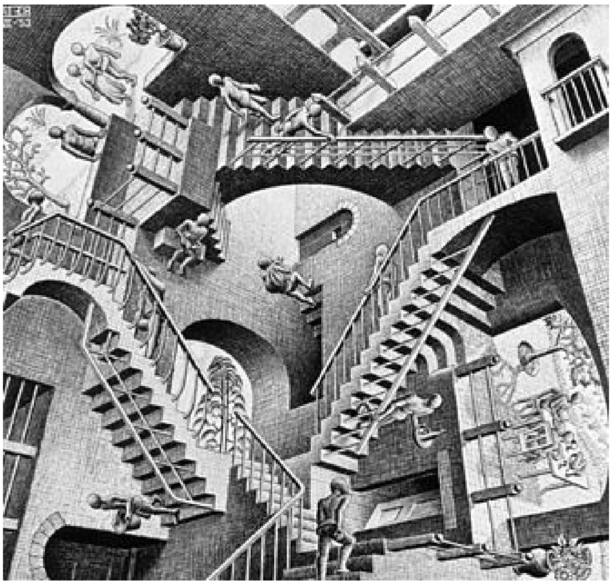

Figure 1: Relativity, by M. C. Escher. Lithograph, 1953. Source: Official M.C. Escher website (`http://www.mcescher.com/`).

There, Lea found many pictures of buildings with stairs. For fun, she tried to imagine if this particular building could be built in the real world. She checked this by counting the amount of steps of every flight of stairs, positive for going up, negative for going down. Then she chose a random starting point and tried to get back to the same point. If she could find a sequence of stairs that led her back where the amount of steps did not equal 0, she could be sure that the building could not be built (assuming all steps are of equal height). Otherwise, Lea is certain that some genius architect will find a way to construct such a building.

Lea easily got lost in the picture while counting. Can you tell her if the building could be real?

Input

The first line of the input contains an integer t. t test cases follow.

Each test case begins with a line containing three integers n m s, where n is the amount of places (indexed from 1 to n), m is the amount of connecting flights of stairs and s is the point Lea chose to start in. m lines follow. The i-th line consists of three integers a_i, b_i, c_i separated by spaces, meaning that there is a flight of stairs from place a_i to b_i

with c_i steps. All flights of stairs can be used in both directions, but are only given going upward, i.e. to go from a_i to b_i you would go c_i steps up, and to go from b_i to a_i you would go c_i steps down.

Output

For each test case, print a line containing "Case #i: possible" if there is no path from s to s such that the sum of steps is different from 0. Otherwise, print "Case #i: k", where k is a minimal number of flights of stairs Lea can take that lead her back to s with a step-sum different from 0. This number should correspond exactly to the path she took, so if she takes the same flight of stairs more than once, it is also counted again.

Constraints

- $1 \leq t \leq 20$
- $1 \leq s \leq n \leq 2000$
- $0 \leq m \leq 3 \cdot 10^5$
- $1 \leq a_i, b_i \leq n$ for all $1 \leq i \leq m$
- $0 \leq c_i \leq 5000$ for all $1 \leq i \leq m$
- The graph is connected.

Sample Data

Input

```
9
3 3 1
1 2 1
2 3 1
1 3 2

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

4 3 1
1 4 4
1 2 4
3 4 9

4 6 1
4 1 1
1 3 0
1 2 0
4 1 1
2 3 1
4 4 1

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

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

5 8 3
1 4 0
4 3 0
3 2 4
1 5 2
2 2 0
4 5 1
3 2 3
5 2 1

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

4 6 1
1 2 4
4 2 4
3 1 3
3 4 3
1 1 1
3 2 8
```

Output

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