Problem A. APT Upgrade

Source file name:	APT.c, APT.cpp, APT.java, APT.py
Input:	`Standard`
Output:	`Standard`

You are using your favourite program, the BAPC ArchLinux Package Configurator, to upgrade your system. There are n outdated packages that will be upgraded, and your package manager is kind enough to inform you of the download size for each package up front. Due to recent advances in parallelism, it downloads up to k packages in parallel, although you do not know the order in which they are downloaded.

```
:: Retrieving packages...
 electron24-24.8.... 59 MB [####################] 100%
 electron25-25.6.... 60 MB [####################] 100%
 chromium-116.0.5... 91 MB [####################] 100%
 signal-desktop-6... 92 MB [#################-----]  85%
 linux-6.4.11.arc... 92 MB [###############------]  73%
 qt5-webengine-5.... 31 MB [#############--------]  64%
 telegram-desktop... 30 MB [##################---]  95%
 Total (  3/143)    457 MB [#############--------]  62%
```

Output of pacman -Su: only 3 of 143 packages have finished downloading, with 4 more in progress, but already 62% of the total download size is done!

You are now looking at the download progress bar in the console, and observe that only m packages have currently finished downloading but the overall download progress is already very high. This does not seem to make sense! You wonder: what is the maximum overall percentage of the total download size that could be done with this many package downloads completed? Note that there is a small duration of time in which a package that is being downloaded is reported as 100% done, but not yet finished.

Input

The input consists of:

- One line with three integers n, m, and k ($1 \leq n \leq 10^5$, $0 \leq m \leq n$, $1 \leq k \leq 10$), the number of packages being upgraded, the number of packages that finished downloading, and the number of packages that can be downloaded in parallel.
- One line with n integers s ($1 \leq s \leq 10^9$), the sizes of the packages being upgraded.

Output

Output the maximum possible percentage of the download that is done.

Your answer should have an *absolute* error of at most 10^{-4}.

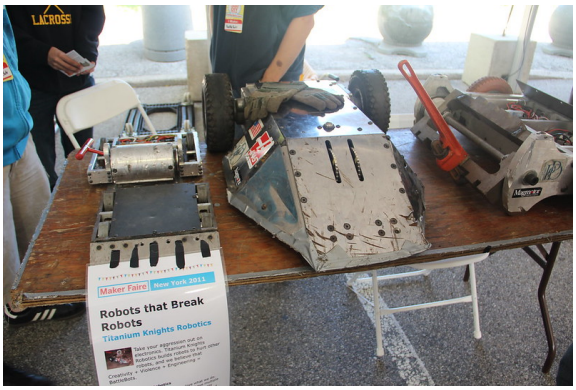
Example

Input	Output
5 1 2 10 25 30 15 20	75
5 0 4 4 2 7 1 3	94.117647059

Problem B. Battle Bots

Source file name:	Battle.c, Battle.cpp, Battle.java, Battle.py
Input:	`Standard`
Output:	`Standard`

You are participating in the Battle-bots Aggressive Power Contest. It is a tournament where each team builds a robot that can battle with other robots, and you win by destroying your opponent's robot. Specifically, you win when your opponent's robot stops moving after its only motor is destroyed.



Battle Bots. CC BY-SA 2.0 by Jeremy Blum on Flickr

You have outfitted your bot with two weapons: it has a sword that can slash the opponent's bot in half, and it has a claw that can take a chunk out of your opponent's bot and crush it into scrap. The attacks take equally long. The program that controls your bot is always running to decide which attack to use next.

If your battle-bot uses the sword attack to cut its opponent in half, the half with the motor will keep moving, and you can ignore the other half. If your battle-bot uses the claw attack, it will take a chunk of size 1 out of the opponent's bot, but unless you can take the bot out entirely you have to assume that the motor of the bot is in the piece you have not clawed, and keep fighting.

For example, consider the second sample case. If your opponent's bot is so big it would take 5 claw attacks to completely crush it, you could act as follows. Start with a sword slash, cutting it down into two pieces of size $2\frac{1}{2}$. Then use your claw on the part that is still moving, so it goes down to size $1\frac{1}{2}$. Cut that piece in half with your sword again to bring it down to size $\frac{3}{4}$. Then finally use your claw to crush the last moving piece of the bot. That way, you can beat it in four attacks.

Your bot is equipped with a quantum computer that can easily simulate a googol attack patterns per microsecond. However, if it does not know what the fastest attack pattern is, it will never know it has reached that, and never stop searching.

Finish your battle bot by writing a program that, given how many claw attacks it would take to take out the opponent, determines the minimal number of attacks you need in the worst case to take it out.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^{18}$), the number of claw attacks it would take to take out your opponent's bot.

Output

Output the least number of attacks needed to destroy your opponent's bot.

Example

Input	Output
5	4
6	4
3	3

Problem C. Compressing Commands

Source file name:	Compressing.c, Compressing.cpp, Compressing.java, Compressing.py
Input:	`Standard`
Output:	`Standard`

Only true terminal-dwellers use green on black colours. CC BY-NC 2.0 by Kjetil Korslien on Flickr

Unlike your friends, you live in a terminal. You are cool. Your terminal is everything to you: you have optimized the font, colour scheme, keyboard shortcuts, and what not.

Once thing still annoys you though: all these commands you're typing involve so many file paths and are so long! Then it occurs to you: all this time you have been working from the *root directory* of the file system, but in fact you can *change directory* to anywhere you like! This should simplify your life a lot!

This way, if your *working directory* is `/a/b`, you can refer to the absolute file path `/a/b/x` using simply `x`. To *go up* a level, you can use `..`, so that you can refer to `/a/y/z` as `../y/z`, and to `/some/other/directory` as `../../some/other/directory`.

You being you, of course you overdo this and will now use relative paths *everywhere*!

Given the n absolute file paths in the command you want to run, find the working directory that minimizes the total number of relative path components. For example, `a/a/b/c`, and `../../a/b` both contain 4 path components. Note that you can only change the working directory to a directory and not to a file path. Filenames will never coincide with directory names in the same directory.

In the first sample it is best to set the working directory to `/home/jury/compressingcommands`, leading to 6 components: `secret`, `solutions`, and `../../hackerman/answers`.

In the second sample it is best to set the working directory to `/a`, leading to 19 path components: `b/a/a/b`, `a/a`, `../b`, `a/b`, `b/a/a/a`, `../c`, and `b/a/b`.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the number of absolute file paths.
- n lines, each with a string s, an absolute file path. Each path contains only lowercase English letters (`a-z`) and slashes ('`/`'), starts with '`/`', does not end with '`/`', and does not contain consecutive slashes ("`//`").

The total number of characters in the n strings is at most 10^6.

Output

Output the minimal number of relative path components that can be achieved by changing to a different working directory.

Example

Input	Output
3 /home/jury/compressingcommands/secret /home/jury/compressingcommands/solutions /home/hackerman/answers	6
7 /a/b/a/a/b /a/a/a /b /a/a/b /a/b/a/a/a /c /a/b/a/b	19
3 /x/y/z /x/y/z /x/y/z	3

Problem D. Democratic Naming

Source file name:	Democratic.c, Democratic.cpp, Democratic.java, Democratic.py
Input:	`Standard`
Output:	`Standard`

A new county has been created on artificially created land out from the coast, with code name "Built Anew Peninsula County", but the final name still needs to be chosen. To establish a new name, the cities within the county get to vote on the individual letters of the name.

As it happens, all cities in the county have a name with exactly m letters, and so they decide the name of the county will also have exactly m letters. Naturally, each city prefers their own name, and thus votes that the ith letter of the county name should match theirs. For each of the m positions, the letter that received the most votes across all cities gets picked. In case of a tie between multiple letters, the one occurring earliest in the English alphabet gets picked.

Artist's impression of the newly built peninsula. CC BY 2.0 by Werner Bayer on Flickr

Given the list of the city names, determine the result of the vote for the new county's name.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 1000$, $1 \leq m \leq 1000$), the number of cities and the number of letters in each city name.
- n lines, each with a string of length m, the name of a city.

The city names only consist of lowercase English letters (`a-z`).

Output

Output the name of the new county.

Example

Input	Output
3 5 apple maple alpha	aaple
3 4 icpc back laps	bapc

Problem E. Exam Study Planning

Source file name:	Examstudy.c, Examstudy.cpp, Examstudy.java, Examstudy.py
Input:	`Standard`
Output:	`Standard`

You have *a lot* of exams today! And you have not yet prepared for any of them! At least you know from experience that if you study enough for an exam, you will for sure pass it quickly, well within the allotted time. Even better, you stared so long at the daunting course curricula that you now know exactly how much time you will need to study for each exam in order to pass it within the given time. If you do not study long enough, you will for sure fail it.

Pixabay Content License by Andrew Tan on Pixabay

As it happens, your university has some weird bureaucratic rules that require you to attend *all* your exams. The horror! And leaving early is not allowed, unless you know for sure you passed it!

Given the full exam schedule of when each exam starts, how long each exam takes, how much time you need to study for each exam, and how quickly you can finish each exam you studied for, how many exams can you pass at most?

You may start studying at time 0, but can only study while not making an exam. The preparation for an exam does not have to be done in a contiguous block of time and may be interrupted.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2000$), the number of exams you have to attend.
- n lines, the ith of which contains four integers describing an exam:
  - s_i ($0 \leq s_i$), the start time of the ith exam,
  - p_i ($s_i < p_i$), the end time of the ith exam if you prepare for it,
  - e_i ($p_i \leq e_i \leq 10^9$), the end time of the ith exam if you do not prepare for it,
  - a_i ($1 \leq a_i \leq 10^9$), the time needed to prepare the ith exam.

The exams are given in ascending order of start time, and do not overlap, i.e. $e_i \leq s_{i+1}$ holds for $1 \leq i < n$.

Output

Output the maximum number of exams you can pass.

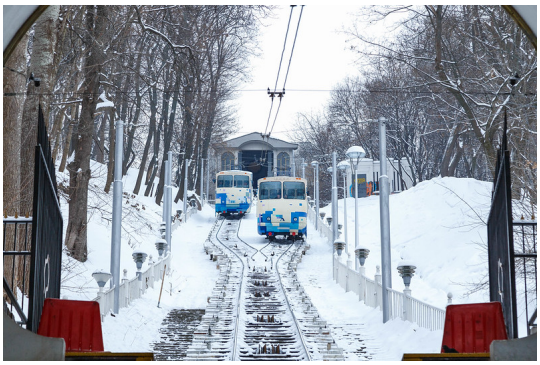
Example

Input	Output
3 10 20 30 5 30 50 100 15 100 101 200 50	3
3 1000 1001 1002 1000 1003 1004 1005 500 1006 1007 1008 500	2

Problem F. Funicular Frenzy

Source file name:	Funicular.c, Funicular.cpp, Funicular.java, Funicular.py
Input:	`Standard`
Output:	`Standard`

You are on a skiing trip in the Alps and need to take a funicular.[1] However, there usually is a long queue for the funicular to bring you to the top of the mountain. Being someone who hates wasting time in the morning, you want to find the best moment to start queueing in order to minimize queueing time.



Two funicular carriages passing each other on the halfway point of the track. CC BY 2.0 by Marco Verch on Flickr

The funicular station is open for n minutes per day. A carriage transports c people at once, and one carriage leaves exactly every minute. For every minute the funicular is open today, you know the number of people arriving.

You want to arrive when the station is open, exactly at the start of a minute, like everyone else. Note that you are a sociable person and if there are other people arriving at the same minute as you, you let them go first, after which you stand in the queue.

Calculate at which minute you should arrive to have minimal waiting time, or determine that it is impossible to catch a ride today. If there are two times achieving the same minimal waiting time, give the earliest occasion.

Input

The input consists of:

- One line with two integers n and c ($1 \leq n \leq 10^5$, $1 \leq c \leq 10^9$), the number of minutes the funicular is open today and the number of people one funicular carriage takes up the mountain each minute.
- One line with n integers $a_0, \ldots, a_{n-1}$ ($0 \leq a_i \leq 10^9$), the number of new people showing up i minutes after the funicular opens.

Output

If it is not possible to take the funicular today, output "`impossible`". Else, output the least number of minutes after opening time you should enter the queue, such that the waiting time is minimized.

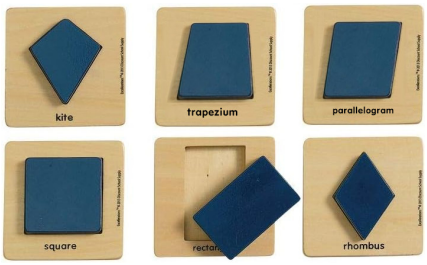
Example

Input	Output
5 1 5 0 0 0 0	impossible
5 4 8 6 4 2 0	0
10 10 12 11 10 9 8 7 6 5 4 3	5

[1] A funicular is a type of cable railway system laid on a steep slope, where two counterbalanced carriages are attached to opposite ends of a haulage cable, which is looped over a pulley at the upper end of the track.

Problem G. Geometry Game

Source file name:	Game.c, Game.cpp, Game.java, Game.py
Input:	`Standard`
Output:	`Standard`



Block puzzle with quadrilaterals. From the top left, in clockwise order: kite, trapezium, parallelogram, rhombus, rectangle, square. Modified from Excellerations® Wooden Shape Puzzles

For the longest time you could keep your toddlers happy by letting them play with triangular, square, and circular wooden blocks that fit exactly through perfectly sized holes. After letting them play a bit too long, they completely mastered this game and are now bored, preventing you from fixing the bugs in your code.

Just now, they decided to reverse roles and started screaming planar coordinates at you, insisting that you determine which shape each four points make: kite, trapezium, parallelogram, rhombus, rectangle, square, or none of those. You do not have time for this, since your bugs still need fixing. Instead, you write a new program to answer your toddlers' questions, ideally without bugs.

The definitions for the quadrilateral shapes are as follows:

- A square has four right angles and four sides with equal lengths.
- A rectangle has four right angles.
- A rhombus has four sides with equal lengths.
- A parallelogram has two pairs of parallel sides.
- A trapezium has one pair of parallel sides.
- A kite has reflection symmetry across a diagonal.

Input

The input consists of:

- Four lines, each containing two integers x and y ($0 \leq x, y \leq 10^9$), the coordinates of a point.

The positive x-axis is oriented to the right and the positive y-axis is oriented upwards.

The four points are distinct, form a convex quadrilateral shape (i.e. all interior angles are strictly less than 180°), and are given in clockwise order.

Output

Output the *most restrictive* type of quadrilateral that the points form, which is the first one of "`square`", "`rectangle`", "`rhombus`", "`parallelogram`", "`trapezium`", or "`kite`" that applies, or "`none`" otherwise.

Example

Input	Output
0 0 0 1 1 1 1 0	square
1 1 2 3 4 5 3 3	parallelogram

Problem H. Hidden Art

Source file name:	Hidden.c, Hidden.cpp, Hidden.java, Hidden.py
Input:	`Standard`
Output:	`Standard`

You have recently moved to a new home, and you are almost done decorating it. However, you still feel like something is missing: you need some art on the wall! Since you have already spent most of your budget on the furniture, you decide to go to the cheapest art shop there is: the Budget Art Printing Company (BAPC).

At the BAPC, you can buy infinitely large sheets of paper on which a decoration is printed. Such a decoration consists of a rectangular pattern which is repeated in all directions. This pattern in turn consists of square pixels that are colored white, red, green or blue. After buying a sheet of paper, customers may then cut out a part of the sheet to create their very own artwork.

You have just found a pattern of pixels you like, but before you have it printed you decide to check whether it is possible to cut a *beautiful* artwork from it. You consider an artwork beautiful if it satisfies the following properties:

- It is cut out along pixel boundaries.
- It is a square.
- The pixels in the four corners of the square have four different colors.

Is there a beautiful artwork hidden in this infinite sheet printed with the selected pattern?

As an example, consider the first sample input, visualized in Figure 1. In the infinitely repeated pattern, it is possible to find several beautiful artworks.

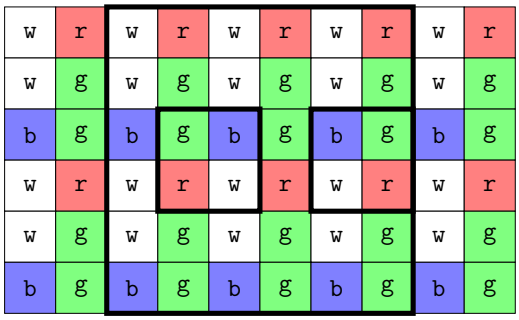


Figure 1: Visualization of the first sample input. The pattern is shown repeated five times in the horizontal direction and two times in the vertical direction, but remember that it repeats indefinitely in all directions. The three bold outlined squares indicate some possible beautiful artworks.

Input

The input consists of:

- One line with two integers h and w ($1 \leq h \leq 4000$, $1 \leq w \leq 50$), the height and width of the chosen pattern.
- h lines, each with a string of w characters c ($c \in \{\texttt{w}, \texttt{r}, \texttt{g}, \texttt{b}\}$), describing the pattern.

Output

If it is possible to cut out a beautiful artwork from the sheet with the selected pattern, output "`possible`", otherwise output "`impossible`".

Example

Input	Output
3 2 wr wg bg	possible
2 4 gbrw wbgr	impossible
6 6 bwwrrr bbbrrr bbbrwr rrrggg rrrgww rwwggg	possible

Problem I. International Irregularities

Source file name:	Irregularities.c, Irregularities.cpp, Irregularities.java, Irregularities.py
Input:	`Standard`
Output:	`Standard`

Long, long ago on a planet far, far away, a highly contagious virus caused an enduring pandemic.

Even so, the people wanted to travel between countries for their summer holidays. In the good old before-days, travelling from any country to any other country took 1 full day. However, during the pandemic, certain countries preferred not to receive travellers from areas that had higher infection rates, so they made them quarantine for a certain number of days before allowing them to continue their trip or start their holiday.

Generated using canva.com with prompt "Traveller with backpack and facemask, in front of mountains".

To keep everything fair, an independent Bureau for Accurate Pandemic Classification was founded. They assigned a r-value to each country based on the infection rate in that country. A higher r-value indicates higher infection rate.

Each country asked tourists to quarantine if the country they just came from had a r-value significantly higher than their own. In particular, when you wanted to travel from country i to country j, you would have to quarantine for t_j days if $r_i > r_j + m$.

Archaeologists have found evidence of q tourists travelling between n countries. For each tourist, the start and destination are known. The question that remains to be answered is: how long was each tourist's minimal travel time?

Input

The input consists of:

- One line with three integers n, q, and m ($2 \leq n \leq 10^5$, $1 \leq q \leq 10^5$, $0 \leq m \leq 10^9$), the number of countries, the number of tourists, and the maximum allowed difference between two r-values when travelling to a country with a lower infection rate.
- One line with n integers $r_1, \ldots, r_n$ ($0 \leq r_1 \leq \cdots \leq r_n \leq 10^9$), the r-value for each country.
- One line with n integers $t_1, \ldots, t_n$ ($0 \leq t_i \leq 10^9$ for all i), the required quarantine time in days when travelling to a country with a significantly lower r-value.
- q lines, each with two integers x and y ($1 \leq x, y \leq n$, $x \neq y$), indicating a tourist departing from country x with final destination y.

Output

For each tourist, output their minimal travel time in days between their departure country and destination country, in the order in which they appear in the input.

Example

Input	Output
5 4 1 0 5 6 7 8 3 4 1 5 10 1 4 4 1 4 2 5 2	1 4 2 3
5 4 10 0 8 20 25 30 5 11 13 6 3 5 1 5 2 5 3 5 4	6 7 1 1

Problem J. Jungle Job

Source file name:	Jungle.c, Jungle.cpp, Jungle.java, Jungle.py
Input:	`Standard`
Output:	`Standard`

Your local jungle is being taken over by monkeys! Trees are quickly being colonized by them! As a Brave Ape Pictures Collector, this is your chance of taking sooo many pictures of primates that for sure will amaze your colleagues.

In particular, each day, one new monkey discovers your favourite tree containing n branches. From your long experience observing primates, you know two things. First, each branch of the tree only has room for a single monkey. Second, monkeys are very social animals and always stick together in groups, that is, the branches they occupy form a single connected component.

This particular invasive species of monkeys happens to be new to you and you have not yet learned to distinguish them from each other. Still you wonder: how many different pictures of the monkey colony could you take on each day, until the tree is full of monkeys?

As an example, consider the first example case, visualized in Figure 2. On the third day, there are four different pictures you can take of the monkey colony.

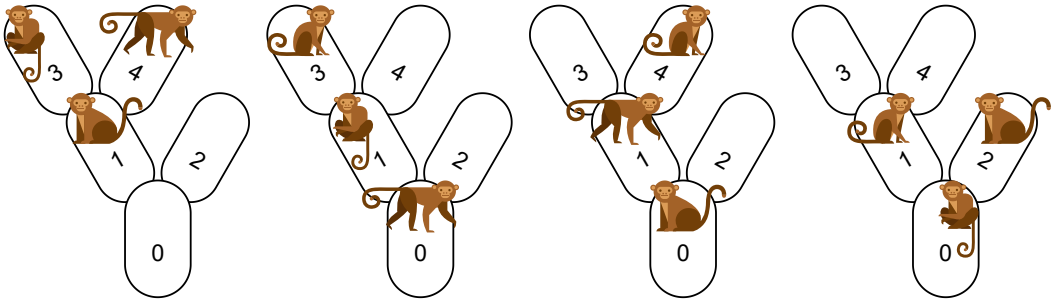


Figure 2: Visualization of the first sample case on the third day. Branches are connected if they overlap (note the small gap between branches 1 and 2, and between branches 3 and 4).
Monkey image from freevector.com

Given the exact structure of your favourite tree, determine for each day from 1 to n the number of different sets of branches the monkeys could occupy on that day, modulo $10^9 + 7$.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1000$), the number of branches in the tree.
- $n - 1$ lines, the ith of which ($1 \leq i \leq n - 1$) contains an integer p_i ($0 \leq p_i < i$), indicating that branch i is connected to the upper end of branch p_i.

The branches are numbered from 0 to $n - 1$, inclusive. Branch 0 is connected to the roots of the tree and can also host a single monkey.

Output

Output n integers. The kth integer should be the number of connected subtrees that consist of exactly k branches, modulo $10^9 + 7$.

Example

Input	Output
5 0 0 1 1	5 4 4 3 1
3 0 1	3 2 1

Problem K. Kindergarten Excursion

Source file name:	Kindergarten.c, Kindergarten.cpp, Kindergarten.java, Kindergarten.py
Input:	`Standard`
Output:	`Standard`

The kindergarten teachers had finally managed to get all the kids in a line for the walk to the bus station and the weekly excursion. What they hadn't thought of was the fact that today different kids were supposed to go to different excursions. They should walk as one line to the bus station, but to avoid total chaos when they arrive there, all the kids going to the zoo should be in the beginning of the line, the ones going to the lake should be in the middle and the rest, going to the science museum, should be in the end.

Since it takes a lot of time to get all the kids to stand in a line no kid may step out of the line. To get the line organized after excursion group, kids standing next to each other can swap places. Now the kindergarten teachers wonder, if they will have time to do this and still catch their bus.

You will be given a sequence of numbers 0, 1, and 2, denoting the excursion destinations of each kid from first to last in the line. Pairs of adjacent kids can swap positions, and the line should be organized after destination number starting with 0 and ending with 2. What is the minimum number of swaps required to organize the line?

Input

The only line of input contains a string consisting of characters 0, 1 or 2, denoting the destinations of kids. The length of the string will be at most 10^6 characters.

Output

Output one line with one integer - the minimum number of swaps needed to get the kids in order.

Example

Input	Output
10210	5

Problem L. Locking Doors

Source file name:	Locking.c, Locking.cpp, Locking.java, Locking.py
Input:	`Standard`
Output:	`Standard`

You just started a new job at a shopping mall, and as it goes, you got the shittiest task of all: closing down at night. The mall consists of many rooms (which can be shops, hallways, or other public spaces) with open doors between them that must be closed. You can walk through a door both ways if it is open, but annoyingly, each door can only be locked from one of the two rooms it connects.

Locking the door, possible from one side only. CC BY 2.0 by Marco Verch on Flickr

Your supervisor already locked the main entrance of the shopping mall, and left you inside with the task to lock all the doors. In order to do so, you may request additional exits to be installed in some of the rooms. If a room has an exit, then you are able to enter or leave the shopping mall through that room.

What is the minimal number of exits you need to request in order for it to be possible to lock all the doors and then leave the building?

Input

The input consists of:

- One line with two integers n and m ($2 \leq n \leq 10^5$, $1 \leq m \leq 10^6$), the number of rooms and doors, respectively.
- Then follow m lines, each containing two distinct integers a and b ($1 \leq a, b \leq n$, $a \neq b$), indicating a door connecting rooms a and b which can only be locked from room a.

You may assume that all ordered pairs (a, b) are distinct and that you can walk from any room in the mall to any other room if all the doors are open.

Output

Output the minimal number of exits that need to be installed.

Example

Input	Output
2 1 1 2	1
3 2 2 1 3 1	2
5 4 1 2 3 1 4 1 5 1	3
10 14 1 2 2 1 3 4 3 5 4 6 5 6 6 3 7 8 8 9 9 7 1 8 2 10 4 9 5 10	2