Problem A

Not That Kind of Graph

Time Limit: 1 second

"You know, it's all very sweet, stealing from the rich, selling to the poor..."

Jose Molina, "Firefly."

Your task is to graph the price of a stock over time. In one unit of time, the stock can either **R**ise, **F**all or stay **C**onstant. The stock's price will be given to you as a string of R's, F's and C's. You need to graph it using the characters '/' (slash), '\' (backslash) and '_' (underscore).

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one contains a string of at least 1 and at most 50 upper case characters (R, F or C).

Output

For each test case, output the line "Case #**x**:", where **x** is the number of the test case. Then print the graph, as shown in the sample output, including the x- and y-axes. The x-axis should be one character longer than the graph, and there should be one space between the y-axis and the start of the graph. There should be no trailing spaces on any line. Do not print unnecessary lines. Finally, print an empty line after each test case.

Sample Input	**Sample Output**

<pre>1
RCRFCRFFCCRRC</pre>
	<pre>Case #1:
	| _
	| _/_/\ /
	| / __/
	+--------------</pre>

Problemsetter: Igor Naverniouk

Problem B

Lift Hopping

Time Limit: 1 second

Ted the bellhop: *"I'm coming up and if there isn't a dead body by the time I get there, I'll make one myself. You!"*

Robert Rodriguez, "Four Rooms."

A skyscraper has no more than 100 floors, numbered from 0 to 99. It has **n** (1<=**n**<=5) elevators which travel up and down at (possibly) different speeds. For each **i** in {1, 2,... n}, elevator number **i** takes $\mathbf{T_i}$ (1<=$\mathbf{T_i}$<=100) seconds to travel between any two adjacent floors (going up or down). Elevators do not necessarily stop at every floor. What's worse, not every floor is necessarily accessible by an elevator.

You are on floor 0 and would like to get to floor **k** as quickly as possible. Assume that you do not need to wait to board the first elevator you step into and (for simplicity) the operation of switching an elevator on some floor always takes exactly a minute. Of course, both elevators have to stop at that floor. You are forbiden from using the staircase. No one else is in the elevator with you, so you don't have to stop if you don't want to. Calculate the minimum number of seconds required to get from floor 0 to floor **k** (passing floor **k** while inside an elevator that does not stop there does not count as "getting to floor **k**").

Input

The input will consist of a number of test cases. Each test case will begin with two numbers, **n** and **k**, on a line. The next line will contain the numbers $\mathbf{T_1}$, $\mathbf{T_2}$,... $\mathbf{T_n}$. Finally, the next **n** lines will contain sorted lists of integers - the first line will list the floors visited by elevator number 1, the next one will list the floors visited by elevator number 2, etc.

Output

For each test case, output one number on a line by itself - the minimum number of seconds required to get to floor **k** from floor 0. If it is impossible to do, print "IMPOSSIBLE" instead.

Sample Input	Sample Output
2 30 10 5 0 1 3 5 7 9 11 13 15 20 99 4 13 15 19 20 25 30 2 30 10 1 0 5 10 12 14 20 25 30 2 4 6 8 10 12 14 22 25 28 29 3 50 10 50 100 0 10 30 40 0 20 30 0 20 50 1 1 2 0 2 4 6 8 10	275 285 3920 IMPOSSIBLE

Explanation of examples

In the first example, take elevator 1 to floor 13 (130 seconds), wait 60 seconds to switch to elevator 2 and ride it to floor 30 (85 seconds) for a total of 275 seconds.

In the second example, take elevator 1 to floor 10, switch to elevator 2 and ride it until floor 25. There, switch back to elevator 1 and get off at the 30'th floor. The total time is
10*10 + 60 + 15*1 + 60 + 5*10 = 285 seconds.

In example 3, take elevator 1 to floor 30, then elevator 2 to floor 20 and then elevator 3 to floor 50.

In the last example, the one elevator does not stop at floor 1.

Problemsetter: Igor Naverniouk
Alternate solutions: Stefan Pochmann, Frank Pok Man Chu

Problem C
Lex Smallest Drive
Time Limit: 2 seconds

"OK. Let's go right."

Bartholomew Furrow

A *graph*, G, consists of a finite set of *vertices*, V, and a set of *edges*, E, where each edge is a set of 2 vertices {u, v}. A *walk* in G is a finite sequence of vertices (v_1, v_2, ..., v_k), such that for each pair (v_{i-1}, v_i) for i in [2, k], {v_{i-1}, v_i} is in E. This is called a "walk from v_1 to v_k". If V is a set of integers, then any two walks in G can be compared lexicographically; for example, the walk (3, 5, 6, 2, 8) is smaller than the walk (3, 5, 6, 5, 7). A walk, W, from a to b is *lexicographically smallest* if there is no other walk from a to b in G that is smaller than W. A *drive* is a walk (v_1, v_2, ..., v_k), where no edge is used twice consecutively. That is, for all i from 2 up to k-1, v_{i-1} is not equal to v_{i+1}.

Given G and a start vertex, **s**, your task is to find the lexicographically smallest drives from **s** to each vertex in G.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing the integers **n**, **m** and **s**. (0 <= **n** <= 100, 0 <= **m** <= 4950). The next **m** lines will list the edges of G. V is the set {0, 1, ..., **n**-1}. **s** is in V.

Output

For each test case, output the line "Case #**x**:", where **x** is the number of the test case. Then print **n** lines, line i listing the lexicographically smallest drive from **s** to i using single spaces to separate consecutive vertices. If there is no such walk, print "No drive." Put an empty line after each test case.

Sample Input	Sample Output
1 6 4 5 5 0 2 5 4 0 3 1	Case #1: 5 0 No drive. 5 2 No drive. 5 0 4 5

Problemsetter: Igor Naverniouk
Alternate solution: Yury Kholondyrev

Problem D
Thunder Mountain
Time Limit: 3 seconds

"I mean, some people got guns, and some people got flashlights, and some people got batteries. These guys had all three."

J. Michael Straczynski, "Jeremiah."

Markus is building an army to fight the evil Valhalla Sector, so he needs to move some supplies between several of the nearby towns. The woods are full of robbers and other unfriendly folk, so it's dangerous to travel far. As Thunder Mountain's head of security, Lee thinks that it is unsafe to carry supplies for more than 10km without visiting a town. Markus wants to know how far one would need to travel to get from one town to another in the worst case.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing **n** (the number of towns, 1<**n**<101). The next **n** lines will give the xy-locations of each town in km (integers in the range [0, 1023]). Assume that the Earth is flat and the whole 1024x1024 grid is covered by a forest with roads connecting each pair of towns that are no further than 10km away from each other.

Output

For each test case, output the line "Case #**x**:", where **x** is the number of the test case. On the next line, print the maximum distance one has to travel from town A to town B (for some A and B). Round the answer to 4 decimal places. Every answer will obey the formula

fabs(ans*1e4 - floor(ans*1e4) - 0.5) > 1e-2

If it is impossible to get from some town to some other town, print "Send Kurdy" instead. Put an empty line after each test case.

Sample Input	**Sample Output**
2 5 0 0 10 0 10 10 13 10 13 14 2 0 0 10 1	Case #1: 25.0000 Case #2: Send Kurdy

Problem E

Gopher Strategy

Time Limit: 3 seconds

Agent Cooper: *"Look at that! Ducks... on the lake!"*

Harley Peyton, "Twin Peaks."

Gophers like to feed in the field, but they always have to look out for hawks that might hunt them. A group of gophers have decided to get more organized and need your help developing an escape strategy in case of a hawk attack.

Given the coordinates of **m** gophers and **n** holes in the field, what is the minimum time required for each gopher to reach a hole (at most one gopher per hole)? Every gopher runs in a straight line at a speed of 1 unit per second, and the group can tolerate the loss of at most **k** gophers. (Gophers are lost when they do not have enough time to reach an empty hole.)

Input

The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing the integers **m**, **n** and **k** (0 <= **m**, **n** <= 50, 0 <= **k** <= **m**). The next **m** lines will give the x,y-coordinates of the gophers. The **n** lines after that will give the coordinates of the holes.

Output

For each test case, output the line "Case #**x**:", where **x** is the number of the test case. Then print the minimum number of seconds required for at least **m-k** gophers to reach a hole, rounded to 3 decimal places. Print "Too bad." if there is no solution. Print an empty line after each test case.

Sample Input	Sample Output
5 3 3 1 0 0 1 0 2 0 0 1 1 1 2 1.5 3 3 1 0 1 1 2 2 1 1 0 1 1 2 0 3 3 0 0 1 1 2 2 1 1 0 1 1 2 0 1 0 0 100.0 200.5 1 0 2 123.4 234.4	Case #1: 1.000 Case #2: 1.000 Case #3: 1.414 Case #4: Too bad. Case #5: 0.000

Problemsetter: Igor Naverniouk
Alternate solution: Yury Kholondyrev

Problem F
Cockroach Escape Networks
Time Limit: 3 seconds

"Bug powder dust an' mugwump jism
And the wild boys runnin' 'round Interzone trippin'"

Justin Warfield, Bomb the Bass

Bill Lee shares his apartment with a group of cockroaches. The bugs are smart and have several nests inside the apartment. When they are inside one of the nests, Bill can not catch them. Some pairs of nests are connected by cockroach trails, and it takes one unit of time to run from one nest along a trail to any neighbouring nest. However, it takes a lot of the cockroaches' resources to maintain all of the trails in good condition. What they need is to destroy some of the trails, but still make sure that it is possible to run from any nest to any other nest along a sequence of trails.

There are **n** nests in the room, and each nest has at least **n**-1 cockroaches in it at any moment. In case of emergency (when Bill comes into the room and turns on the light), the roaches go into a state of panic - from every nest, **n**-1 roaches strat running, one to every other nest along the trails. Several roaches can run along the same trail without interfering with each other. The time it takes for the last cockroach to reach its destination is called the Emergency Response Time. The cockroaches are smart and always choose the shortest path.

Your task is, given a description of the cockroaches' network, find the set, T, of trails that need to be kept so that it is possible to reach any nest B from any nest A along a path in T. If there are multiple such sets, pick the one that has the fewest trails. If there is still a tie, pick the one that guarantees the smallest Emergency Response Time. Print that time.

Input
The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing the integers **n** (the number of nests) and **m** (the number of trails). The next **m** lines will give the pairs of nests that are connected by a trail. The nests are numbered from 0 to **n**-1. **n** is at most 25. There will be no trails from a node to itself and no duplicate trails.

Output
For each test case, output the line "Case #**x**:", where **x** is the number of the test case. On the next line, print the smallest possible Emergency Response Time. Print an empty line after each test case.

Sample Input	Sample Output
2 4 4 0 1 1 2 2 3 3 0 5 7 0 1 1 4 0 2 1 2 1 3 4 3 2 3	Case #1: 3 Case #2: 2

Problemsetter: Igor Naverniouk

Problem G

Dijkstra, Dijkstra.

Time Limit: 10 seconds

Dexter: *"You don't understand. I can't walk... they've tied my shoelaces together."*
Topper Harley: *"A knot. Bastards!"*

Jim Abrahams and Pat Proft,
"Hot Shots! Part Deux."

You are a political prisoner in jail. Things are looking grim, but fortunately, your jailmate has come up with an escape plan. He has found a way for both of you to get out of the cell and run through the city to the train station, where you will leave the country. Your friend will escape first and run along the streets of the city to the train station. He will then call you from there on your cellphone (which somebody smuggled in to you inside a cake), and you will start to run to the same train station. When you meet your friend there, you will both board a train and be on your way to freedom.

Your friend will be running along the streets during the day, wearing his jail clothes, so people will notice. This is why you can not follow any of the same streets that your friend follows - the authorities may be waiting for you there. You have to pick a completely different path (although you may run across the same intersections as your friend).

What is the earliest time at which you and your friend can board a train?

Problem, in short

Given a weighed, undirected graph, find the shortest path from **S** to **T** and back without using the same edge twice.

Input

The input will contain several test cases. Each test case will begin with an integer **n** (2<=**n**<=100) - the number of nodes (intersections). The jail is at node number 1, and the train station is at node number **n**. The next line will contain an integer **m** - the number of streets. The next **m** lines will describe the **m** streets. Each line will contain 3 integers - the two nodes connected by the street and the time it takes to run the length of the street (in seconds). No street will be longer than 1000 or shorter than 1. Each street will connect two different nodes. No pair of nodes will be directly connected by more than one street. The last test case will be followed by a line containing zero.

Output

For each test case, output a single integer on a line by itself - the number of seconds you and your friend need between the time he leaves the jail cell and the time both of you board the train. (Assume that you do not need to wait for the train - they leave every second.) If there is no solution, print "Back to jail".

Sample Input	Sample Output
2 1 1 2 999 3 3 1 3 10 2 1 20 3 2 50 9 12 1 2 10 1 3 10 1 4 10 2 5 10 3 5 10 4 5 10 5 7 10 6 7 10 7 8 10 6 9 10 7 9 10 8 9 10 0	Back to jail 80 Back to jail

Problemsetter: Igor Navernіouk

Problem H

Prim, Prim.

Time Limit: 3 seconds

Xander: *"Calax Research and Development. It's a computer research lab. Third largest employer in Sunnydale till it closed down last year. What, I can't have information sometimes?"*
Giles: *"Well, it-it's just somewhat unprecedented."*

Ashley Gable and Thomas A. Swyden,
"Buffy the Vampire Slayer."

Calax Research and Development is a large high tech corporation. They have an antitrust lawsuit on their hands because they are too big. The judge has ordered that the corporation be split into two new companies, A and B.

Calax has a large network of communication lines that connect a number of cities (each city is connected to every other city by a path of communication lines). They now need to split those lines into two sets, A and B. It is important that each of the two sets still connects all of the cities because the two companies will not be allowed to share communication lines. It has also been decided that all redundant lines will be sold off to protect the two new companies from more antitrust lawsuits. And of course, the total cost of this operation needs to be as small as possible.

Input

The input will contain a number of cases. Each case will begin with **n** - the number of cities (at most 10), followed by **m** - the number of communication lines (at most 25). The next **m** lines will contain 3 numbers each - the two cities connected by the line and the cost of keeping the line. Each city will be identified by an integer in the range [1,**n**]. The cost of a line is at most 1000. The input will be terminated by the case where **n** is zero.

Output

Output one integer per test case on a line by itself - the minimum total cost of the communication lines in sets A and B. Print "No way!" if there is no solution.

Sample Input	Sample Output
2 3 1 2 10 2 1 20 1 2 30 3 3 1 2 10 1 2 20 2 3 50 5 8 1 2 10 1 3 10 1 4 10 1 4 20 1 5 20 2 3 20 3 4 20 4 5 30 0	30 No way! 140

Comments

In the third test case, one possible solution is to let company A keep these communication lines:

1 2 10
1 3 10
1 4 10
1 5 20
and let company B keep these ones:
1 4 20
2 3 20
3 4 20
4 5 30
for a total cost of 50 + 90 = 140.

Problemsetter: Igor Naverniouk

Problem I
Rational Resistors
Time Limit: 15 seconds

"Resistance will be punished.
Cooperation will be rewarded."

J. Michael Straczynski, "Babylon 5" and "Jeremiah".

Given a network of resistors, what is the equivalent resistance between two given points in the network? More precisely, consider an undirected, weighted graph, where each edge is a wire with the edge weight representing its resistance, in Ohms. Given a pair of nodes, A and B, in this graph, imagine passing 1 Ampere of current from A to B. What will be the voltage between A and B, in Volts?

A brief review of high school physics. For any pair of points, P and Q, in the network, the voltage between the points is the difference in potentials at the two points (V(P) - V(Q)) and is equal to the current from P to Q times the resistance between P and Q. For any point in the network, the sum of the currents entering the point is zero (conservation of charge).

Warning! This problem is harder than it seems.

Input
The first line of input gives the number of cases, **N** (**N**<30). **N** test cases follow. Each one starts with a description of a graph:

n m (the number of nodes and wires in the graph)
n will not be larger than 16. The next **m** lines contain 3 integers each:
u v r
specifying that there is a wire with resistance **r** (0<**r**<10) connecting node **u** to node **v**. The nodes are numbered from 0 to **n**-1. There can be multiple wires connecting the same pair of nodes and wires connecting a node to itself. The next line of each test case will contain the number of queries, **Q** (0<=**Q**<=10). The next **Q** lines will list pairs of nodes **A** and **B**.

Output
For each test case, output the line "Case #**x**:", where **x** is the number of the test case. Then print **Q** lines of the form "Resistance between **A** and **B** is **s/t**.", where **s/t** is a fraction in lowest terms. Print "1/0" if no current can go from **A** to **B**. Finally, print an empty line after each test case.

Sample Input	Sample Output
4 3 2 0 1 1 1 2 2 2 0 2 1 0 2 2 0 1 1 1 0 2 1 0 1 2 0 1 0 1 4 4 0 1 1 1 2 2 2 3 3 1 3 1 1 0 3	Case #1: Resistance between 0 and 2 is 3/1 Resistance between 1 and 0 is 1/1 Case #2: Resistance between 0 and 1 is 2/3 Case #3: Resistance between 0 and 1 is 1/0 Case #4: Resistance between 0 and 3 is 11/6

Problemsetter: Igor Naverniouk