

Task: BAJ

Bytering

Maratona de Verão, Day 3. Source file baj.* Available memory: 128 MB.

22.01.2014

Bytering consists of n towns numbered from 0 to $n-1$. The towns numbered $1, 2, \dots, n-1$ are arranged in this order in a circle, with pairs of adjacent towns connected by bidirectional roads. Town no. 0 is the capital of Bytering, it lies in the center of the circle and is connected with all the remaining towns. The time of travelling each road in Bytering is a positive integer number of seconds.

The authorities of Bytering would like to improve the communication network. They have decided to build two airports, in two most distant towns of Bytering (the distance is measured as the shortest travel time).

Input

The first line of input contains one integer n ($3 \leq n \leq 500\,000$), the number of towns in Bytering. The second line contains $n-1$ positive integers that show the travel time between the adjacent towns on the circle; the i -th number represents the travel time between town no. i and the next town on the circle. The third line contains $n-1$ positive integers that show the travel time between the capital and the towns on the circle; the i -th number represents the travel time between the capital and the town no. i . The sum of all travel times in the input does not exceed 10^9 .

Output

The only line of output should contain one integer: the travel time between two most distant towns in Bytering.

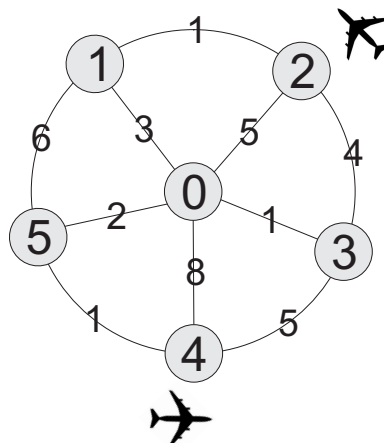
Example

For the input data:

```
6
1 4 5 1 6
3 5 1 8 2
```

the correct result is:

```
7
```



Explanation of the example: Two furthest towns in Bytering are (2, 4) with travel time 7. These are the towns where the two airports should be built.

Task: KAM

Stone Game

Maratona de Verão, Day 3. Source file kam.* Available memory: 128 MB.

22.01.2014

Bitie and Bytie are spending their vacation at the Bytic Sea. The boys really enjoy intellectual riddles, even when they are at the beach.

This time they gathered a number of round stones that were washed ashore and started playing a new game. The rules of the game are fairly simple. In the first move Bitie can take any number of stones that is greater than zero but smaller than the total number of stones in the heap. Since then the boys alternate moves, starting from Bytie. In each move a player can take any non-zero number of stones from the heap (possibly even all the stones from the heap) provided that this number of stones was not taken in *any* previous move. In other words, in each move a different number of stones must be taken. The player who cannot make a move loses.

Given the number of stones in the initial heap, your task is to check if Bitie can win the game assuming that both boys play optimally.

Input

The first line of input contains one integer t ($1 \leq t \leq 1\,000\,000$), the number of test cases.

Each of the following t lines contains one integer n ($1 \leq n \leq 1\,000\,000\,000$), the number of stones at the beginning of the game.

Output

Your program should output t lines with answers to the respective test cases. Each line should contain the word TAK (Polish for *yes*) or NIE (Polish for *no*) depending on whether Bitie can win the game.

Example

For the input data:

```
1
3
```

the correct result is:

```
NIE
```

Task: LOG

Logarithm

Maratona de Verão, Day 3. Source file log.* Available memory: 64 MB.

22.01.2014

Write a program that computes $\log_a b$.

Input

The first line of input contains an integer a ($2 \leq a \leq 40$). The second line contains an integer b ($1 \leq b \leq 10^{10^5}$).

Output

Output a single line containing $\log_a b$. You can assume that the result is an integer.

Example

For the input data:

2
16

the correct result is:

4

Task: PLA

Planar Graph

Maratona de Verão, Day 3. Source file pla.* Available memory: 512 MB.

22.01.2014

You are given a two-vertex-connected* planar[‡] graph[¶] G . In this graph at most two faces** are surrounded by an odd number of edges. You are also given a planar embedding of the graph G . You should verify whether there exists a partition of the edges of G into a number of simple cycles^{††} of even length.

Input

The first line of input contains two integers n and m ($2 \leq n \leq 1\,000\,000$, $1 \leq m \leq 5\,000\,000$), denoting the number of vertices and edges of the graph G . Vertices are numbered 1 to n , whereas edges are numbered 1 to m . Each edge connects two different vertices. A pair of vertices may be connected by several edges.

The following n lines describe the edges of the graph. The i -th of those lines contains a description of edges incident to the vertex i ; the description starts with an integer s_i ($1 \leq s_i \leq m$), followed by a list of s_i integers between 1 and m . Each of those numbers represents an edge adjacent to the vertex i . Moreover the list of s_i edges is sorted clockwise with respect to the vertex i .

Output

If a desired partition of the edges does not exist, then your program should output NIE (Polish for *no*).

Otherwise the first line of output should contain TAK (Polish for *yes*). The following lines should contain a valid partition of the edges of G into simple cycles. Each of those lines should start with an even integer j ($2 \leq j \leq n$), followed by j indices of edges belonging to the described simple cycle. Every two consecutive edges should have a common endpoint. Each edge of G should appear in the output exactly once.

Example

For the input data:

```
10 16
2 1 8
2 8 7
4 1 9 2 14
4 6 13 7 14
4 16 10 9 15
4 16 15 13 12
4 2 10 3 11
4 5 12 6 11
2 3 4
2 4 5
```

the correct result is:

```
TAK
6 16 10 3 4 5 12
4 6 11 2 14
6 8 1 9 15 13 7
```

*A graph $G = (V, E)$ is two-vertex-connected if for any $v \in V$, the graph $(V \setminus \{v\}, E)$ is connected[†].

[†]A graph $G = (V, E)$ is connected, if for each partition into non-empty subsets $V_1, V_2 \subseteq V, V_1 \cap V_2 = \emptyset, V_1 \cup V_2 = V$ there exists an edge $uv \in E$, such that $u \in V_1$ and $v \in V_2$.

[‡]A graph $G = (V, E)$ is planar, if there exists a planar embedding[§] of G into the plane.

[§]A planar embedding of a planar graph into the plane is such its drawing, where each vertex is assigned a point, whereas each edge is assigned a curve connecting the points assigned to the endpoints of the edge. Two curves corresponding to edges may intersect only in their endpoints.

[¶]A graph is a pair (V, E) , where E is a multiset^{||} of two-element subsets of V .

^{||}A multiset is a set in which an element may appear several times; formally, a multiset is a function from any set into the set of natural numbers \mathbf{N} .

**Consider a planar embedding of a planar graph into the plane. Each of the regions bounded by the curves corresponding to edges is called a face. Note that in each planar graph there exists an infinite face "surrounding" the graph.

^{††}A set of edges $C \subseteq E$ is a simple cycle, if the edges of C form a connected graph in which each vertex is adjacent to exactly two edges.

Task: SWI

Glowworms

Maratona de Verão, Day 3. Source file swi.* Available memory: 128 MB.

22.01.2014

Early morning Byteasar woke up and went fishing in the Bytic Lake. At some moment he spotted a group of glowworms that were flying above the surface of the lake. Byteasar really enjoyed this view and decided to take a photo of the glowworms.

Photos made by Byteasar's camera have the shape of a square. Before a photo is taken, Byteasar decides on the vertical and horizontal alignment of the photo and zooms in or out. He never rotates the camera, otherwise the photo would turn out crooked.

Byteasar wants all the glowworms to be on the photo. He would like to adjust the parameters of the photo, in particular the camera zoom, so that the glowworms appear as big as possible. He is even willing to wait for a while until the glowworms form the best configuration for the photo to be taken.

We simplify things a bit and assume that all the glowworms are flying in a plane that is parallel to the camera sensor and that each glowworm has a constant velocity vector.

Input

The first line of input contains one integer n ($1 \leq n \leq 100\,000$), the number of glowworms. Each of the following n lines contains four integers x_i, y_i, a_i, b_i ($-10^6 \leq x_i, y_i, a_i, b_i \leq 10^6$) that represent the initial position (x_i, y_i) and the velocity vector $[a_i, b_i]$ of the i -th glowworm. In other words, after t units of time the i -th glowworm has coordinates $(x_i + t \cdot a_i, y_i + t \cdot b_i)$. All points are specified in a rectangular coordinate system with axes parallel to the sides of the camera sensor.

Output

Your program should output one line with a non-negative real number d — the minimal side length of a square that can be used to cover all the glowworms at some moment of time (the sides of the square must be parallel to the axes of the coordinate system). Your result may differ from the exact result by at most 10^{-3} .

Example

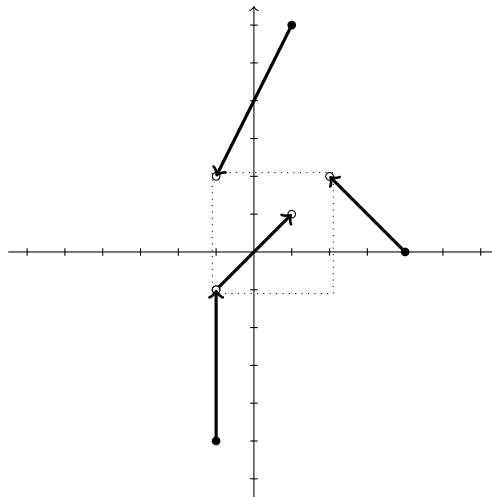
For the input data:

```
4
4 0 -1 1
1 6 -1 -2
-1 -5 0 2
-1 -1 1 1
```

the correct result is:

```
3.00000000000000000000
```

Explanation of the example: The figure shows the initial positions of the glowworms and their positions after 2 units of time. A 3×3 square that covers all the glowworms at the time $t = 2$ is also shown.



Task: ZAR

Turn Off the Light

Maratona de Verão, Day 3. Source file zar.* Available memory: 64 MB.

22.01.2014

It may seem that turning the lights off is a simple task, but not this time. We have a very complex lighting installation to deal with. The room is full of lights, which are currently on or off. There are also some switches, each of which is connected to two distinct lights. When a switch is pressed, the state of both lights connected to the switch changes (from on to off, and from off to on). Your task is to determine, whether all lights can be switched off by using the switches in the room.

Input

The first line of input contains an integer t ($1 \leq t \leq 100$) that denotes the number of test cases that follow.

The first line of each test case contains two integers n and m ($1 \leq n \leq 1000$, $1 \leq m \leq 100\,000$) that specify the number of lights and the number of switches in the room. The lights are numbered 1 to n . The next line describes the lights. It contains a sequence of n integers c_i ($c_i \in \{0, 1\}$). If $c_i = 0$, the i -th light is turned off. Otherwise, it is initially turned on.

The following m lines describe the switches. Each of them contains two integers a_i, b_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$). They mean that pressing i -th switch changes the state of lights a_i and b_i .

Output

Output the answers to consecutive test cases. If it is not possible to turn the lights off, the answer to the test case is a single line with a word NIE (Polish for *no*). Otherwise, output a line with a word TAK (Polish for *yes*) followed by an integer k ($0 \leq k \leq 500\,000$). After that, output k more lines containing the numbers of switches that have to be pressed in order to turn the lights off. The switches are numbered 1 to m in the order they are given in the input. If there are multiple correct answers, your program may output any of them.

Example

For the input data:

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

the correct result is:

```
NIE
TAK 2
1
2
```