CodeForces Problems

For Future Reference

Jerry Sun

December 11, 2022

§1 2000-2500

D2. Tree Queries (Hard Version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between this problem and D1 is the bound on the size of the tree.

You are given an unrooted tree with n vertices. There is some hidden vertex x in that tree that you are trying to find.

To do this, you may ask k queries $v_1, v_2, \ldots, v_k$ where the v_i are vertices in the tree. After you are finished asking all of the queries, you are given k numbers $d_1, d_2, \ldots, d_k$, where d_i is the number of edges on the shortest path between v_i and x. Note that you know which distance corresponds to which query.

What is the minimum k such that there exists some queries $v_1, v_2, \ldots, v_k$ that let you always uniquely identify x (no matter what x is).

Note that you don't actually need to output these queries.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

Each of the next $n - 1$ lines contains two integers x and y ($1 \le x, y \le n$), meaning there is an edges between vertices x and y in the tree.

It is guaranteed that the given edges form a tree.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case print a single nonnegative integer, the minimum number of queries you need, on its own line.

Example

PROBLEMS SUBMIT STATUS STANDINGS CUSTOM TEST

E. Doremy's Number Line

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Doremy has two arrays a and b of n integers each, and an integer k.

Initially, she has a number line where no integers are colored. She chooses a permutation p of $[1, 2, \ldots, n]$ then performs n moves. On the i-th move she does the following:

- Pick an **uncolored** integer x on the number line such that either:
  - $x \leq a_{p_i}$; or
  - there exists a **colored** integer y such that $y \leq a_{p_i}$ and $x \leq y + b_{p_i}$.
- Color integer x with color p_i.

Determine if the integer k can be colored with color 1.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers n and k ($1 \leq n \leq 10^5$, $1 \leq k \leq 10^9$).

Each of the following n lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output "YES" (without quotes) if the point k can be colored with color 1. Otherwise, output "NO" (without quotes).

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes" and "Yes" will be recognized as a positive response).

E. Doremy's Number Line

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Doremy has two arrays a and b of n integers each, and an integer k.

Initially, she has a number line where no integers are colored. She chooses a permutation p of $[1, 2, \ldots, n]$ then performs n moves. On the i-th move she does the following:

- Pick an **uncolored** integer x on the number line such that either:
  - $x \leq a_{p_i}$; or
  - there exists a **colored** integer y such that $y \leq a_{p_i}$ and $x \leq y + b_{p_i}$.
- Color integer x with color p_i.

Determine if the integer k can be colored with color 1.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line contains two integers n and k ($1 \leq n \leq 10^5$, $1 \leq k \leq 10^9$).

Each of the following n lines contains two integers a_i and b_i ($1 \leq a_i, b_i \leq 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

F. Doremy's Experimental Tree

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Doremy has an edge-weighted tree with n vertices whose weights are **integers** between 1 and 10^9. She does $\frac{n(n+1)}{2}$ experiments on it.

In each experiment, Doremy chooses vertices i and j such that $j \leq i$ and connects them directly with an edge with weight 1. Then, there is exactly one cycle (or self-loop when $i = j$) in the graph. Doremy defines $f(i, j)$ as the sum of lengths of shortest paths from every vertex to the cycle.

Formally, let $\mathrm{dis}_{i,j}(x, y)$ be the length of the shortest path between vertex x and y when the edge (i, j) of weight 1 is added, and $S_{i,j}$ be the set of vertices that are on the cycle when edge (i, j) is added. Then,

$$f(i, j) = \sum_{x=1}^{n} \left(\min_{y \in S_{i,j}} \mathrm{dis}_{i,j}(x, y) \right).$$

Doremy writes down all values of $f(i, j)$ such that $1 \leq j \leq i \leq n$, then goes to sleep. However, after waking up, she finds that the tree has gone missing. Fortunately, the values of $f(i, j)$ are still in her notebook, and she knows which i and j they belong to. Given the values of $f(i, j)$, can you help her restore the tree?

It is guaranteed that at least one suitable tree exists.

Input

The first line of input contains a single integer n ($2 \leq n \leq 2000$) — the number of vertices in the tree.

The following n lines contain a lower-triangular matrix with i integers on the i-th line; the j-th integer on the i-th line is $f(i, j)$ ($0 \leq f(i, j) \leq 2 \cdot 10^{15}$).

It is guaranteed that there exists a tree whose weights are integers between 1 and 10^9 such that the values of $f(i, j)$ of the tree match those given in the input.

Output

Print $n - 1$ lines describing the tree. In the i-th line of the output, output three integers u_i, v_i, w_i ($1 \leq u_i, v_i \leq n$, $1 \leq w_i \leq 10^9$), representing an edge (u_i, v_i) whose weight is w_i.

If there are multiple answers, you may output any.

All edges must form a tree and all values of $f(i, j)$ must match those given in the input.

E. Make It Connected

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a simple undirected graph consisting of n vertices. The graph doesn't contain self-loops, there is at most one edge between each pair of vertices. Your task is simple: make the graph connected.

You can do the following operation any number of times (possibly zero):

- Choose a vertex u arbitrarily.
- For each vertex v satisfying $v \neq u$ in the graph individually, if v is adjacent to u, remove the edge between u and v, otherwise add an edge between u and v.

Find the minimum number of operations required to make the graph connected. Also, find any sequence of operations with the minimum length that makes the graph connected.

Input

Each test contains multiple test cases. The first line contains a single integer t ($1 \leq t \leq 800$) — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer n ($2 \leq n \leq 4000$) — the number of vertices in the graph.

Then n lines follow. The i-th row contains a binary string s_i of length n, where $s_{i,j}$ is '1' if there is an edge between vertex i and j initially, otherwise $s_{i,j}$ is '0'.

It is guaranteed that $s_{i,i}$ is always '0' and $s_{i,j} = s_{j,i}$ for $1 \leq i, j \leq n$.

It is guaranteed that the sum of n over all test cases does not exceed 4000.

Output

For each test case, in the first line, output an integer m — the minimum number of operations required.

If m is greater than zero, then print an extra line consisting of m integers — the vertices chosen in the operations in your solution. If there are multiple solutions with the minimum number of operations, print any.

E. Tick, Tock

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Tannhaus, the clockmaker in the town of Winden, makes mysterious clocks that measure time in h hours numbered from 0 to $h-1$. One day, he decided to make a puzzle with these clocks.

The puzzle consists of an $n \times m$ grid of clocks, and each clock always displays some hour exactly (that is, it doesn't lie between two hours). In one move, he can choose any row or column and shift all clocks in that row or column one hour forward[†].

The grid of clocks is called *solvable* if it is possible to make all the clocks display the same time.

While building his puzzle, Tannhaus suddenly got worried that it might not be possible to make the grid solvable. Some cells of the grid have clocks already displaying a certain initial time, while the rest of the cells are empty.

Given the partially completed grid of clocks, find the number of ways[‡] to assign clocks in the empty cells so that the grid is solvable. The answer can be enormous, so compute it modulo 10^9+7.

[†] If a clock currently displays hour t and is shifted one hour forward, then the clock will instead display hour $(t+1) \bmod h$.

[‡] Two assignments are different if there exists some cell with a clock that displays a different time in both arrangements.

Input

The first line of input contains t ($1 \le t \le 5 \cdot 10^4$) — the number of test cases.

The first line of each test case consists of 3 integers n, m, and h ($1 \le n, m \le 2 \cdot 10^5$; $1 \le h \le 10^9$) — the number of rows in the grid, the number of columns in the grid, and the number of the hours in the day respectively.

The next n lines each contain m integers, describing the clock grid. The integer x ($-1 \le x < h$) in the i-th row and the j-th column represents the initial hour of the corresponding clock, or if $x = -1$, an empty cell.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the number of ways to assign clocks in the empty cells so that the grid is solvable. The answer can be huge, so output it modulo 10^9+7.