Problem A. Chocolate

Input file: `chocolate.in`
Output file: `chocolate.out`

Mom gave Bitie and Bytie a bar of chocolate. The bar has the shape of a $n \times m$ rectangle divided into $n \cdot m$ unit squares. Instead of simply dividing it into halves, the boys decided to play a game using it. They labeled one of the unit squares with a cross. The boys take turns playing the game. Bitie starts the game. Each move consists in breaking the bar along a horizontal or vertical line which does not cross any of the unit squares and eating one of the resulting pieces of chocolate. The player who eats the labeled square loses.

The game was quite exciting and Bytie turned out to be the winner. Bitie is not happy with this outcome. He would like to know to what extent the choice of the labeled square influenced the fact that he was defeated. Bitie is interested in counting for how many of the nm unit squares, the game ends with Bitie's loss if the boys choose to label that square. We assume that the boys play optimally.

Input

The first line of the input contains one integer t ($1 \leqslant t \leqslant 10\,000$) denoting the number of test cases. The following t lines each contain two integers n, m ($1 \leqslant n, m \leqslant 10^9$) denoting the dimensions of the bar of chocolate.

Output

Your program should output t lines with the answers to the consecutive test cases. The i-th line should contain one integer denoting the number of unit squares such that Bitie loses the game if the given square is labeled in the beginning.

Examples

chocolate.in	chocolate.out
2 1 2 2 6	0 4

Problem B. Programming Contest

Input file: `contest.in`
Output file: `contest.out`

Bartie and his friends compete in the Team Programming Contest. There are n contestants on each team, and each team has access to n computers. The contest lasts t minutes, during which the contestants are to solve m programming problems. Furthermore, penalties are imposed on the teams: solving a problem s minutes since the beginning of the contest amounts to s penal points. The team that solved the most problems wins the contest, with ties broken in favour of the team with smaller penalty.

On the contest day Bartie quickly glances over the problem statements and distributes them among his teammates. He knows his team so well that he can exactly assess who is able to solve which problem. Solving any problem takes any contestant that is able to solve it exactly r minutes of using the computer.

Bartie's team did not fare well in this year's contest. Bartie is obsessed with the thought that it might be his fault, due to wrong decisions regarding the distribution of problems. He asks you to write a program that, given what Bartie knew at the beginning of the contest, determines the best possible result of Bytie's team, together with the assignment of problems to team members that attains the result.

Input

Five integers n, m, r, t, and k ($1 \leqslant n, m \leqslant 500$, $1 \leqslant r, t \leqslant 1\,000\,000$) are given in the first line of the input, separated by single spaces. These denote, respectively: the number of contestants on a team, the number of problems, the time it takes a contestant to solve a problem, the duration of the contest, and the number of contestant–problem pairs given on the input. Each of the following k lines holds two integers a and b ($1 \leqslant a \leqslant n$, $1 \leqslant b \leqslant m$), separated by a single space, denoting that the contestant a is able to solve the problem b. Each such pair appears at most once in the input.

Output

In the first line of the output the best possible result of Bytie's team should be printed as two numbers separated by a single space: the number of solved problems z and the total penal points. An exemplary assignment of problems that attains this result should be given in the following z lines. Each of those should hold three integers a, b and c ($1 \leqslant a \leqslant n$, $1 \leqslant b \leqslant m$, $0 \leqslant c \leqslant t - r$), separated by single spaces, signifying that the contestant a should start solving the problem b at time c (the contest starts at time 0). No contestant should be assigned a problem that they cannot solve. If more that one optimal assignment exists, your program can output any of them.

Examples

contest.in	contest.out
2 4 3 15 4 1 1 2 3 1 4 1 3	3 12 1 4 0 2 3 0 1 1 3

Problem C. Drawing

Input file: `drawing.in`
Output file: `drawing.out`

Byteman drew a convex polygon with n vertices on a sheet of paper. He labeled the vertices with integers from 1 to n. The labels were assigned in an arbitrary order. Afterwards, Byteman drew a number diagonals of the polygon. The diagonals did not intersect each other. Several diagonals could, however, meet in a single vertex of the polygon. Byteman enjoyed the drawing very much, so he wrote down the numbers of pairs of vertices connected by line segments.

After some time Byteman was willing to recover the drawing using his notes. Surprisingly, this turned out to be quite complex. He asked you to for help in writing a program which will solve his problem.

Input

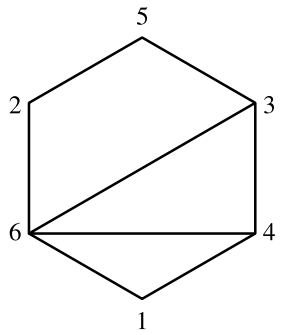
The first line of the input contains two integers n and m ($3 \leqslant n \leqslant 500\,000$, $n \leqslant m \leqslant 2n-3$) denoting the number of vertices of the polygon and the number of pairs of vertices connected by line segments. Each of the following m lines contains a pair of integers a_i, b_i ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$), which represents a line segment connecting the vertices a_i and b_i. Each unordered pair $\{a_i, b_i\}$ occurs in the input at most once.

Output

The only line of the output should contain n integers: the labels of the consecutive vertices on the perimeter of the polygon. Among all possible outputs, your program should choose the one which starts with the vertex number 1 and for which the second element of the sequence is the smallest possible.

Examples

drawing.in	drawing.out
6 8 1 4 1 6 4 6 3 6 2 5 2 6 3 4 3 5	1 4 3 5 2 6



Problem D. Evacuation

Input file: `evacuation.in`
Output file: `evacuation.out`

The next year's World Byteball Cup finals will be held in Byteland. Byteman participates in organizing the event, and he is responsible for creating an evacuation plan from each byteball stadium where tournament matches will be held. Byteman has encountered particular problems with preparing an evacuation plan for Bytetown. He keeps receiving updates about which streets will be closed during the tournament due to roadworks. On the other hand, from time to time he needs to answer questions from the members of Bytean Byteball Association about how quick people can be evacuated from a given stadium to a given junction located in its town. Write a program which will help Byteman in answering these questions.

In Bytetown there are n junctions connected with unidirectional streets. The evacuation may only lead along the streets, and in the same direction as of the corresponding street. Reaching the end of a street starting from its beginning takes exactly one minute.

Input

The first line of the input contains three integers n, m and q ($1 \leqslant n \leqslant 1\,000, 1 \leqslant m \leqslant 100\,000$, $1 \leqslant q \leqslant 200\,000$) which denote the number of junctions in Bytetown, the number of streets and the number of queries, respectively. The junctions are numbered from 1 to n. The stadium is located at the junction number 1. Each of the following m lines contains a description of a street: a pair of integers a_i, b_i ($1 \leqslant a_i, b_i \leqslant n, a_i \neq b_i$) representing a unidirectional street from a_i to b_i. Each pair of junctions is connected by at most one street in each of the directions.

The following q lines contain descriptions of queries given in a time-based order. Each query is formed of a single character t_i ($t_i \in \{\texttt{U}, \texttt{E}\}$) followed by an integer p_i.

A query starting with the character $t_i = \texttt{U}$ describes a street which is being closed for roadworks. In this case, the number p_i belongs to the interval $[1, m]$ and denotes the number of the street which is being closed. The streets are numbered as they appear in the input, any street cannot be closed twice.

A query starting with the character $t_i = \texttt{E}$ denotes a question about the quickest evacuation scheme to the junction number p_i (in this case we have $2 \leqslant p_i \leqslant n$). You can assume that the input will contain at least one query of the type `E`.

Output

For each query of type `E` our program should output one line. The answer to every such query is the shortest time needed for getting from the stadium (junction number 1) to the junction number p_i, using only the streets which have not been closed yet. If it is impossible to reach the junction p_i, the correct answer is -1.

Examples

evacuation.in	evacuation.out
7 8 8 1 2 1 3 1 5 2 4 3 1 3 5 4 5 4 6 E 7 E 5 U 7 E 6 E 5 U 2 E 5 E 4	-1 1 3 1 1 2

Problem E. Inspection

Input file: `inspection.in`
Output file: `inspection.out`

The railway network of the Byteotian Railways (BR) consists of bidirectional tracks connecting certain pairs of stations. Each pair of stations is connected by at most one segment of tracks. Furthermore, there is a unique route from every station to every other station. (The route may consist of several segments of tracks, but it may not pass through any station more than once.)

Byteasar is an undercover inspector of the BR. His job is to pick one of the stations (denote it by S) for centre of his operations and to travel to all other stations. His journey should be as follows:

- Byteasar starts in station S.
- Next, he picks one of the stations he did not yet control and goes to it along the shortest path (by train, of course), inspects the station, and then goes back to S.
- The crooked employees of BR warn one another of Byteasar's comings. To deceive them, Byteasar picks the next station for control in such a way that he sets off from the station S in different direction than the last time, i.e., along a different segment of tracks leaving from S.
- Each station (except S) is inspected exactly once.
- After inspecting the last station Byteasar **does not come back** to S.

The travel time along every segment of tracks takes the same amount of time: one hour.

Byteasar intends to consider all the stations as the initial station S. For each of them he wants to know the order of inspecting the remaining stations that minimises the total travel time, provided that it is possible at all for that particular S.

Input

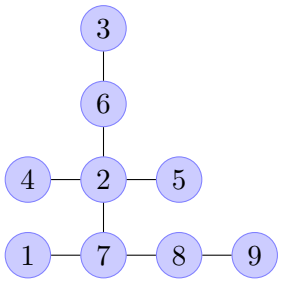
The first line of the input contains a single integer n ($1 \leqslant n \leqslant 1\,000\,000$) that denotes the number of stations. These are numbered from 1 to n. The following $n-1$ lines specify the track segments, one per line. Each of them holds two integers a, b ($1 \leqslant a, b \leqslant n$, $a \neq b$), separated by a single space, indicating that there is a track segment connecting the stations a and b. Each track segments appears exactly once in the description.

Output

Your program should print n lines on the output, each holding a single integer. The one in the i-th line should be the minimum number of hours Byteasar has to spend travelling to inspect the stations when $S = i$ — if inspecting them all is possible for $S = i$; if it is not, the i-th line should hold the number -1.

Examples

inspection.in	inspection.out
9 3 6 2 4 2 6 2 5 1 7 2 7 8 9 7 8	-1 23 -1 -1 -1 -1 -1 -1 -1



The figure illustrates the railway network as specified in the example. It is possible to inspect all the stations only for $S = 2$. One optimal order of inspection is: 7, 4, 8, 6, 1, 5, 3, 9. It results in 23 hours of travel.

Problem F. Meteors

Input file: `meteors.in`
Output file: `meteors.out`

Byteotian Interstellar Union (BIU) has recently discovered a new planet in a nearby galaxy. The planet is unsuitable for colonisation due to strange meteor showers, which on the other hand make it an exceptionally interesting object of study.

The member states of BIU have already placed space stations close to the planet's orbit. The stations' goal is to take samples of the rocks flying by. The BIU Commission has partitioned the orbit into m sectors, numbered from 1 to m, where the sectors 1 and m are adjacent. In each sector there is a single space station, belonging to one of the n member states.

Each state has declared a number of meteor samples it intends to gather before the mission ends. Your task is to determine, for each state, when it can stop taking samples, based on the meter shower predictions for the years to come.

Input

The first line of the input gives two integers, n and m ($1 \leqslant n, m \leqslant 300\,000$), separated by a single space, that denote, respectively, the number of BIU member states and the number of sectors the orbit has been partitioned into.

In the second line there are m integers o_i ($1 \leqslant o_i \leqslant n$), separated by single spaces, that denote the states owning stations in successive sectors.

In the third line there are n integers p_i ($1 \leqslant p_i \leqslant 10^9$), separated by single spaces, that denote the numbers of meteor samples that the successive states intend to gather.

In the fourth line there is a single integer k ($1 \leqslant k \leqslant 300\,000$) that denotes the number of meteor showers predictions. The following k lines specify the (predicted) meteor showers chronologically. The i-th of these lines holds three integers l_i, r_i, a_i (separated by single spaces), which denote that a meteor shower is expected in sectors $l_i, l_{i+1}, \ldots, r_i$ (if $l_i \leqslant r_i$) or sectors $l_i, l_{i+1}, \ldots, m, 1, \ldots, r_i$ (if $l_i > r_i$), which should provide each station in those sectors with a_i meteor samples ($1 \leqslant a_i \leqslant 10^9$).

Output

Your program should print n lines on the output. The i-th of them should contain a single integer w_i, denoting the number of shower after which the stations belonging to the i-th state are expected to gather at least p_i samples, or the word `NIE` (Polish for *no*) if that state is not expected to gather enough samples in the foreseeable future.

Examples

meteors.in	meteors.out
3 5 1 3 2 1 3 10 5 7 3 4 2 4 1 3 1 3 5 2	3 NIE 1

Problem G. Party

Input file: `party.in`
Output file: `party.out`

Byteasar intends to throw up a party. Naturally, he would like it to be a success. Furthermore, Byteasar is quite certain that to make it so it suffices if all invited guests know each other. He is currently trying to come up with a list of his friends he would like to invite.

Byteasar has n friends, where n is divisible by 3. Fortunately, most of Byteasar's friends know one another. Furthermore, Byteasar recalls that he once attended a party where there were $\frac{2}{3}n$ of his friends, and where everyone knew everyone else. Unfortunately, Byteasar does not quite remember anything else from that party... In particular, he has no idea which of his friends attended it.

Byteasar does not feel obliged to throw a huge party, but he would like to invite at least $\frac{n}{3}$ of his friends. He has no idea how to choose them, so he asks you for help.

Input

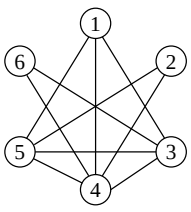
In the first line of the input two integers, n and m ($3 \leqslant n \leqslant 3\,000$, $\frac{\frac{2}{3}n(\frac{2}{3}n-1)}{2} \leqslant m \leqslant \frac{n(n-1)}{2}$), are given, separated by a single space. These denote the number of Byteasar's friends and the number of pairs of his friends who know each other, respectively. Byteasar's friends are numbered from 1 to n. Each of the following m lines holds two integers separated by a single space. The numbers in line no. $i+1$ (for $i = 1, 2, \ldots, m$) are a_i and b_i ($1 \leqslant a_i < b_i \leqslant n$), separated by a single space, which denote that the persons a_i and b_i know each other. Every pair of numbers appears at most once on the input.

Output

In the first and only line of the output your program should print $\frac{n}{3}$ numbers, separated by single spaces, in increasing order. These number should specify the numbers of Byteasar's friends whom he should invite to the party. As there are multiple solutions, pick one arbitrarily.

Examples

party.in	party.out
6 10 2 5 1 4 1 5 2 4 1 3 4 5 4 6 3 5 3 4 3 6	2 4



Explanation of the example: Byteasar's friends numbered 1, 3, 4, 5 know one another. However, any pair of Byteasar's friends who know each other, like 2 and 4 for instance, constitutes a correct solution, i.e., such a pair needs not be part of aforementioned quadruple.

Problem H. Periodicity

Input file: `periodicity.in`
Output file: `periodicity.out`

Byteasar, the king of Bitotia, has ordained a reform of his subjects' names. The names of Bitotians often contain repeating phrases, e.g., the name `Abiabuabiab` has two occurrences of the phrase `abiab`. Byteasar intends to change the names of his subjects to sequences of bits matching the lengths of their original names. Also, he would very much like to reflect the original repetitions in the new names.

In the following, for simplicity, we will identify the upper- and lower-case letters in the names. For any sequence of characters (letters or bits) $w = w_1 w_1 \ldots w_k$ we say that the integer p $(1 \leqslant p < k)$ is a period of w if $w_i = w_{i+p}$ for all $i = 1, \ldots, k - p$. We denote the set of all periods of w by $Per(w)$. For example, $Per(\texttt{ABIABUABIAB}) = \{6, 9\}$, $Per(\texttt{01001010010}) = \{5, 8, 10\}$, and $Per(\texttt{0000}) = \{1, 2, 3\}$.

Byteasar has decided that every name is to be changed to a sequence of bits that:

- has length matching that of the original name,
- has the same set of periods as the original name,
- is the smallest (lexicographically[1]) sequence of bits satisfying the previous conditions.

For example, the name `ABIABUABIAB` should be changed to `01001101001`, `BABBAB` to `010010`, and `BABURBAB` to `01000010`.

Byteasar has asked you to write a program that would facilitate the translation of his subjects' current names into new ones. If you succeed, you may keep your current name as a reward!

Input

In the first line of the input there is a single integer k — the number of names to be translated $(1 \leqslant k \leqslant 20)$. The names are given in the following lines, one in each line. Each name consists of no less than 1 and no more than 200 000 upper-case (capital) letters (of the English alphabet).

Output

Your program should print k lines to the output. Each successive line should hold a sequence of zeroes and ones (without spaces in between) corresponding to the names given on the input. If an appropriate sequence of bits does not exists for some names, then "`XXX`" (without quotation marks) should be printed for that name.

Examples

periodicity.in	periodicity.out
3 ABIABUABIAB BABBAB BABURBAB	01001101001 010010 01000010

[1] The sequence of bits $x_1 x_2 \ldots x_k$ is lexicographically smaller than the sequence of bits $y_1 y_2 \ldots y_k$ if for some i, $1 \leqslant i \leqslant k$, we have $x_i < y_i$ and for all $j = 1, \ldots, i - 1$ we have $x_j = y_j$.

Problem I. Sticks

Input file: `sticks.in`
Output file: `sticks.out`

Little Johnny was given a birthday present by his grandparents. This present is a box of sticks of various lengths and colours. Johnny wonders if there are three sticks in the set he has been given that would form a triangle with different-coloured sides. Let us note that Johnny is interested in non-degenerate triangles only, i.e., those with positive area.

Input

In the first line of the input an integer k ($3 \leqslant k \leqslant 50$) is given, which is the number of different colours of sticks. The colours themselves are numbered from 1 to k.

The following k lines contain descriptions of the sticks of particular colours. The line no. $i + 1$ holds integers that describe the sticks of colour i, separated by single spaces. The first of these numbers, n_i ($1 \leqslant n_i \leqslant 1\,000\,000$), denotes the number of sticks of colour i. It is followed, in the same line, by n_i integers denoting the lengths of the sticks of colour i. All lengths are positive and do not exceed $1\,000\,000\,000$. Furthermore, the total number of all sticks does not exceed $1\,000\,000$.

Output

Your program should print (on the first and only line of the output) either:

- six integers, separated by single spaces, that describe the construction of a triangle with different-coloured sides as follows: the colour and the length of the first stick, the colour and the length of the second stick, and the colour and the length of the third stick,
- or the word `NIE` (Polish for *no*) if no such triple of sticks exists.

If there are multiple triples of different-coloured sticks that give rise to a triangle, your program may pick one such triple arbitrarily.

Examples

sticks.in	sticks.out
4 1 42 2 6 9 3 8 4 8 1 12	3 8 4 12 2 9
3 1 1 1 2 1 3	NIE

Problem J. Trams

Input file: `trams.in`
Output file: `trams.out`

Due to increasing prices of electricity, the mayor of Bytetown was forced to perform severe budget cuts. As a result of these cuts, the tram service in Bytetown was totally closed. The citizens of Bytetown are quite upset with this situation. The only hope for improvement is applying for a subsidy to the Ministry of Infrastructure. You are responsible for preparing such an application.

The tram network in Bytetown consists of n junctions and $n-1$ track segments. Each junction is reachable from every other junction along this network. In each junction which is adjacent to exactly one track segment there is a balloon loop. In the network there are k balloon loops in total.

Bytetown owns $\lfloor k/2 \rfloor$ trams. Each tram goes along a route which connects two different balloon loops and which does not visit any junction more than once. Routes of different trams may visit same junctions and track segments, however a given balloon loop may be the endpoint of a route of at most one tram.

Each tram consumes daily a quantity of electricity proportional to the length of its route. So the total amount of the subsidy for which you will need to apply depends to a large extent on what routes are chosen for the trams. The mayor would like to know what is the minimum and the maximum electricity cost of a correct set of tram routes. Your task is to compute these two numbers.

Input

The first line of input contains one integer n ($1 \leqslant n \leqslant 100\,000$) denoting the number of junctions in the tram network. The junctions are numbered from 1 to n. The i-th of the following $n-1$ lines contains three integers a_i, b_i, c_i ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$, $1 \leqslant c_i \leqslant 1\,000\,000$) representing the numbers of junctions connected by the i-th track segment and the length of this segment.

Output

Your program should output exactly two lines. The first of these lines should contain the minimum total length of tram routes for a correct set of routes. The second line should contain the maximum length of the tram routes.

Examples

trams.in	trams.out
8 1 3 1 2 3 1 3 4 1 4 5 1 5 6 1 5 7 2 5 8 1	4 9