Helvetic Coding Contest 2016 online mirror (teams, unrated)

A1. Collective Mindsets (easy)

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

Tonight is brain dinner night and all zombies will gather together to scarf down some delicious brains. The artful Heidi plans to crash the party, incognito, disguised as one of them. Her objective is to get away with at least one brain, so she can analyze the zombies' mindset back home and gain a strategic advantage.

They will be N guests tonight: $N - 1$ real zombies and a fake one, our Heidi. The living-dead love hierarchies as much as they love brains: each one has a unique rank in the range 1 to $N - 1$, and Heidi, who still appears slightly different from the others, is attributed the highest rank, N. Tonight there will be a chest with brains on display and every attendee sees how many there are. These will then be split among the attendees according to the following procedure:

The zombie of the highest rank makes a suggestion on who gets how many brains (every brain is an indivisible entity). A vote follows. If at least half of the attendees accept the offer, the brains are shared in the suggested way and the feast begins. But if majority is not reached, then the highest-ranked zombie is killed, and the next zombie in hierarchy has to make a suggestion. If he is killed too, then the third highest-ranked makes one, etc. (It's enough to have exactly half of the votes – in case of a tie, the vote of the highest-ranked alive zombie counts twice, and he will of course vote in favor of his own suggestion in order to stay alive.)

You should know that zombies are very greedy and sly, and they know this too – basically all zombie brains are alike. Consequently, a zombie will never accept an offer which is suboptimal for him. That is, if an offer is not *strictly* better than a potential later offer, he will vote against it. And make no mistake: while zombies may normally seem rather dull, tonight their intellects are perfect. Each zombie's priorities for tonight are, in descending order:

1. survive the event (they experienced death already once and know it is no fun),
2. get as many brains as possible.

Heidi goes first and must make an offer which at least half of the attendees will accept, and which allocates at least one brain for Heidi herself.

What is the smallest number of brains that have to be in the chest for this to be possible?

Input

The only line of input contains one integer: N, the number of attendees ($1 \leq N \leq 10^9$).

Output

Output one integer: the smallest number of brains in the chest which allows Heidi to take one brain home.

Examples

input
```
1
```
output
```
1
```

input
```
4
```
output
```
2
```

Note

A2. Collective Mindsets (medium)

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

Way to go! Heidi now knows how many brains there must be for her to get one. But throwing herself in the midst of a clutch of hungry zombies is quite a risky endeavor. Hence Heidi wonders: what is the smallest number of brains that must be in the chest for her to get out at all (possibly empty-handed, but alive)?

The brain dinner night will evolve just as in the previous subtask: the same crowd is present, the $N - 1$ zombies have the exact same mindset as before and Heidi is to make the first proposal, which must be accepted by at least half of the attendees for her to survive.

Input

The only line of input contains one integer: N, the number of attendees ($1 \leq N \leq 10^9$).

Output

Output one integer: the smallest number of brains in the chest which allows Heidi to merely survive.

Examples

input
```
1
```
output
```
0
```

input
```
3
```
output
```
1
```

input
```
99
```
output
```
49
```

A3. Collective Mindsets (hard)

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Heidi got one brain, thumbs up! But the evening isn't over yet and one more challenge awaits our dauntless agent: after dinner, at precisely midnight, the N attendees love to play a very risky game...

Every zombie gets a number n_i $(1 \le n_i \le N)$ written on his forehead. Although no zombie can see his own number, he can see the numbers written on the foreheads of all $N - 1$ fellows. Note that not all numbers have to be unique (they can even all be the same). From this point on, no more communication between zombies is allowed. Observation is the only key to success. When the cuckoo clock strikes midnight, all attendees have to simultaneously guess the number on their own forehead. If at least one of them guesses his number correctly, all zombies survive and go home happily. On the other hand, if not a single attendee manages to guess his number correctly, all of them are doomed to die!

Zombies aren't very bright creatures though, and Heidi has to act fast if she does not want to jeopardize her life. She has one single option: by performing some quick surgery on the brain she managed to get from the chest, she has the ability to remotely reprogram the decision-making strategy of all attendees for their upcoming midnight game! Can you suggest a sound strategy to Heidi which, given the rules of the game, ensures that at least one attendee will guess his own number correctly, for any possible sequence of numbers on the foreheads?

Given a zombie's rank R and the $N - 1$ numbers n_i on the other attendees' foreheads, your program will have to return the number that the zombie of rank R shall guess. Those answers define your strategy, and we will check if it is flawless or not.

Input

The first line of input contains a single integer T $(1 \le T \le 50000)$: the number of scenarios for which you have to make a guess.

The T scenarios follow, described on two lines each:

- The first line holds two integers, N $(2 \le N \le 6)$, the number of attendees, and R $(1 \le R \le N)$, the rank of the zombie who has to make the guess.
- The second line lists $N - 1$ integers: the numbers on the foreheads of all other attendees, listed in increasing order of the attendees' rank. (Every zombie knows the rank of every other zombie.)

Output

For every scenario, output a single integer: the number that the zombie of rank R shall guess, based on the numbers n_i on his $N - 1$ fellows' foreheads.

Examples

input

```
4
2 1
1
2 2
1
2 1
2
2 2
2
```

output

```
1
2
2
1
```

input

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

output

```
5
2
```

Note

For instance, if there were $N = 2$ two attendees, a successful strategy could be:

- The zombie of rank 1 always guesses the number he sees on the forehead of the zombie of rank 2.
- The zombie of rank 2 always guesses the opposite of the number he sees on the forehead of the zombie of rank 1.

B1. Recover Polygon (easy)

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

The zombies are gathering in their secret lair! Heidi will strike hard to destroy them once and for all. But there is a little problem... Before she can strike, she needs to know where the lair is. And the intel she has is not very good.

Heidi knows that the lair can be represented as a rectangle on a lattice, with sides parallel to the axes. Each vertex of the polygon occupies an integer point on the lattice. For each cell of the lattice, Heidi can check the level of Zombie Contamination. This level is an integer between 0 and 4, equal to the number of corners of the cell that are inside or on the border of the rectangle.

As a test, Heidi wants to check that her Zombie Contamination level checker works. Given the output of the checker, Heidi wants to know whether it could have been produced by a single non-zero area *rectangular-shaped* lair (with axis-parallel sides).

Input

The first line of each test case contains one integer N, the size of the lattice grid ($5 \leq N \leq 50$). The next N lines each contain N characters, describing the level of Zombie Contamination of each cell in the lattice. Every character of every line is a digit between `0` and `4`.

Cells are given in the same order as they are shown in the picture above: rows go in the decreasing value of y coordinate, and in one row cells go in the order of increasing x coordinate. This means that the first row corresponds to cells with coordinates $(1, N)$, ..., (N, N) and the last row corresponds to cells with coordinates $(1, 1)$, ..., $(N, 1)$.

Output

The first line of the output should contain `Yes` if there exists a single non-zero area rectangular lair with corners on the grid for which checking the levels of Zombie Contamination gives the results given in the input, and `No` otherwise.

Example

input

```
6
000000
000000
012100
024200
012100
000000
```

output

```
Yes
```

Note

The lair, if it exists, has to be rectangular (that is, have corners at some grid points with coordinates (x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)), has a non-zero area and be contained inside of the grid (that is, $0 \leq x_1 < x_2 \leq N$, $0 \leq y_1 < y_2 \leq N$), and result in the levels of Zombie Contamination as reported in the input.

B2. Recover Polygon (medium)

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Now that Heidi has made sure her Zombie Contamination level checker works, it's time to strike! This time, the zombie lair is a strictly convex polygon on the lattice. Each vertex of the polygon occupies a point on the lattice. For each cell of the lattice, Heidi knows the level of Zombie Contamination – the number of corners of the cell that are inside or on the border of the lair.

Given this information, Heidi wants to know the exact shape of the lair to rain destruction on the zombies. Help her!

Input

The input contains multiple test cases.

The first line of each test case contains one integer N, the size of the lattice grid ($5 \le N \le 500$). The next N lines each contain N characters, describing the level of Zombie Contamination of each cell in the lattice. Every character of every line is a digit between 0 and 4.

Cells are given in the same order as they are shown in the picture above: rows go in the decreasing value of y coordinate, and in one row cells go in the order of increasing x coordinate. This means that the first row corresponds to cells with coordinates $(1, N)$, ..., (N, N) and the last row corresponds to cells with coordinates $(1, 1)$, ..., $(N, 1)$.

The last line of the file contains a zero. This line should not be treated as a test case. The sum of the N values for all tests in one file will not exceed 5000.

Output

For each test case, give the following output:

The first line of the output should contain one integer V, the number of vertices of the polygon that is the secret lair. The next V lines each should contain two integers, denoting the vertices of the polygon in the clockwise order, starting from the lexicographically smallest vertex.

Examples

input

```
8
00000000
00000110
00012210
01234200
02444200
01223200
00001100
00000000
5
00000
01210
02420
01210
00000
7
0000000
0122100
0134200
0013200
0002200
0001100
0000000
0
```

output

```
4
2 3
2 4
6 6
5 2
4
2 2
2 3
3 3
3 2
3
2 5
4 5
4 2
```

Note

It is guaranteed that the solution always exists and is unique. It is guaranteed that in the correct solution the coordinates of the polygon vertices are between 2 and $N - 2$. A vertex (x_1, y_1) is lexicographically smaller than vertex (x_2, y_2) if $x_1 < x_2$ or .

B3. Recover Polygon (hard)

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Zombies have found out about the Zombie Contamination level checker and managed to damage it! Now detecting the shape of their main compound will be a real challenge for Heidi. As before, a lair can be represented as a strictly convex polygon on a lattice. Each vertex of the polygon occupies a point on the lattice. However, the damaged Zombie Contamination level checker can only tell, for each cell, whether the level of Zombie Contamination for that cell is in the set $\{1, 2, 3\}$. In other words, Heidi knows all the cells of the lattice for which the Contamination level is not 0 and not 4.

Given this information, Heidi still wants to know the exact shape of the lair to rain destruction on the zombies. Help her!

Input

The input contains multiple test cases.

The first line of each test case contains two space-separated integers N and M, where N is the size of the lattice grid ($5 \leq N \leq 100000$) and M is the number of lattice points for which the Zombie Contamination level is 1, 2, or 3 ($8 \leq M \leq 200000$).

The second line of each test case contains M pairs of integers $x_1, y_1, ..., x_M, y_M$ – coordinates of the cells with Zombie Contamination level not equal to 0 nor 4. It is guaranteed that $1 \leq x_i, y_i \leq N$. All pairs x_i, y_i are different.

Cells are enumerated based on the coordinates of their upper right corner. This means that the bottommost leftmost cell that touches the origin has coordinates $(1, 1)$, and the uppermost leftmost cell is identified as $(1, N)$.

The last line of the file contains two zeroes. This line should not be treated as a test case. The sum of the M values for all tests in one file will not exceed 200000.

Output

For each test case, the following output is expected:

The first line of the output should contain one integer V, the number of vertices of the polygon that is the secret lair. The next V lines each should contain two integers, denoting the vertices of the polygon in the clockwise order, starting from the lexicographically smallest vertex.

Example

input

```
8 19
2 3 2 4 2 5 3 3 3 5 4 3 4 5 4 6 5 2 5 3 5 6 6 2 6 3 6 4 6 5 6 6 6 7 7 6 7 7
5 8
2 2 2 3 2 4 3 2 3 4 4 2 4 3 4 4
0 0
```

output

```
4
2 3
2 4
6 6
5 2
4
2 2
2 3
3 3
3 2
```

Note

It is guaranteed that the solution always exists and is unique. It is guaranteed that in the correct solution the coordinates of the polygon vertices are between 1 and $N - 1$. A vertex (x_1, y_1) is lexicographically smaller than vertex (x_2, y_2) if $x_1 < x_2$ or .

C1. Brain Network (easy)

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

One particularly well-known fact about zombies is that they move and think terribly slowly. While we still don't know why their movements are so sluggish, the problem of laggy thinking has been recently resolved. It turns out that the reason is not (as previously suspected) any kind of brain defect – it's the opposite! Independent researchers confirmed that the nervous system of a zombie is highly complicated – it consists of n brains (much like a cow has several stomachs). They are interconnected by *brain connectors*, which are veins capable of transmitting thoughts between brains. There are two important properties such a brain network should have to function properly:

1. It should be possible to exchange thoughts between any two pairs of brains (perhaps indirectly, through other brains).
2. There should be no redundant brain connectors, that is, removing any brain connector would make property 1 false.

If both properties are satisfied, we say that the nervous system is *valid*. Unfortunately (?), if the system is not valid, the zombie stops thinking and becomes (even more) dead. Your task is to analyze a given nervous system of a zombie and find out whether it is valid.

Input

The first line of the input contains two space-separated integers n and m ($1 \le n, m \le 1000$) denoting the number of brains (which are conveniently numbered from 1 to n) and the number of brain connectors in the nervous system, respectively. In the next m lines, descriptions of brain connectors follow. Every connector is given as a pair of brains a b it connects ($1 \le a, b \le n$, $a \ne b$).

Output

The output consists of one line, containing either `yes` or `no` depending on whether the nervous system is valid.

Examples

input

```
4 4
1 2
2 3
3 1
4 1
```

output

```
no
```

input

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

output

```
yes
```

C2. Brain Network (medium)

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

Further research on zombie thought processes yielded interesting results. As we know from the previous problem, the nervous system of a zombie consists of n brains and m brain connectors joining some pairs of brains together. It was observed that the intellectual abilities of a zombie depend mainly on the topology of its nervous system. More precisely, we define the distance between two brains u and v ($1 \le u, v \le n$) as the minimum number of brain connectors used when transmitting a thought between these two brains. The brain latency of a zombie is defined to be the maximum distance between any two of its brains. Researchers conjecture that the brain latency is the crucial parameter which determines how smart a given zombie is. Help them test this conjecture by writing a program to compute brain latencies of nervous systems.

In this problem you may assume that any nervous system given in the input is valid, i.e., it satisfies conditions (1) and (2) from the easy version.

Input

The first line of the input contains two space-separated integers n and m ($1 \le n, m \le 100000$) denoting the number of brains (which are conveniently numbered from 1 to n) and the number of brain connectors in the nervous system, respectively. In the next m lines, descriptions of brain connectors follow. Every connector is given as a pair of brains a b it connects ($1 \le a, b \le n$ and $a \ne b$).

Output

Print one number – the brain latency.

Examples

input

```
4 3
1 2
1 3
1 4
```

output

```
2
```

input

```
5 4
1 2
2 3
3 4
3 5
```

output

```
3
```

C3. Brain Network (hard)

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

Breaking news from zombie neurology! It turns out that – contrary to previous beliefs – every zombie is *born* with a single brain, and only later it evolves into a complicated brain structure. In fact, whenever a zombie consumes a brain, a new brain appears in its nervous system and gets immediately connected to one of the already existing brains using a single brain connector. Researchers are now interested in monitoring the brain latency of a zombie. Your task is to write a program which, given a history of evolution of a zombie's nervous system, computes its brain latency at every stage.

Input

The first line of the input contains one number n – the number of brains in the final nervous system ($2 \le n \le 200000$). In the second line a history of zombie's nervous system evolution is given. For convenience, we number all the brains by $1, 2, ..., n$ in the same order as they appear in the nervous system (the zombie is born with a single brain, number 1, and subsequently brains $2, 3, ..., n$ are added). The second line contains $n - 1$ space-separated numbers $p_2, p_3, ..., p_n$, meaning that after a new brain k is added to the system, it gets connected to a parent-brain .

Output

Output $n - 1$ space-separated numbers – the brain latencies after the brain number k is added, for $k = 2, 3, ..., n$.

Example

input

```
6
1
2
2
1
5
```

output

```
1 2 2 3 4
```

D1. The Wall (easy)

time limit per test: 0.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

"The zombies are lurking outside. Waiting. Moaning. And when they come..."

"When they come?"

"I hope the Wall is high enough."

Zombie attacks have hit the Wall, our line of defense in the North. Its protection is failing, and cracks are showing. In places, gaps have appeared, splitting the wall into multiple segments. We call on *you* for help. Go forth and explore the wall! Report how many disconnected segments there are.

The wall is a two-dimensional structure made of bricks. Each brick is one unit wide and one unit high. Bricks are stacked on top of each other to form columns that are up to R bricks high. Each brick is placed either on the ground or directly on top of another brick. Consecutive non-empty columns form a *wall segment*. The entire wall, all the segments and empty columns in-between, is C columns wide.

Input

The first line of the input consists of two space-separated integers R and C, $1 \le R, C \le 100$. The next R lines provide a description of the columns as follows:

- each of the R lines contains a string of length C,
- the c-th character of line r is `B` if there is a brick in column c and row $R - r + 1$, and `.` otherwise.

The input will contain at least one character `B` and it will be valid.

Output

The number of wall segments in the input configuration.

Examples

input

```
3 7
.......
.......
.BB.B..
```

output

```
2
```

input

```
4 5
..B..
..B..
B.B.B
BBB.B
```

output

```
2
```

input

```
4 6
..B...
B.B.BB
BBB.BB
BBBBBB
```

output

```
1
```

input

```
1 1
B
```

output

```
1
```

input

```
10 7
.......
.......
.......
.......
.......
.......
.......
```

```
.......
...B...
B.BB.B.
```

output

```
3
```

input

```
8 8
........
........
........
........
.B......
.B.....B
.B.....B
.BB...BB
```

output

```
2
```

Note

In the first sample case, the 2nd and 3rd columns define the first wall segment, and the 5th column defines the second.

D2. The Wall (medium)

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

Heidi the Cow is aghast: cracks in the northern Wall? Zombies gathering outside, forming groups, preparing their assault? This must not happen! Quickly, she fetches her HC^2 (Handbook of Crazy Constructions) and looks for the right chapter:

How to build a wall:

1. *Take a set of bricks.*
2. *Select one of the possible wall designs. Computing the number of possible designs is left as an exercise to the reader.*
3. *Place bricks on top of each other, according to the chosen design.*

This seems easy enough. But Heidi is a Coding Cow, not a Constructing Cow. Her mind keeps coming back to point 2b. Despite the imminent danger of a zombie onslaught, she wonders just how many possible walls she could build with up to n bricks.

A *wall* is a set of wall segments as defined in the easy version. How many different walls can be constructed such that the wall consists of at least 1 and at most n bricks? Two walls are different if there exist a column c and a row r such that one wall has a brick in this spot, and the other does not.

Along with n, you will be given C, the width of the wall (as defined in the easy version). Return the number of different walls modulo $10^6 + 3$.

Input
The first line contains two space-separated integers n and C, $1 \le n \le 500000$, $1 \le C \le 200000$.

Output
Print the number of different walls that Heidi could build, modulo $10^6 + 3$.

Examples

input
```
5 1
```
output
```
5
```

input
```
2 2
```
output
```
5
```

input
```
3 2
```
output
```
9
```

input
```
11 5
```
output
```
4367
```

input
```
37 63
```
output
```
230574
```

Note
The number $10^6 + 3$ is prime.

In the second sample case, the five walls are:

```
                B        B
B., .B, BB, B., and .B
```
In the third sample case, the nine walls are the five as in the second sample case and in addition the following four:

```
B     B
B     B  B         B
B., .B, BB, and BB
```

D3. The Wall (hard)

time limit per test: 15 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

So many wall designs to choose from! Even modulo $10^6 + 3$, it's an enormous number. Given that recently Heidi acquired an unlimited supply of bricks, her choices are endless! She really needs to do something to narrow them down.

Heidi is quick to come up with criteria for a *useful* wall:

- In a useful wall, at least one segment is wider than W bricks. This should give the zombies something to hit their heads against. Or,
- in a useful wall, at least one column is higher than H bricks. This provides a lookout from which zombies can be spotted at a distance.

This should rule out a fair amount of possibilities, right? Help Heidi compute the number of *useless* walls that do not confirm to either of these criteria. In other words, a wall is useless if every segment has width at most W and height at most H.

Parameter C, the total width of the wall, has the same meaning as in the easy version. However, note that the number of bricks is now unlimited.

Output the number of useless walls modulo $10^6 + 3$.

Input

The first and the only line of the input contains three space-separated integers C, W and H ($1 \le C \le 10^8$, $1 \le W, H \le 100$).

Output

Output the number of different walls, modulo $10^6 + 3$, which are useless according to Heidi's criteria.

Examples

input
```
1 1 1
```
output
```
2
```

input
```
1 2 2
```
output
```
3
```

input
```
1 2 3
```
output
```
4
```

input
```
3 2 2
```
output
```
19
```

input
```
5 4 9
```
output
```
40951
```

input
```
40 37 65
```
output
```
933869
```

Note

If there is no brick in any of the columns, the structure is considered as a useless wall.

E1. Photographs (I)

time limit per test: 15 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Human-Cow Confederation (HC^2), led by Heidi, has built a base where people and cows can hide, guarded from zombie attacks. The entrance to the base is protected by an automated gate which performs a kind of a Turing test: it shows the entering creature a photograph and asks them whether the top and bottom halves of this photograph have been swapped or not. A person (or even a cow) will have no problem answering such questions; on the other hand, a zombie would just randomly smash one of the two buttons.

The creature is asked a series of such questions. If at least 75% of them are answered correctly, the gate is unlocked; otherwise, a side door opens, beneath which a huge fan is spinning...

Heidi is now building a robot army to fight the zombies, and she wants the robots to also be able to enter the base. You are tasked with programming them to distinguish the images.

The first two images from the test set. The first picture has been rearranged, but not the second.

Input

The first line of the input contains the number q of questions ($1 \le q \le 220$). After that, q questions follow, each of which in the format described below.

The first line of every question contains two space-separated integers h and w ($1 \le h, w \le 600$) – the height (number of rows) and width (number of columns) of the photograph. (Most photographs are roughly 200×300.) After this, h lines follow, each describing a single row of the picture. The picture is monochrome (in shades of grey). Its i-th row is described by w space-separated integers a_{ij} ($j = 1, ..., w$), where a_{ij} is the brightness of the corresponding pixel ($0 \le a_{ij} < 256$, where 0 is black and 255 is white).

Each picture will be either a real-life photograph, or a real-life photograph which has been broken up into two pieces and rearranged. More precisely, in the latter case, the topmost rows have been moved to the bottom of the picture. It is guaranteed that h is even.

There is only a single input file to be processed, called `all.in`, and it is downloadable from the online judge. You are also a given another input file, called `sample.in`, which contains the first 20 pictures from `all.in`; you are provided the correct answers for `sample.in` in `sample.out`. You are also given a directory `easy_bmp`, which contains the first 50 input photographs in the form of `.bmp` image files, as well as a directory `easy_sample_original_bmp`, which contains the first 20 images *before rearrangement*. Check the notes for the download links.

Output

Your program should print q lines. The i-th line should contain your answer for the i-th question: `YES` if the photograph has been rearranged and `NO` otherwise. Your answers will be accepted if they all conform to this format and if at least 75% of them are correct.

Because the input is rather huge, feel free to process it locally and submit just your precomputed answers (i.e., a program which just prints your output for the input file `all.in`).

Note

The link to download all necessary files is http://assets.codeforces.com/files/690/easy_contestant_package.zip

E2. Photographs (II)

time limit per test: 15 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Zombies seem to have become much more intelligent lately – a few have somehow wandered into the base through the automatic gate. Heidi has had to beef up security, and a new gate has been installed. Unfortunately, now the questions being asked are more complicated, and even humans have trouble answering them. Can you still program the robot army to do this reliably?

The new questions are of the following form: a grayscale photograph has been divided into several horizontal pieces, which have been arbitrarily rearranged. The task is to assemble the original image back from these pieces (somewhat like in a jigsaw puzzle). To further delay the zombies, significant Gaussian-distributed noise has been added to the image.

Input

The input format is the same as in the previous version, except that the first line of every question now contains three space-separated numbers h, w and k ($1 \le h, w \le 600$, $2 \le k \le 16$) – the height (number of rows) and width (number of columns) of the photograph and the number of pieces, respectively. The number of pieces evenly divides the height, and each piece is of the same height h / k.

Again, there is only one input file to be processed, and the same resources are provided to you as in the previous version (except that now you are given *all* input images in `.bmp` format, rather than the first 50).

Output

Your program should print q lines. The i-th line should contain your answer for the i-th question: a space-separated sequence of k numbers $\pi_1, \pi_2, ..., \pi_k$ such that:

- π is a permutation of $\{1, 2, ..., k\}$, that is, each number from 1 to k appears exactly once in π,
- for each $j = 1, ..., k$, π_j is the position (index), in the original image, of the piece which is at position j in the input image. (See the illustration below for clarity.)

The second image from the test set. If the three pieces in the original image are numbered $1, 2, 3$ from top to bottom, then the numbering in the image on the right should be $2, 3, 1$. The correct answer for this image is thus `2 3 1`.

Again, your answers will be accepted if they conform to this format and if at least 75% of them are correct.

Again, you may process the input locally and submit just your precomputed answers (i.e., a program which just prints your output for the input file `all.in`).

Note

The link to download all the necessary materials is http://assets.codeforces.com/files/690/medium_contestant_package.zip

F1. Tree of Life (easy)

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

Heidi has finally found the mythical Tree of Life – a legendary combinatorial structure which is said to contain a prophecy crucially needed to defeat the undead armies.

On the surface, the Tree of Life is just a regular undirected tree well-known from computer science. This means that it is a collection of n points (called vertices), some of which are connected using $n - 1$ line segments (edges) so that each pair of vertices is connected by a *path* (a sequence of one or more edges).

To decipher the prophecy, Heidi needs to perform a number of steps. The first is counting the number of *lifelines* in the tree – these are paths of length 2, i.e., consisting of two edges. Help her!

Input

The first line of the input contains a single integer n – the number of vertices in the tree ($1 \leq n \leq 10000$). The vertices are labeled with the numbers from 1 to n. Then $n - 1$ lines follow, each describing one edge using two space-separated numbers a b – the labels of the vertices connected by the edge ($1 \leq a < b \leq n$). It is guaranteed that the input represents a tree.

Output

Print one integer – the number of lifelines in the tree.

Examples

input

```
4
1 2
1 3
1 4
```

output

```
3
```

input

```
5
1 2
2 3
3 4
3 5
```

output

```
4
```

Note

In the second sample, there are four lifelines: paths between vertices 1 and 3, 2 and 4, 2 and 5, and 4 and 5.

F2. Tree of Life (medium)

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Heidi got tired of deciphering the prophecy hidden in the Tree of Life and decided to go back to her headquarters, rest a little and try there. Of course, she cannot uproot the Tree and take it with her, so she made a drawing of the Tree on a piece of paper. On second thought, she made more identical drawings so as to have n in total (where n is the number of vertices of the Tree of Life) – who knows what might happen?

Indeed, on her way back Heidi was ambushed by a group of zombies. While she managed to fend them off, they have damaged her drawings in a peculiar way: from the i-th copy, the vertex numbered i was removed, along with all adjacent edges. In each picture, the zombies have also erased all the vertex numbers and relabeled the remaining n - 1 vertices arbitrarily using numbers 1 to n (fortunately, each vertex still has a distinct number). *What's more, the drawings have been arbitrarily shuffled/reordered.*

Now Heidi wants to recover the Tree of Life from her descriptions of all the drawings (as lists of edges).

Input

The first line of the input contains $Z \leq 20$ – the number of test cases. Z descriptions of single test cases follow.

In each test case, the first line of input contains numbers n ($2 \leq n \leq 100$) and k (where k is the number of drawings; we have $k = n$). In the following lines, the descriptions of the k drawings are given. The description of the i-th drawing is a line containing m_i – the number of edges in this drawing, followed by m_i lines describing edges, each of which contains two space-separated integers –- the numbers of the two vertices connected by the edge.

Output

If Heidi's drawings cannot possibly come from a single tree, you should output the word NO. Otherwise, output one line containing the word YES and n - 1 lines describing any tree that Heidi's drawings could have come from. For every edge you should output the numbers of the vertices that it connects, separated with a single space. If there are many solutions, print any of them.

Example

input

```
1
5 5
2
4 1
2 1
1
3 1
3
4 1
4 3
2 1
3
3 1
3 2
4 1
3
2 1
3 2
4 2
```

output

```
YES
2 5
4 2
3 2
5 1
```

F3. Tree of Life (hard)

time limit per test: 8 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

To add insult to injury, the zombies have taken all but two drawings from Heidi! Please help her recover the Tree of Life from only these two drawings.

Input

The input format is the same as in the medium version, except that now the bound on n is $2 \le n \le 1000$ and that $k = 2$.

Output

The same as in the medium version.

Example

input

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

output

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