A. Building Skyscrapers

time limit per test: 3.5 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

We are going to build a new city: the Metropolis. The city is going to be built on an infinite square grid. The finished city will consist of n skyscrapers, each occupying a different cell of the grid. At any moment during the construction, the cells that currently do not contain a skyscraper are called empty.

You are given the planned coordinates of the n skyscrapers. Your task is to find an order in which they can be built while satisfying the rules listed below.

- The building crew has only one crane, so the Metropolis has to be constructed one skyscraper at a time.
- The first skyscraper can be built anywhere on the grid.
- Each subsequent skyscraper has to share a side or a corner with at least one of the previously built skyscrapers (so that it's easier to align the new skyscraper to the grid properly).
- When building a skyscraper, there has to be a way to deliver material to the construction site from the outside of Metropolis by only moving it through empty cells that share a side. In other words, there should be a path of side-adjacent empty cells that connects the cell that will contain the skyscraper to some cell (r, c) with $|r| > 10^9$ and/or $|c| > 10^9$.

If a solution exists, let's denote the numbers of skyscrapers in the order in which they should be built by $s_1, \dots, s_n$. There are two types of subtasks:

Type 1: You may produce any valid order.

Type 2: You must find the order that maximizes s_n. Among those, you must find the one that maximizes s_{n-1}. And so on. In other words, you must find the valid order of building for which the sequence $(s_n, s_{n-1}, \dots, s_1)$ is lexicographically largest.

Input

The first line contains a single integer n ($1 \le n \le 150,000$) – the number of skyscrapers.

The second line contains a single integer t ($1 \le t \le 2$) describing the type of the subtask as defined above.

Then, n lines follow. The i-th of these lines contains two space-separated integers r_i and c_i ($|r_i|, |c_i| \le 10^9$) denoting the coordinates of the cell containing skyscraper i.

(The skyscrapers are not numbered in any particular order. The only reason why they have numbers is that they are used in the output format.)

It is guaranteed that no two skyscrapers coincide.

Output

If it is impossible to build the skyscrapers according to the given rules, print a single line containing the string "NO".

Otherwise, print $n + 1$ lines. The first of these lines should contain the string "YES". For each i, the i-th of the remaining n lines should contain a single integer s_i.

In subtasks with $t = 1$, if there are multiple valid orders, you may output any one of them.

Scoring

Subtask 1 (8 points): $t = 1$ and $n \le 10$

Subtask 2 (14 points): $t = 1$ and $n \le 200$

Subtask 3 (12 points): $t = 1$ and $n \le 2,000$

Subtask 4 (17 points): $t = 2$ and $n \le 2,000$

Subtask 5 (20 points): $t = 1$

Subtask 6 (10 points): $t = 2$, $n \le 70,000$ and $|r_i|, |c_i| \le 900$ for each i

Subtask 7 (19 points): $t = 2$

Examples

input

```
3
2
```

```
0 0
0 1
0 2
```

output

```
YES
1
2
3
```

input

```
3
1
0 0
1 1
2 2
```

output

```
YES
2
3
1
```

input

```
2
1
0 0
0 2
```

output

```
NO
```

Note

In the first example, there are three skyscrapers in a row. All of them can always be reached from outside the Metropolis, and there are four build orders which preserve connectivity:

- 1, 2, 3
- 2, 1, 3
- 2, 3, 1
- 3, 2, 1

Since $t = 2$, we must choose the first option.

In the second example, the only difference from the first example is that skyscraper 2 shares only corners with skyscrapers 1 and 3, the same set of orders as in the first sample is valid. Since $t = 1$, each of these answers is correct.

In the third example, the Metropolis is disconnected. We obviously can't build that.

B. Dynamic Diameter

time limit per test: 6 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

You are given a weighted undirected tree on n vertices and a list of q updates. Each update changes the weight of one edge. The task is to output the diameter of the tree after each update.

(The distance between two vertices is the sum of the weights on the unique simple path that connects them. The diameter is the largest of all those distances.)

Input

The first line contains three space-separated integers n, q and w ($2 \leq n \leq 100,000, 1 \leq q \leq 100,000,$ $1 \leq w \leq 20,000,000,000,000$) – the number of vertices in the tree, the number of updates and the limit on the weights of edges. The vertices are numbered 1 through n.

Next, $n-1$ lines describing the initial tree follow. The i-th of these lines contains three space-separated integers a_i, b_i, c_i ($1 \leq a_i, b_i \leq n, 0 \leq c_i < w$) meaning that initially, there is an edge between vertices a_i and b_i with weight c_i. It is guaranteed that these $n-1$ lines describe a tree.

Finally, q lines describing queries follow. The j-th of these lines contains two space-separated integers d_j, e_j ($0 \leq d_j < n-1, 0 \leq e_j < w$). These two integers are then transformed according to the following scheme:

- $d'_j = (d_j + last) \bmod (n-1)$
- $e'_j = (e_j + last) \bmod w$

where $last$ is the result of the last query (initially $last = 0$). Tuple (d'_j, e'_j) represents a query which takes the $d'_j + 1$-th edge from the input and sets its weight to e'_j.

Output

Output q lines. For each i, line i should contain the diameter of the tree after the i-th update.

Scoring

Subtask 1 (11 points): $n, q \leq 100$ and $w \leq 10,000$

Subtask 2 (13 points): $n, q \leq 5,000$ and $w \leq 10,000$

Subtask 3 (7 points): $w \leq 10,000$ and the edges of the tree are exactly all valid edges of the form $\{1, i\}$ (Hence, the tree is a star centered at vertex 1.)

Subtask 4 (18 points): $w \leq 10,000$, and the edges of the tree are exactly all valid edges of the forms $\{i, 2i\}$ and $\{i, 2i+1\}$ (Hence, if we were to root the tree at vertex 1, it would be a balanced binary tree.)

Subtask 5 (24 points): it is guaranteed that after each update a longest simple path goes through vertex 1

Subtask 6 (27 points): no additional constraints

Examples

input

```
4 3 2000
1 2 100
2 3 1000
2 4 1000
2 1030
1 1020
1 890
```

output

```
2030
2080
2050
```

input

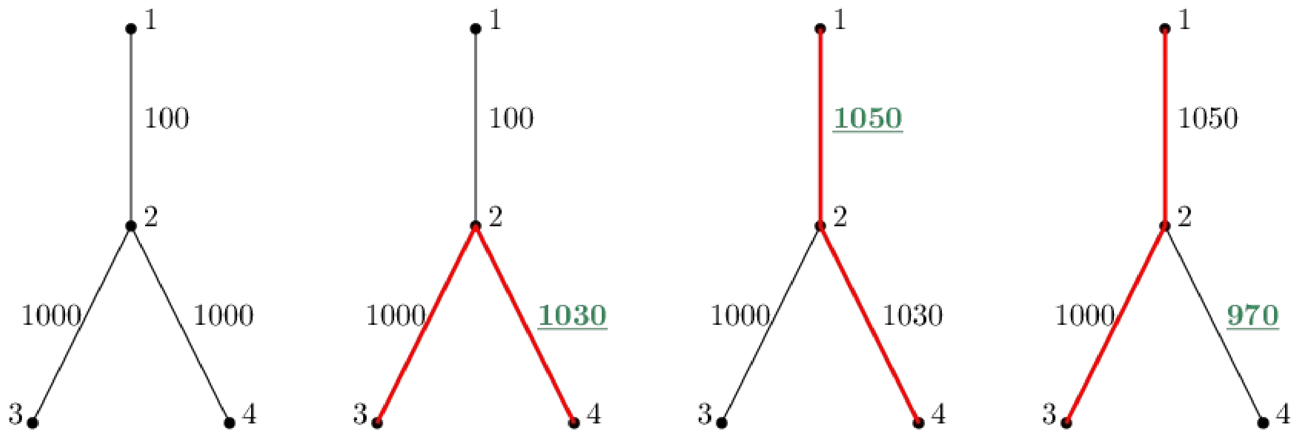
```
10 10 10000
1 9 1241
5 6 1630
10 5 1630
2 6 853
10 1 511
5 3 760
8 3 1076
4 10 1483
7 10 40
8 2051
5 6294
5 4168
7 1861
0 5244
6 5156
3 3001
8 5267
5 3102
8 3623
```

output

```
6164
7812
8385
6737
6738
7205
6641
7062
6581
5155
```

Note

The first sample is depicted in the figure below. The left-most picture shows the initial state of the graph. Each following picture depicts the situation after an update. The weight of the updated edge is painted green, and the diameter is red.



The first query changes the weight of the 3rd edge, i.e. $\{2,4\}$, to 1030. The largest distance between any pair of vertices is 2030 – the distance between 3 and 4.

As the answer is 2030, the second query is

$$d'_2 = (1 + 2030) \bmod 3 = 0$$

$$e'_2 = (1020 + 2030) \bmod 2000 = 1050$$

Hence the weight of the edge $\{1,2\}$ is changed to 1050. This causes the pair $\{1,4\}$ to be the pair with the greatest distance, namely 2080.

The third query is decoded as

$$d'_3 = (1 + 2080) \bmod 3 = 2$$

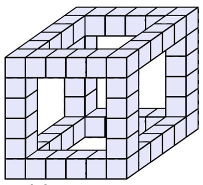
$$e'_3 = (890 + 2080) \bmod 2000 = 970$$

As the weight of the edge $\{2,4\}$ decreases to 970, the most distant pair is suddenly $\{1,3\}$ with 2050.

C. Cubeword

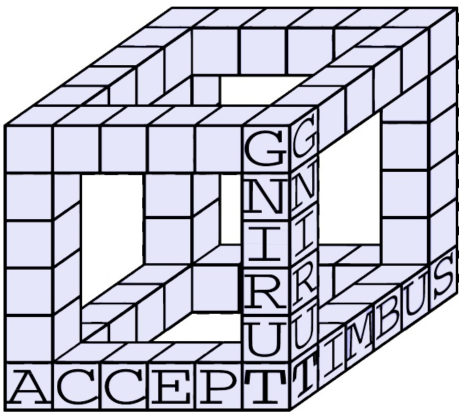
time limit per test: 1.7 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

A **cubeword** is a special type of a crossword. When building a cubeword, you start by choosing a positive integer a: the side length of the cube. Then, you build a big cube consisting of $a \times a \times a$ unit cubes. This big cube has 12 edges. Then, you discard all unit cubes that do not touch the edges of the big cube. The figure below shows the object you will get for $a = 6$.



Finally, you assign a letter to each of the unit cubes in the object. You must get a meaningful word along each edge of the big cube. Each edge can be read in either direction, and it is sufficient if one of the two directions of reading gives a meaningful word.

The figure below shows the object for $a = 6$ in which some unit cubes already have assigned letters. You can already read the words '`SUBMIT`', '`ACCEPT`' and '`TURING`' along three edges of the big cube.



You are given a list of valid words. Each word from the wordlist may appear on arbitrarily many edges of a valid cubeword. Find and report the number of different cubewords that can be constructed, modulo $998,244,353$.

If one cubeword can be obtained from another by rotation or mirroring, they are considered **distinct**.

Input

The first line contains a single integer n ($1 \le n \le 100,000$) – the number of words.

Then, n lines follow. Each of these lines contains one word that can appear on the edges of the big cube. The length of each word is between 3 and 10, inclusive.

It is guaranteed that all words are different.

Output

Output a single integer, the number of distinct cubewords for the given list of valid words modulo $998,244,353$.

Scoring

Subtask 1 (21 points): the words consist only of letters 'a' - 'f' (lowercase)

Subtask 2 (29 points): the words consist only of letters 'a' - 'p' (lowercase)

Subtask 3 (34 points): the words consist of letters 'a' - 'p' (lowercase) and 'A' - 'P' (uppercase)

Subtask 4 (16 points): the words consist of letters 'a' - 'z' (lowercase), 'A' - 'Z' (uppercase) and digits '0' - '9'

Examples

input

```
1
radar
```

output

```
1
```

input

```
1
robot
```

output

```
2
```

input

```
2
FLOW
WOLF
```

output

```
2
```

input

```
2
baobab
bob
```

output

```
4097
```

input

```
3
TURING
SUBMIT
ACCEPT
```

output

```
162
```

input

```
3
MAN1LA
MAN6OS
AN4NAS
```

output

```
114
```

Note

In the first sample, the only possibility is for the word "`radar`" to be on each edge of the cube.

In the second sample, there are two cubes, which are just rotations of each other – the word "`robot`" is on every edge, and the difference between the two cubes is whether the lower left front corner contains '`r`' or '`t`'.

The third sample is similar to the second one. The fact that we can read the word on each edge in both directions does not affect the answer.

In the fourth sample, there is one cube with the word "bob" on each edge. There are also $2^{12} = 4096$ cubes with the word "baobab" on each edge. (For each of the 12 edges, we have two possible directions in which the word "baobab" can appear.)
