Problem A: Lockpicking

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The Global Burglar Federation is currently holding a convention in your city and hosting a series of events to mark the occasion. One of them is the lockpicker maze:

The participants compete one after another. Each participant starts in the first of a large number of rooms (numerated from 1 to N), some of which are connected by one-way doors. While all doors are closed when not in use, only some of them are locked in the beginning (and thus have to be broken before they can be used). The competition's goal is to cross the maze (i.e. reach the last room) as fast as possible.

As a big fan of mazes you just have to participate. Unfortunately you are not very fast at lockpicking, so you want to optimize your route before you start, in order to minimize the number of locks you have to pick.

Input

The input consists of

- one line containing two integers N and M ($2 \leq N \leq 10^5, 1 \leq M \leq 10^6$) – the amounts of rooms and doors, respectively
- M lines describing the doors, with the i-th line containing numbers u_i, v_i and a character l_i ($1 \leq u_i, v_i \leq N$, $\ell_i \in \{\texttt{'l'}, \texttt{'u'}\}$), indicating a door from room u_i to room v_i that is either unlocked ($\ell_i = u$) or locked ($\ell_i = l$).

Output

Output the minimum number of locks you need to pick to get from room 1 to room N. If you can't reach room N at all, output -1.

Sample input and output

Input	Output
4 6 1 2 l 1 3 u 2 3 u 3 2 u 2 4 u 3 4 l	0
9 9 1 2 u 2 3 l 3 4 l 4 8 u 1 5 l 5 6 u 6 7 u 7 8 u 8 9 l	2

Problem B: Strategy Game

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You recently stumbled upon an old single player strategy game and naturally began to analyze it mathematically. The game can only ever be in one of its S possible states (which you numerated from 1 to S). With each turn, the player changes the state of the game and receives a certain nonnegative number of penalty points for that action (the exact amount depends solely on which state the game was in before and which it is in afterwards; not all transitions are allowed). The game ends as soon as it reaches one of its "final states" and the player's single goal is to achieve this while obtaining as few penalty points as possible.

The only thing you still need to find out to complete your analysis of the game is how many penalty points one receives in a perfect game (i.e. you want to know what's the minimum amount of penalty points one can have when reaching a final state).

Input

The input consists of

- one line containing S, T and F ($2 \leq S \leq 5000$, $1 \leq T \leq 5 \cdot 10^4$, $1 \leq F \leq 500$) – the amounts of states, permitted state transitions and final states, respectively
- one line containing F numbers $f_1, ..., f_F$ ($1 \leq f_i \leq S$), the indices of the final states
- T lines describing the permitted state transitions, with the i-th line containing b_i, a_i and p_i ($1 \leq b_i, a_i \leq S, 1 \leq p_i \leq 1000$) – the state the game is in before that turn, the state it is in afterwards and how many penalty points the player receives for that turn. In the beginning the game is always in state 1.

Output

Output two numbers, the minimum amount of penalty points one can have when reaching a final state and the index of the final state one has to approach to achieve this. If there are multiple possibilities for the latter, output the smallest one. If it is not possible to reach a final state at all, output "IMPOSSIBLE".

Sample input and output

Input	**Output**
5 5 2 3 5 1 2 12 2 3 10 2 4 7 4 3 2 1 5 21	21 3
2 0 1 2	IMPOSSIBLE

Problem C: Navigation System

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You suspect your navigation system of picking inefficient routes sometimes, so you decided to collect some data about the road system in your region and write a program that calculates the shortest (w.r.t. combined road length) as well as the fastest (wrt. expected travel time) routes from your home to certain other places.

To make this a manageable task, you came up with an easy model: The road system consist of junctions (numerated from 1 to N) and roads connecting them, and every road has a speed limit (in km/h) as well as a length (in km). Switching roads at one of the junctions costs neither time nor distance. For the sake of simplicity you also assume all roads to be bidirectional and that you can always drive at speed limit.

Input

The input consists of

- one line containing N and M ($2 \leq N \leq 5 \cdot 10^4, 1 \leq M \leq 2 \cdot 10^5$) – the amounts of junctions and roads, respectively
- M lines describing the roads, with the i-th line containing b_i, e_i, v_i and ℓ_i ($1 \leq b_i, e_i \leq N, 20 \leq v_i \leq 150, 1 \leq \ell_i \leq 300$) – the junctions the i-th road connects, its speed limit and its length (all these inputs are integers)
- one line containing Q ($1 \leq Q \leq 1000$) – the number of queries
- Q lines giving the queries, each consisting of an integer j – the index of the destination junction – and a character $c \in \{\texttt{'s'}, \texttt{'f'}\}$ – the criterion according to which the route should be optimized (see below)

Output

For `'s'`-type queries, output the length of the shortest route from your home (junction 1) to the given destination. For `'f'`-type queries, output the minimum travel time from your home to the given destination (assuming driving at speed limit at all times) with precision 10^{-3}. Use the same units as the input and separate quantities and their units by spaces. If a given destination cannot be reached, output "IMPOSSIBLE".

Sample input and output

Input	**Output**
3 3 1 2 50 10 2 3 120 30 1 3 40 20 2 3 s 3 f	20 km 0.4500 h
3 1 1 2 150 1 1 3 f	IMPOSSIBLE

Problem D: Worm Holes

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

As a researcher for the new Galactic Guide it is your responsibility to calculate the distances between all important stars in a certain sector of the galaxy. What makes this a difficult task is the circumstance that there are a number of passable worm holes in the sector that connect some otherwise distant points with no distance whatsoever, which needs to be accounted for.

Input

The input consists of

- one line containing N and M ($2 \leq N \leq N + M \leq 500$) – the amounts of stars and worm holes, respectively
- $N + M$ lines containing the (all-integer) coordinates of the stars and worm holes (i.e. the first N lines describe the stars, the last M lines the worm holes) in the format "x_i y_i z_i" ($-10^4 \leq x_i, y_i, z_i \leq 10^4$). There is a worm hole connection between the first and second worm hole (numerated as given), one between the third and fourth, and so on (M is guaranteed to be even).

Output

Output N lines of N floating point numbers each, with the i-th number in the j-th line stating the distance between the i-th and j-th star (according to the order in which they are given). The precision should be at least 10^{-3}.

Sample input and output

Input	Output
2 2 0 0 0 10000 0 0 1 0 0 9996 0 0	0.0000 5.0000 5.0000 0.0000
3 2 -6 2 -5 -2 -2 -2 7 5 3 5 5 5 -5 -5 -5	0.00000 6.40312 9.89949 6.40312 0.00000 8.02458 9.89949 8.02458 0.00000

Problem E: The Maze of Knowledge

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

On the special event from week 4, tickets for the "maze of knowledge" were given to the guests. The maze consists of N rooms numerated as $1, .., N$ and there are doors between some of them. Whenever someone inside the maze goes through one of the doors his knowledge index changes by a certain value that is painted on that door. The entrance to the maze is located in room 1 and the exit in room N. Every participant enters the maze once and exits with some gained (or lost) knowledge (in the begining everyones knowledge index is 0). Your task is to cross the maze on a path that maximizes your knowledge index.

Input

The input consists of

- one line containing N ($1 \leq N \leq 2000$) – the number of rooms – and M ($1 \leq M \leq 10^4$) – the number of doors
- M lines contain the door descriptions, with the i-th line containing integers b_i, a_i and v_i ($1 \leq a_i, b_i \leq N$, $-10^4 \leq v_i \leq 10^4$) – the room you are in before going through the door, the room you are in afterwards (you can only go through in one direction) and the value painted on the door. A door can lead from a room into the same room and there might be several doors between two rooms.

Output

Output ":)" if you can leave the maze with unlimited amounts of knowledge and ":(" if it's impossible to cross the maze at all. Otherwise output the maximal possible knowledge to be gained before leaving the maze.

Sample input and output

Input	Output
2 2 1 2 -5 1 1 1	:)

Problem F: Fallout shelter

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

What you had been secretly expecting for basically your entire life finally just happened: Some depraved ruler pushed the button and started a nuclear world war. The first bombs are already headed for your country, but there is some hope: Since the global political situation had already been escalating for a while, your city wisely built a number of well-equipped fallout shelters to protect its inhabitants in a situation like this. Unfortunately you are not sure which of them is closest and how long it would take you to get there.

The city consists of M streets and N junctions, where each street connects two distinct junctions to one another. Since, understandably, you are in full panic mode, you intend to run the entire way, starting out with velocity v. However, as you are not very well-trained right now, you expect your velocity to decrease by 10% after each street you ran along (so that you'd run the first street with velocity v, the second with $0.9v$, the third with $0.81v$ etc.).

But before you even go to the trouble of trying to save yourself, you want to know if you even have a chance to reach one of the shelters before it's too late.

Input

The input consists of

- one line containing N ($2 \leq N \leq 250$) – the number of junctions – M ($N - 1 \leq M \leq 10^4$) – the number of roads between them – and F ($1 \leq F \leq 10$) – the number of fallout shelters
- one line containing an integer c ($1 \leq c \leq N$) – the number of the junction you are currently standing on – and a float v ($1.0 \leq v \leq 10.0$) – your initial speed in meters per second
- one line containing F numbers $s_1, ..., s_F$ ($1 \leq s_i \leq N$) – the indices of the fallout shelters

- M lines containing descriptions of the streets, each of them consisting of three integers v, u and l ($1 \leq u, v \leq N,\ 1 \leq l \leq 1000$), denoting that there's a street between junctions v and u that is l meters long. Crossing a junction doesn't cost any extra time, since their size is already factored into the street lengths. Also, it is guaranteed that each two junctions are connected to one another through a series of streets.

Output

Output the minimal amount of time you would need to reach one of the fallout shelters (in seconds), with a relative error of at most 10^{-6}.

Sample input and output

Input	**Output**
3 3 1 1 1.0 3 1 2 1 2 3 3 3 1 5	4.33333333