GLOBUSSOFT

TECHNOLOGY AHEAD OF TIME

Greetings From Globussoft

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net, PHP

- To solve these 3 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any **Search Engine** like Google, Yahoo, Bing …

All the best for your test

Globussoft

QUESTION - 1

Two players take part in the game **polygons**. A convex polygon with *n* vertices divided by *n-3* diagonals into *n-2* triangles is necessary. These diagonals may intersect in vertices of the polygon only. One of the triangles is black and the remaining ones are white. Players proceed in alternate turns. Each player, when its turn comes, cuts away one triangle from the polygon. players are allowed to cut off triangles along the given diagonals. The winner is the player who cuts away the black triangle.
NOTE: We call a polygon **convex** if a segment joining any two points of the polygon is contained in the polygon.

Task

Write a program which:

- reads from the standard input the description of the polygon,
- verifies whether the player who starts the game has a winning strategy,
- writes the result to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. The first line of each test case contains an integer *n*, *4 <= n <= 50000*. This is the number of vertices in the polygon. The vertices of the polygon are numbered, clockwise, from *0* to *n-1*.
The next *n-2* lines comprise descriptions of triangles in the polygon. In the *i+1*-th line, *1 <= i <= n-2*, there are three non-negative integers *a, b, c* separated by single spaces. Theses are numbers of vertices of the *i*-th triangle. The first triangle in a sequence is black.

Output

The output for each test case should have one line with the word:

- YES, if the player, who starts the game has a winning strategy,
- NO, if he does not have a winning strategy.

Example

Sample input:

```
1
6
0 1 2
2 4 3
4 2 0
0 5 4
```

Sample output:

```
YES
```

QUESTION – 2

In the time of Louis XIII and his powerful minister cardinal Richelieu in the Full Barrel Inn *n* musketeers had consumed their meal and were drinking wine. Wine had not run short and therefore the musketeers were eager to quarrel, a drunken brawl broke out, in which each musketeer insulted all the others.

A duel was inevitable. But who should fight who and in what order? They decided (for the first time since the brawl they had done something together) that they would stay in a circle and draw lots in order. A drawn musketeer fought against his neighbor to the right. A looser "quit the game" and to be more precise his corpse was taken away by servants. The next musketeer who stood beside the looser became the neighbor of a winner.

After years, when historians read memories of the winner they realized that a final result depended in a crucial extent on the order of duels. They noticed that a fence practice had indicated, who against who could win a duel. It appeared that (in mathematical language) the relation "**A** wins **B**" was not transitive! It could happen that the musketeer **A** fought better than **B**, **B** better than **C** and **C** better than **A**. Of course, among three of them the first duel influenced the final result. If **A** and **B** fight as the first, **C** wins eventually. But if **B** and **C** fight as the first, **A** wins finally. Historians fascinated by their discovery decided to verify which musketeers could survive. The fate of France and the whole civilized Europe indeed depended on that!

Task

N persons with consecutive numbers from *1* to *n* stay in a circle. They fight *n-1* duels. In the first round one of these persons (e.g. with the number *i*) fights against its neighbor to the right, i.e. against the person numbered *i+1* (or, if *i=n*, against the person numbered *1*). A looser quits the game, and the circle is tighten so that the next person in order becomes a winner's neighbor. We are given the table with possible duels results, in the form of a matrix. If **A***i,j* = 1 then the person with the number *i* always wins with the person *j*. If **A***i,j* = 0 the person *i* looses with *j*. We can say that the person *k* may win the game if there exists such a series of *n-1* drawings, that *k* wins the final duel.
Write a program which:

- reads matrix **A** from the standard input,
- computes numbers of persons, who may win the game,
- writes them into the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case integer *n* which satisfies the inequality *3<=n<=100* is written. In each of the following *n* lines appears one word consisting of *n* digits 0 or 1. A digit on *j*-th position in *i*-th line denote **A***i,j*. Of course **A***i,j* = 1 - **A***j,i*, for *i*<>*j*. We assume that **A***i,i* = 1, for each *i*.

Output

For each test case in the first line there should be written *m* - the number of persons, who may win the game. In the following *m* lines numbers of these persons should be written in ascending order, one number in each line.

Example

Sample input:

```
1
7
1111101
0101100
0111111
0001101
0000101
1101111
0100001
```

Sample output:

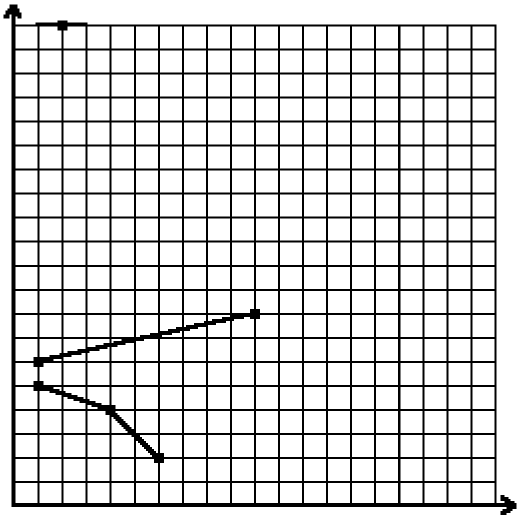
```
3
1
3
6
```

The order of duels: 1-2, 1-3, 5-6, 7-1, 4-6, 6-1 gives a final victory to the person numbered 6. You can also check that only two persons more (1 and 3) may win the game.

QUESTION – 3

We have a cartesian coordinate system drawn on a sheet of paper. Let us consider broken lines that can be drawn with a single pencil stroke from the left to the right side of the sheet. We also require that for each segment of the line the angle between the straight line containing this segment and the OX axis belongs to [-45°, 45°] range. A broken line fulfilling above conditions is called a flat broken line. Suppose we are given *n* distinct points with integer coordinates. What is the minimal number of flat broken lines that should be drawn in order to cover all the points (a point is covered by a line if it belongs to this line)?

Example



For 6 points whose coordinates are (1,6), (10,8), (1,5), (2,20), (4,4), (6,2) the minimal number of flat broken lines covering them is 3.

Task

Write a program that for each test case:

- reads the number of points and their coordinates;
- computes the minimal number of flat broken lines that should be drawn to cover all the points;
- outputs the result.

Input

The number of test cases *t* is in the first line of input, then *t* test cases follow separated by an empty line.

In the first line of a test case there is one positive integer *n*, not greater than 30000, which denotes the number of points. In the following *n* lines there are coordinates of points. Each line contains two integers *x*, *y* separated by a single space, 0 <= *x* <= 30000, 0 <= *y* <= 30000. The numbers in the *i*-th line are the coordinates of the *i*-th point.

Output

For each test case you should output one line with the minimal number of flat broken lines that should be drawn to cover all the points.

Example

Sample input:

```
1
```

```
6
1 6
10 8
1 5
2 20
4 4
6 2
```

Sample output:

```
3
```

QUESTION – 4

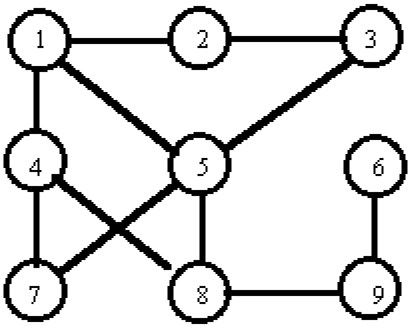
Chase is a two-person board game. A board consists of squares numbered from 1 to *n*. For each pair of different squares it is known if they are adjacent to one another or they are not. Each player has a piece at his disposal. At the beginning of a game pieces of players are placed on fixed, distinct squares. In one turn a player can leave his piece on the square it stands or move it to an adjacent square.

A game board has the following properties:

- it contains no triangles, i.e. there are no three distinct squares such that each pair of them is adjacent,
- each square can be reached by each player.

A game consists of many turns. In one turn each player makes a single move. Each turn is started by player A. We say that player A is caught by player B if both pieces stand on the same square. Decide, if for a given initial positions of pieces, player B can catch player A, independently of the moves of his opponent. If so, how many turns player B needs to catch player A if both play optimally (i.e. player A tries to run away as long as he can and player B tries to catch him as quickly as possible).

Example



Consider the board in the figure. Adjacent squares (denoted by circles) are connected by edges. If at the beginning of a game pieces of players A and B stand on the squares 9 and 4 respectively, then player B can catch player A in the third turn (if both players move optimally). If game starts with pieces on the squares 8 (player A) and 4 (player B) then player B cannot catch player A (if A plays correctly).

Task

Write a program that for each test case:

- reads the description of a board and numbers of squares on which pieces are placed initially.
- decides if player B can catch player A and if so, computes how many turns he needs (we assume that both players play optimally);
- outputs the result.

Input

The number of test cases *t* is in the first line of input, then *t* test cases follow separated by an empty line.

In the first line of a test case there are four integers *n*, *m*, *a* and *b* separated by single spaces, where 2 <= *n* <= 3000, *n*-1 <= *m* <= 15000, 1 <= *a*, *b* <= *n*. These are (respectively): the number of squares of the board, the number of adjacent (unordered) pairs, the number of the square on which the piece of player A is placed, the number of the square on which the piece of player B is placed. In each of the following lines there are two distinct positive integers separated by a single space, which denote numbers of adjacent squares.

Output

For each test case you should output one line containing:

- one word "`No`", if player B cannot catch player A, or
- one integer - the number of turns needed by B to catch A (if B can catch A).

Example

Sample input:

```
1
9 11 9 4
1 2
3 2
1 4
4 7
7 5
5 1
6 9
8 5
9 8
5 3
```

```
4 8
```

Sample output:

```
3
```

QUESTION – 5

Every member of Byteland Credit Society is entitled to loan any amount of Bytelandish ducats unless it is 10^{30} or more, but he has to return the whole amount within seven days. There are 100 ATMs in the Client Service Room of the Society. They are numbered from 0 to 99. Every ATM can perform one action only: it can pay or receive a fixed amount. The i-th ATM pays 2^i ducats if i is even or it receives 2^i ducats if i is odd. If a client is going to loan a fixed sum of money it is necessary to check if he is able to get the money using every ATM at most once. If so, numbers of ATMs he has to use should be determined. It is also necessary to check if the client can return the money in a similar way, and if so, to determine numbers of ATMs he has to use.

Example

A client who is going to loan 7 ducats gets 16 ducats from the ATM # 4 and 1 ducat from the ATM # 0 and then he returns 8 ducats in the ATM # 3 and 2 ducats in the ATM # 1. In order to return the amount of 7 ducats he receives 1 ducat from the ATM # 0 and then he returns 8 ducats in ATM # 3.

Task

Write a program that:

- reads the number of clients *n*, for every client reads from the same file the amount of money he is going to loan;
- checks for every client if he is able to get the money using every ATM at most once and if so, determines the numbers of ATMs he has to use;
- outputs the results.

Input

In the first line of input there is one positive integer n <= 10000, which equals the number of clients.

In each of the following n lines there is one positive integer less than 10^{30} (at most 30 decimal digits). The number in the i-th line describes the amount of ducats which the client i is going to loan.

Output

For each client you should output two lines with a decreasing sequence of positive integers from the range [0..99] separated by single spaces, or one word "`No`":

In the first line of the *i*-th pair of lines there should be numbers of ATMs (in decreasing order) that the client *i* should use to get his loan or one word "`No`" if the loan cannot be received according to the rules;

In the second line of the *i*-th pair there should be numbers of ATMs (in decreasing order) which the client *i* should use to return his loan or the word "`No`".

Example

Sample input:

```
2
7
633825300114114700748351602698
```

Sample output:

```
4 3 1 0
3 0
No
99 3 1
```