15-295 Spring 2019 #6 Trees

A. Valera and Elections

1 second, 256 megabytes

The city Valera lives in is going to hold elections to the city Parliament.

The city has n districts and n - 1 bidirectional roads. We know that from any district there is a path along the roads to any other district. Let's enumerate all districts in some way by integers from 1 to n, inclusive. Furthermore, for each road the residents decided if it is the problem road or not. *A problem road* is a road that needs to be repaired.

There are n candidates running the elections. Let's enumerate all candidates in some way by integers from 1 to n, inclusive. If the candidate number i will be elected in the city Parliament, he will perform exactly one promise — to repair all problem roads on the way from the i-th district to the district 1, where the city Parliament is located.

Help Valera and determine the subset of candidates such that if all candidates from the subset will be elected to the city Parliament, all problem roads in the city will be repaired. If there are several such subsets, you should choose the subset consisting of the minimum number of candidates.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the number of districts in the city.

Then n - 1 lines follow. Each line contains the description of a city road as three positive integers x_i, y_i, t_i ($1 \le x_i, y_i \le n$, $1 \le t_i \le 2$) — the districts connected by the i-th bidirectional road and the road type. If t_i equals to one, then the i-th road isn't the problem road; if t_i equals to two, then the i-th road is the problem road.

It's guaranteed that the graph structure of the city is a tree.

Output

In the first line print a single non-negative number k — the minimum size of the required subset of candidates. Then on the second line print k space-separated integers $a_1, a_2, \ldots a_k$ — the numbers of the candidates that form the required subset. If there are multiple solutions, you are allowed to print any of them.

input
5 1 2 2 2 3 2 3 4 2 4 5 2
output
1 5

input
5 1 2 1 2 3 2 2 4 1 4 5 1
output
1 3

Another sample input/output is available on-line.

B. Fools and Roads

2 seconds, 256 megabytes

They say that Berland has exactly two problems, fools and roads. Besides, Berland has n cities, populated by the fools and connected by the roads. All Berland roads are bidirectional. As there are many fools in Berland, between each pair of cities there is a path (or else the fools would get upset). Also, between each pair of cities there is no more than one simple path (or else the fools would get lost).

But that is not the end of Berland's special features. In this country fools sometimes visit each other and thus spoil the roads. The fools aren't very smart, so they always use only the simple paths.

A simple path is the path which goes through every Berland city not more than once.

The Berland government knows the paths which the fools use. Help the government count for each road, how many distinct fools can go on it.

Note how the fools' paths are given in the input.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the number of cities.

Each of the next n - 1 lines contains two space-separated integers u_i, v_i ($1 \le u_i$, $v_i \le n$, $u_i \ne v_i$), that means that there is a road connecting cities u_i and v_i.

The next line contains integer k ($0 \le k \le 10^5$) — the number of pairs of fools who visit each other.

Next k lines contain two space-separated numbers. The i-th line ($i > 0$) contains numbers a_i, b_i ($1 \le a_i$, $b_i \le n$). That means that the fool number $2i$ - 1 lives in city a_i and visits the fool number $2i$, who lives in city b_i. The given pairs describe simple paths, because between every pair of cities there is only one simple path.

Output

Print n - 1 integer. The integers should be separated by spaces. The i-th number should equal the number of fools who can go on the i-th road. The roads are numbered starting from one in the order, in which they occur in the input.

input
5 1 2 1 3 2 4 2 5 2 1 4 3 5
output
2 1 1 1

input
5 3 4 4 5 1 4 2 4 3 2 3 1 3 3 5
output
3 1 1 1

In the first sample the fool number one goes on the first and third road and the fool number 3 goes on the second, first and fourth ones.

In the second sample, the fools number 1, 3 and 5 go on the first road, the fool number 5 will go on the second road, on the third road goes the fool number 3, and on the fourth one goes fool number 1.

C. Distance in Tree

3 seconds, 512 megabytes

A *tree* is a connected graph that doesn't contain any cycles.

The *distance* between two vertices of a tree is the length (in edges) of the shortest path between these vertices.

You are given a tree with n vertices and a positive number k. Find the number of distinct pairs of the vertices which have a distance of exactly k between them. Note that pairs (v, u) and (u, v) are considered to be the same pair.

Input

The first line contains two integers n and k ($1 \le n \le 50000$, $1 \le k \le 500$) — the number of vertices and the required distance between the vertices.

Next $n - 1$ lines describe the edges as "a_i b_i" (without the quotes) ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), where a_i and b_i are the vertices connected by the i-th edge. All given edges are different.

Output

Print a single integer — the number of distinct pairs of the tree's vertices which have a distance of exactly k between them.

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

input
5 2 1 2 2 3 3 4 2 5
output
4

input
5 3 1 2 2 3 3 4 4 5
output
2

In the first sample the pairs of vertexes at distance 2 from each other are (1, 3), (1, 5), (3, 5) and (2, 4).

Problem D
Particle Swapping
Problem ID: particles

The research team of prof. Feynmansson is preparing a new groundbreaking experiment in particle physics. On a special plate they have prepared a system consisting of a number of nodes connected via wires[2]. In the beginning of the experiment a pair of particles appears at two different nodes of the system: one normal particle of matter appears at some node A, and one corresponding particle of antimatter appears at some node B. The goal of the experiment is to swap these particles, i.e., to arrive at a state where the normal particle is at node B and the antiparticle is at node A. This state should be reached by a sequence of *moves*, where each move consists of transmitting one of the particles from its current location to a neighbouring node via a wire.

From flickr under Creative Commons licence, by Tom Fassbender

As you probably remember from popular science TV programmes, playing with matter and antimatter is usually not that safe. In particular, if particles of matter and antimatter get too close to each other, they will annihilate each other blowing up the whole experiment. Therefore, the research team would like to swap the locations of the particles in such a manner that the minimum Euclidean distance between them during the experiment is as large as possible. This minimum distance is called the *safeness* of the experiment. For simplicity, we assume that while a particle is transmitted via a wire we do not consider its location; in other words, the only risky moments during the experiment are when both particles are at some nodes. You may assume that it is always possible to swap the particles with positive safeness, that is, so that the particles are never placed at the same node during swapping.

Another catch is that the physicists do not know precisely where the particles will appear. They have made a list of potential pairs of initial locations (A, B), and for each of them they would like to know the maximum possible safeness of swapping the particles. Help them in this task.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 500$), denoting the number of nodes in the system. Then follow n lines, each containing two integers x, y ($-10\,000 \leq x, y \leq 10\,000$); the numbers in the i-th line denote the coordinates on the plate of the i-th node. No two nodes are located at the same point.

The next line of the input contains a single integer m ($0 \leq m \leq 2\,000$), denoting the number of wires in the system. Then follow m lines; each line contains a description of a wire as a pair of integers a, b ($1 \leq a, b \leq n$, $a \neq b$), denoting the indices of the nodes that are connected by the wire. You may assume that no two nodes are connected by more than one wire, and no wire connects a node with itself.

[2]The wires may cross each other on the plate.

The next line of the input contains a single integer ℓ ($1 \leq \ell \leq \binom{n}{2}$), denoting the length of the list of potential initial positions prepared by the physicists. Then follow ℓ lines, each containing two integers a, b ($1 \leq a, b \leq n$, $a \neq b$), denoting the indices of the initial nodes A and B, respectively.

Output

Output exactly ℓ lines. The i-th line of the output should contain a single floating point number, being the maximum possible safeness for the i-th pair of initial positions listed by the physicists. Absolute or relative errors of value at most 10^{-6} will be tolerated.

Sample Input 1	Sample Output 1
6 0 0 -1 3 -1 0 -1 -3 3 0 0 1 6 1 2 2 3 3 4 4 1 1 5 5 6 5 6 5 2 4 2 6 3 6 4 6	1.00000000 3.16227766 2.23606798 1.41421356 3.16227766

E. Conveyor Belts

3 seconds, 256 megabytes

Automatic Bakery of Cyberland (ABC) recently bought an $n \times m$ rectangle table. To serve the diners, ABC placed seats around the table. The size of each seat is equal to a unit square, so there are $2(n+m)$ seats in total.

ABC placed conveyor belts on each unit square on the table. There are three types of conveyor belts: "^", "<" and ">". A "^" belt can bring things upwards. "<" can bring leftwards and ">" can bring rightwards.

Let's number the rows with 1 to n from top to bottom, the columns with 1 to m from left to right. We consider the seats above and below the top of the table are rows 0 and $n+1$ respectively. Also we define seats to the left of the table and to the right of the table to be column 0 and $m+1$. Due to the conveyor belts direction restriction there are currently no way for a diner sitting in the row $n+1$ to be served.

Given the initial table, there will be q events in order. There are two types of events:

- "A x y" means, a piece of bread will appear at row x and column y (we will denote such position as (x, y)). The bread will follow the conveyor belt, until arriving at a seat of a diner. It is possible that the bread gets stuck in an infinite loop. Your task is to simulate the process, and output the final position of the bread, or determine that there will be an infinite loop.
- "C x y c" means that the type of the conveyor belt at (x, y) is changed to c.

Queries are performed separately meaning that even if the bread got stuck in an infinite loop, it won't affect further queries.

Input

The first line of input contains three integers n, m and q ($1 \le n \le 10^5$, $1 \le m \le 10$, $1 \le q \le 10^5$), separated by a space.

Next n lines, each line contains m characters, describing the table. The characters can only be one of "<^>".

Next q lines, each line describes an event. The format is "`C x y c`" or "`A x y`" (Consecutive elements are separated by a space). It's guaranteed that $1 \le x \le n$, $1 \le y \le m$. c is a character from the set "<^>".

There are at most 10000 queries of "C" type.

Output

For each event of type "A", output two integers tx, ty in a line, separated by a space, denoting the destination of (x, y) is (tx, ty).

If there is an infinite loop, you should output $tx = ty = -1$.

input
2 2 3 >> ^^ A 2 1 C 1 2 < A 2 1
output
1 3 -1 -1

input
4 5 7 ><<^< ^<^^> >>>^> >^>>^ A 3 1 A 2 2 C 1 4 < A 3 1 C 1 2 ^ A 3 1 A 2 2
output
0 4 -1 -1 -1 -1 0 2 0 2

For the first sample:

If the bread goes from $(2, 1)$, it will go out of the table at $(1, 3)$.

After changing the conveyor belt of $(1, 2)$ to "<", when the bread goes from $(2, 1)$ again, it will get stuck at "><", so output is $(-1, -1)$.

F. Tourists

2 seconds, 256 megabytes

There are n cities in Cyberland, numbered from 1 to n, connected by m bidirectional roads. The j-th road connects city a_j and b_j.

For tourists, souvenirs are sold in every city of Cyberland. In particular, city i sell it at a price of w_i.

Now there are q queries for you to handle. There are two types of queries:

- "C a w": The price in city a is changed to w.
- "A a b": Now a tourist will travel from city a to b. He will choose a route, he also doesn't want to visit a city twice. He will buy souvenirs at the city where the souvenirs are the cheapest (possibly exactly at city a or b). You should output the minimum possible price that he can buy the souvenirs during his travel.

More formally, we can define routes as follow:

- A route is a sequence of cities $[x_1, x_2, ..., x_k]$, where k is a certain positive integer.
- For any $1 \le i < j \le k, x_i \ne x_j$.
- For any $1 \le i < k$, there is a road connecting x_i and x_{i+1}.
- The minimum price of the route is $min(w_{x_1}, w_{x_2}, ..., w_{x_k})$.
- The required answer is the minimum value of the minimum prices of all valid routes from a to b.

Input

The first line of input contains three integers n, m, q ($1 \le n, m, q \le 10^5$), separated by a single space.

Next n lines contain integers w_i ($1 \le w_i \le 10^9$).

Next m lines contain pairs of space-separated integers a_j and b_j ($1 \le a_j, b_j \le n, a_j \ne b_j$).

It is guaranteed that there is at most one road connecting the same pair of cities. There is always at least one valid route between any two cities.

Next q lines each describe a query. The format is "`C` a w" or "`A` a b" ($1 \le a, b \le n, 1 \le w \le 10^9$).

Output

For each query of type "`A`", output the corresponding answer.

input
3 3 3 1 2 3 1 2 2 3 1 3 A 2 3 C 1 5 A 2 3
output
1 2

input
7 9 4 1 2 3 4 5 6 7 1 2 2 5 1 5 2 3 3 4 2 4 5 6 6 7 5 7 A 2 3 A 6 4 A 6 7 A 3 3
output
2 1 5 3

For the second sample, an optimal routes are:

From 2 to 3 it is $[2, 3]$.

From 6 to 4 it is $[6, 5, 1, 2, 4]$.

From 6 to 7 it is $[6, 5, 7]$.

From 3 to 3 it is $[3]$.