Problem X (don't pick as seminar topic)

When writing text, people sometimes rely on braces to inject additional details to certain statements. Due to human defectiveness, braces (especially closing ones are prone to be missing. A different race – the compilers – heavily relies on braces to be balanced and well nested. To prevent misapprehensions and ensure proper interaction between humans and compilers, all humans' texts need to be verified prior to passing them on to the compilers.

Did you know that . . . ?

. . . this problem was originally named *Brace Yourself*? Amazing name, isn't it?

The pairs of characters `( )`, `{ }`, `[ ]`, and `< >` are used as control symbols in source code and each define a *control block*. An opening brace (i.e., one of `({[<`) marks the start of a control block, the corresponding closing brace (i.e., the matching brace `)}]>`) marks the end of the control block. Control blocks can encapsulate other control blocks, but no two control blocks may intersect each other. A text may also contain *string literals*, which are character sequences delimited by a pair of quotation marks `"`. Quotation marks cancel out control characters, which means that after the first quotation mark no character is interpreted as control character until the next quotation mark. Write a source code check that tests if all string literals and control blocks are valid and closed.

Input

The input consists of:

- One or more lines (separated by `\n`) made up of whitespace, numbers, alphabetic characters, and all other printable ASCII characters. More precisely, the set of allowed characters is the ASCII range `0x20`-`0x7e`, as well as `0x09` (tab) and `0x0a` (newline).

The total number of characters in the input is at most 1 048 576 (= 1 MiB).

Output

Output `correct` if all control blocks and strings are valid and closed or `incorrect` if not.

Sample Input 1	Sample Output 1

`{ [ < "this is a > string" > ]`
`this is not a string } ()`
	`correct`

Sample Input 2	Sample Output 2

`{ [ < "this is a > string " ]`
`this is not a string } ()`
	`incorrect`

Problem A
Picky Servers

As part of his new job at the IT security department of his company, Bob got the task to track how fast messages between the companies' different offices are transmitted through the internet. Since some offices are currently rebuilt and not finished yet he is not able to simply measure the transmission speed but needs to compute it somehow. Therefore, Bob created a map of all servers that may be involved in routing the packages through the internet. He also gathered the times each server needs to process a message. The total processing time of a message is the sum of the processing times of the sender, all servers along the path, and the receiver. Furthermore, Bob read that messages are sent through the network of servers along a path such that the total processing time of all servers on the path is minimal.

Bob thought that it might be an easy problem to compute the total transmission time between two offices, but he forgot the intelligence agencies! Each server on the internet is controlled by some agency that can decide which packages are routed and which of them are not. All servers are configured in a way that they read all incoming data, since gathering all kind of information is exactly what the intelligence agencies want to do, but not all data is forwarded to other servers.

Each server has a list of pairs of other servers such that messages from the first of them are not sent to the second one. Can you still help Bob to compute how fast his messages will be transmitted?

Input

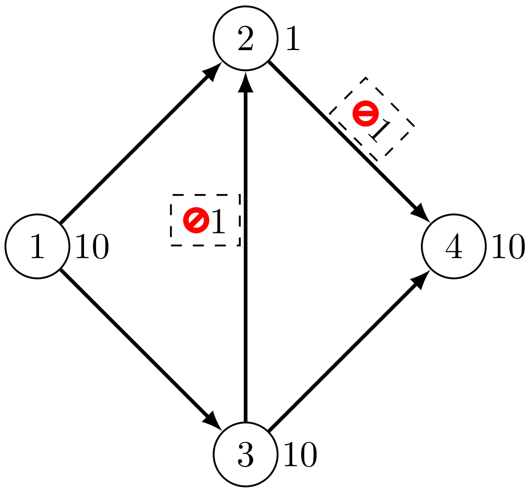
The input consists of:

- one line with an integer n ($2 \leq n \leq 100$), where n is the number of servers labeled from 1 to n;
- n blocks describing the servers. Each server is described by:
  - one line with two integers m ($0 \leq m \leq n-1$) and t ($0 \leq t \leq 1000$), where m is the number of outgoing connections from this server and t is the processing time of this server;
  - m lines with two integers s ($0 \leq s \leq n-1$) and x ($1 \leq x \leq n$) and s more distinct integers $a_1, \ldots, a_s$ ($1 \leq a_j \leq n$, $a_j \neq i$ for all $1 \leq j \leq s$) indicating that server i sends messages to server x, but only if it was not directly transmitted from one of the servers $a_1, \ldots, a_s$ to server i.

Bob's messages start at server 1 and should go to server n.

Output

Output the sum of the processing times of all servers on the shortest path for Bob's message including the first and last one, or "`impossible`" if there is no such path.



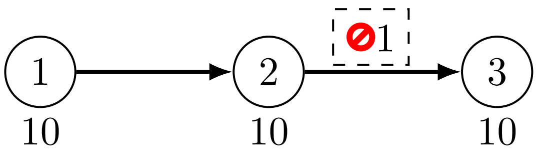


Figure K.1: Illustration of the sample inputs.

Sample Input 1

```
4
2 10
0 2
0 3
1 1
1 4 1
2 10
1 2 1
0 4
0 10
```

Sample Output 1

```
30
```

Sample Input 2

```
3
1 10
0 2
1 10
1 3 1
0 10
```

Sample Output 2

```
impossible
```

Problem B: **Avast!**

The world is at the brink of extinction. A mutated virus threatens to destroy all living organisms. As a last hope, a team of super-smart scientists, including – of course – you, is currently working on an antivirus. Unfortunately, your team is unable to analyse the DNA in time. They sequenced n parts of the virus' DNA and need to match them with n available strands for antiviruses. As the algorithms expert, you need to implement a specialised procedure to solve this problem. Your approach needs to be fast – there is not much time left!

You first need to determine the *repetition score* of each DNA sequence. The repetition score of a sequence s is equal to the length of the shortest sequence u such that s is equal to the k-fold repetition of u, for some positive integer k. For instance, `ATGATG` has a repetition score of 3, since it can be produced by repeating `ATG` two times. On the other hand, `ATATA` has a repetition score of 5, as it cannot be produced from any proper substring.

Once you obtained the scores of all sequences, you need to match the n antivirus sequences with the n virus sequences in a way that minimises the damage caused by the virus. When two sequences are matched, the damage caused by the virus is equal to the squared difference between the two repetition scores. For instance, matching the antivirus sequence `ATGATG` with the virus sequence `ATATA` causes $(3-5)^2 = 4$ units of damage.

If you match the DNA sequences optimally, what is the minimal total damage caused by the virus, taken as a sum over all matched pairs?

Input

The input consists of:

- A line with an integer n ($1 \leq n \leq 50$), the number of DNA sequences of the virus and antivirus each.
- n lines, each with a virus DNA sequence.
- n lines, each with an antivirus DNA sequence.

Each DNA sequence is a non-empty string with a length of at most 250 and consists of lowercase letters `a-z` and uppercase letters `A-Z`.

Output

Output one integer, the minimal total damage.

Sample Input 1

```
2
TTTTTT
TATG
TATATA
AAAGAAAG
```

Sample Output 1

```
1
```

Sample Input 2

```
3
abcdef
aaa
bab
AbAb
xyzxyz
X
```

Sample Output 2

```
10
```

Problem C: **Teamwork**

Din Djarin the bounty hunter is tracking another criminal through space. Luckily for him hyperspace travel has made the task of visiting several planets a lot easier. Each planet has a number of Astral Gates; each gate connects with a gate on another planet. These hyperspace connections are, for obvious safety reasons, one-way only with one gate being the entry point and the other gate being the exit point from hyperspace. Furthermore, the network of hyperspace connections must be loop-free to prevent the Astral Gates from exploding, a tragic lesson learned in the gate accident of 2022 that destroyed most of the moon.
While looking at his star map Din wonders how many friends he needs to conduct a search on every planet. Each planet should not be visited by more than one friend otherwise the criminal might get suspicious and flee before he can be captured. While each person can start at a planet of their choosing and travel along the hyperspace connections from planet to planet they are still bound by the limitations of hyperspace travel.

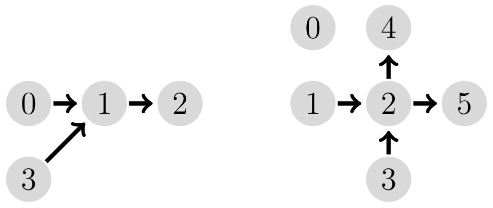


Figure B.1: Illustration of the Sample Inputs.

Input

The input begins with an integer N specifying the number of planets ($0 < N \leq 1000$). The planets are numbered from 0 to $N-1$. The following N lines specify the hyperspace connections. The i-th of those lines first contains the count of connections K ($0 \leq K \leq N-1$) *from* planet i followed by K integers specifying the destination planets.

Output

Output the minimum number of persons needed to visit every planet.

Sample Input 1

```
4
1 1
1 2
0
1 1
```

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
4
```

Problem D: **Toving Liles**

The computer science Professor Toving Liles loves the floor tiles in his office so much that he wants to protect them from damage by careless students. Therefore, he would like to buy cheap small rectangular carpets from the supermarket and cover the floor such that:

1. The entire floor is covered.
2. The carpets do not overlap.
3. The carpets are rotated arbitrarily.
4. No carpet is cut into pieces.

But when checking the supermarket's stock he begins to wonder whether he can accomplish his plan at all. Can you help him?

Input

The first line contains two integers W and H describing the size of his room ($1 \leq W, H \leq 100$).
The second line contains an integer c, denoting the number of different carpet colors the supermarket has in stock ($1 \leq c \leq 7$).
Each of the following c lines consists of three integers a_i, w_i, and h_i, which means: the supermarket's stock contains a_i carpets of size w_i, h_i and color i ($1 \leq a_i \leq 7$; $1 \leq w_i \leq 100$; $1 \leq h_i \leq 100$).
The supermarket has at most 7 carpets, i.e. $\sum_i a_i \leq 7$.

Output

For the given room dimensions and the supermarket's stock of carpets, print "`yes`" if it is possible to cover the room with carpets as specified above and "`no`" otherwise.

Sample Input 1

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

Sample Output 1

```
yes
```

Sample Input 2

```
100 100
3
4 42 42
1 100 16
1 32 42
```

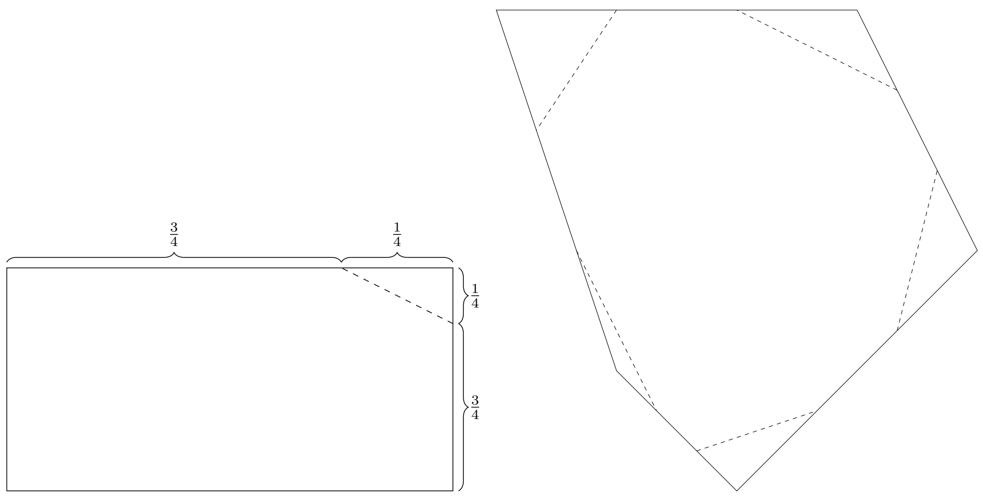
Sample Output 2

```
no
```

Problem E: **A problem of packaging**

Sophie loves to bake cakes and share them with friends. For the wedding of her best friend Bea she made a very special cake using only the best ingredients she could get and added a picture of the engaged couple on top of the cake. To make it even more special she did not make it round or square, but made a custom convex shape for the cake. Sophie decided to send the cake by a specialized carrier to the party. Unfortunately, the cake is a little too heavy for their default cake package and the overweight fees are excessive. Therefore, Sophie decides to remove some parts of the cake to make it a little lighter.

Sophie wants to cut the cake the following way: First, she chooses a real number $s \geq 2$. For each vertex and each incident edge of the cake she marks where $1/s$ of the edge's length is. Afterwards, she makes a direct cut between the two markings for each vertex and removes the vertex that way.



(a) Cutting the upper-right corner of a rectangle with $s = 4$

(b) Cutting a cake with $s = 3$

Figure C.1: Illustration of the first two Sample Inputs.

Sophie does not want to cut more from the cake than necessary for obvious reasons. Can you tell her how to choose s?

Input

The first line contains a floating point number a and an integer N, where a denotes the ratio of the cake's weight allowed by the carrier and N the number of vertices of the cake ($0.25 \leq a < 1$; $3 \leq N \leq 100$). a will be specified with at most 7 digits after the decimal point.
Then follow N lines, each describing one of the cake's vertices with two integers x_i and y_i, the coordinates of the vertex ($0 \leq x_i, y_i \leq 10^8$ for all $1 \leq i \leq N$). The vertices are given in the order in which they form a strictly convex shape.
You may safely assume that the weight is uniformly distributed over the area of the cake. Furthermore, it will always be possible to cut the cake with some $2 \leq s \leq 1000$ such that the proportion of the remaining cake is a of the original weight.

Output

Print a line containing s, the biggest value as specified above such that the remaining cake's weight is at most the proportion a of its original weight.
Your answer will be considered correct if the absolute error is at most 10^{-4}.

Sample Input 1

```
0.875 4
0 0
8 0
8 4
0 4
```

Sample Output 1

```
4.000000
```

Sample Input 2

```
0.85 5
6 0
12 6
9 12
0 12
3 3
```

Sample Output 2

```
3.000000
```

Sample Input 3

```
0.999998 4
20008 10000
15004 15005
10001 20009
15005 15004
```

Sample Output 3

```
1000.000000
```

Problem F: **Telephone a taxi**

For a long time Tim wanted to visit Greece. He has already purchased his flight to and from Athens. Tim has a list of historical sites he wants to visit, e.g., Olympia and Delphi. However, due to recent political events in Greece, the public transport has gotten a little complicated. To make the Greek happy and content with their new government, many short-range bus and train lines have been created. They shall take the citizens around in their neighborhoods, to work or to their doctor. At the same time, long-range trains that are perfect for tourists have been closed down as they are too expensive. This is bad for people like Tim, who really likes to travel by train. Moreover, he has already purchased the Greece' Card for Public Conveyance (GCPC) making all trains and buses free for him.

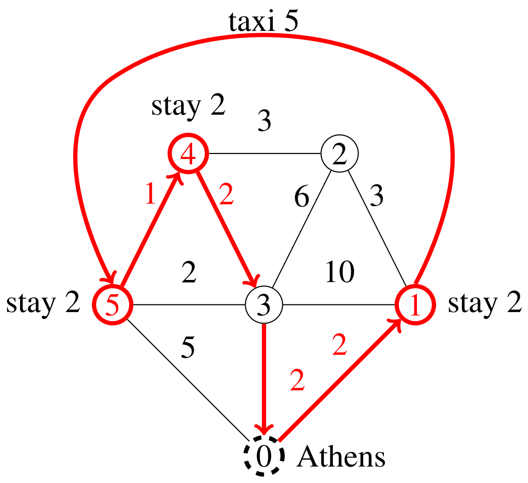


Figure A.1: Visual representation of the Sample Input: Tim's tour has length 18.

Despite his preferred railway lines being closed down, he still wants to make his travel trough Greece. But taking all these local bus and train connections is slower than expected, so he wants to know whether he can still visit all his favorite sites in the timeframe given by his flights. He knows his schedule will be tight, but he has some emergency money to buy a single ticket for a special Greek taxi service. It promises to bring you from any point in Greece to any other in a certain amount of time.
For simplicity we assume, that Tim does never have to wait for the next bus or train at a station. Tell Tim, whether he can still visit all sites and if so, whether he needs to use this taxi ticket.

Input

The first line contains five integers N, P, M, G and T, where N denotes the number of places in Greece, P the number of sites Tim wants to visit, M the number of connections, G the total amount of time Tim can spend in Greece, and T the time the taxi ride takes ($1 \leq N \leq 2 \cdot 10^4$; $1 \leq P \leq 15$; $1 \leq M, G \leq 10^5$; $1 \leq T \leq 500$).
Then follow P lines, each with two integers p_i and t_i, specifying the places Tim wants to visit and the time Tim spends at each site ($0 \leq p_i < N$; $1 \leq t_i \leq 500$). The sites p_i are distinct from each other.
Then follow M lines, each describing one connection by three integers s_i, d_i and t_i, where s_i and d_i specify the start and destination of the connection and t_i the amount of time it takes ($0 \leq s_i, d_i < N$; $1 \leq t_i \leq 500$).
All connections are bi-directional. Tim's journey starts and ends in Athens, which is always the place 0.

Output

Print either "`impossible`", if Tim cannot visit all sites in time, "`possible without taxi`", if he can visit all sites without his taxi ticket, or "`possible with taxi`", if he needs the taxi ticket.

Sample Input 1

```
6 3 10 18 5
1 2
4 2
5 2
0 1 2
1 2 3
2 4 3
1 3 10
2 3 6
0 3 2
3 4 2
4 5 1
3 5 2
0 5 5
```

Sample Output 1

```
possible with taxi
```

Problem G
Patent Claims

The year is 1902. Albert Einstein is working in the patent office in Bern. Many patent proposals contain egregious errors; some even violate the law of conservation of energy. To make matters worse, the majority of proposals make use of non-standard physical units that are not part of the metric system (or not even documented). All proposals are of the following form:

- Every patent proposal contains n energy converters.
- Every converter has an unknown input energy unit associated with it.
- Some energy converters can be connected: If converter a can be connected to converter b such that one energy unit associated with a is turned into c input units for b, then this is indicated by an arc $a \xrightarrow{c} b$ in the proposal. The output of a can be used as input for b if and only if such an arc from a to b exists.

Einstein would like to dismiss all those proposals out of hand where the energy converters can be chained up in a cycle such that more energy is fed back to a converter than is given to it as input, thereby violating the law of conservation of energy.

Einstein's assistants know that he is born for higher things than weeding out faulty patent proposals. Hence, they take care of the most difficult cases, while the proposals given to Einstein are of a rather restricted form: Every *admissible* patent proposal given to Einstein does not allow for a cycle where the total product of arc weights exceeds 0.9. By contrast, every *inadmissible* patent proposal given to Einstein contains a cycle where the the number of arcs constituting the cycle does not exceed the number of converters defined in the proposal, and the total product of arc weights is greater or equal to 1.1.

Could you help Einstein identify the inadmissible proposals?

Input

The input consists of:

- one line with two integers n and m, where
  - n ($2 \leq n \leq 800$) is the number of energy converters;
  - m ($0 \leq m \leq 4000$) is the number of arcs.
- m lines each containing three numbers a_i, b_i, and c_i, where
  - a_i and b_i ($1 \leq a_i, b_i \leq n$) are integers identifying energy converters;
  - c_i ($0 < c_i \leq 5.0$) is a decimal number

  indicating that the converter a_i can be connected to the converter b_i such that one input unit associated with a_i is converted to c_i units associated with b_i. The number c_i may have up to 4 decimal places.

Output

Output a single line containing `inadmissible` if the proposal given to Einstein is inadmissible, `admissible` otherwise.

Sample Input 1

```
2 2
1 2 0.5
2 1 2.3
```

Sample Output 1

```
inadmissible
```

Sample Input 2

```
2 2
1 2 0.5
2 1 0.7
```

Sample Output 2

```
admissible
```

Problem H
Single-lane Serpentine

In Franconian Switzerland, there is a narrow mountain road. With only a single lane, this is a bottleneck for two-way traffic. Your job is to schedule incoming cars at both ends so that the last car leaves the road as early as possible.

Each car is specified by three values: the direction in which it is going, the arrival time at the corresponding beginning of the road, and the driving time this car needs to get through, provided it is not slowed down by other cars in front. Cars cannot overtake each other on the mountain road, and reordering cars in the queues at the ends of the road is not allowed.

For safety reasons, two successive cars going in the same direction may not pass any point of the road within less than 10 seconds. This ensures that the second car will not crash into the first car if the latter brakes hard. However, if another car passes in the other direction in between, it will be clear that the road is empty, so in this case, this rule does not apply.

Input

The first line of the input consists of a single integer c ($1 \leq c \leq 200$), the number of test cases.

Then follow the test cases, each beginning with a single line consisting of an integer n ($1 \leq n \leq 200$), the number of cars you are to consider in this test case. The remainder of each test case consists of n lines, one line per car, starting with a single upper case letter ("`A`" or "`B`"), giving the direction in which the car is going. Then follow, on the same line, two integers t ($0 \leq t \leq 100\,000$) and d ($1 \leq d \leq 100\,000$), giving the arrival time at the beginning of the road and the minimum travel time, respectively, both in seconds.

Within a test case, the cars are given in order of increasing arrival time, and no two cars will arrive at the same time.

Output

For each test case, print a single line consisting of the point in time (in seconds) the last car leaves the road when the cars are scheduled optimally.

Sample Input

```
2
4
A 0 60
B 19 10
B 80 20
A 85 100
4
A 0 100
B 50 100
A 100 1
A 170 100
```

Sample Output

```
200
270
```

Problem K
Flipping out

Once there was an inventor congress, where inventors from all over the world met in one place. The organizer of the congress reserved exactly one hotel room for each inventor. Each inventor, however, had its own preference regarding which room he would like to stay in. Being a clever inventor himself, the organizer soon found an objective way of doing the room assignments in a fair manner: each inventor wrote two different room numbers on a fair coin, one room number on each side. Then, each inventor threw his coin and was assigned the room number which was shown on the upper side of his coin. If some room had been assigned to more than one inventor, all inventors had to throw their coins again.
As you can imagine, this assignment process could take a long time or even not terminate at all. It has the advantage, however, that among all possible room assignments, one assignment is chosen randomly according to a uniform distribution. In order to apply this method in modern days, you should write a program which helps the organizer.
The organizer himself needs a hotel room too. As the organizer, he wants to have some advantage: he should be able to rate each of the rooms (the higher the rating, the better), and the program should tell him which two room numbers he should write on his coin in order to maximize the expected rating of the room he will be assigned to. The program also has access to the choices of the other inventors before making the proposal. It should never propose two rooms for the organizer such that it is not possible to assign all inventors to the rooms, if a valid assignment is possible at all.

Input

The input starts with a single number c ($1 \leq c \leq 200$) on one line, the number of test cases. Each test case starts with one line containing a number n ($2 \leq n \leq 50\,000$), the number of inventors and rooms. The following $n-1$ lines contain the choices of the $n-1$ guests (excluding the organizer). For each inventor, there is a line containing two numbers a and b ($1 \leq a < b \leq n$), the two room numbers which are selected by the inventor. The last line of each test case consists of n integers $v_1, \dots, v_n$ ($1 \leq v_i \leq 1\,000\,000$), where v_i is the organizer's rating for room i.

Output

For each test case, print a single line containing the two different room numbers a and b which should be selected by the organizer in order to maximize the expected rating of the room he will be assigned to. If there is more than one optimal selection, break ties by choosing the smallest a and, for equal a, the smallest b. If there is no way for the organizer to select two rooms such that an assignment of inventors to rooms is possible, print "`impossible`" instead.

Sample Input

```
3
4
1 2
2 3
1 3
2 3 4 1
3
1 2
2 3
100 40 70
5
1 2
1 2
1 2
3 4
1 1 1 1 1
```

Sample Output

```
1 4
1 3
impossible
```

Problem L: **Molvanian Castles**

The royal castles in Molvania follow the design of king Sane, first of his dynasty. He ruled by divide and conquer. Therefore, all castles are built according to a hierarchical pattern based on interconnected buildings. A building consists of *halls* and *corridors* that connect halls.

Initially, a castle consists of only one building (the *main building*). When its population grows, the castle is extended as follows: A new peripheral building is constructed, attached to one of the existing buildings. Like any other building, the new building also consists of halls and corridors. An additional corridor is created to connect a hall in the existing building to a hall in the new building. That corridor is the only way to access the new building.

The number of halls in a building is at most 10.

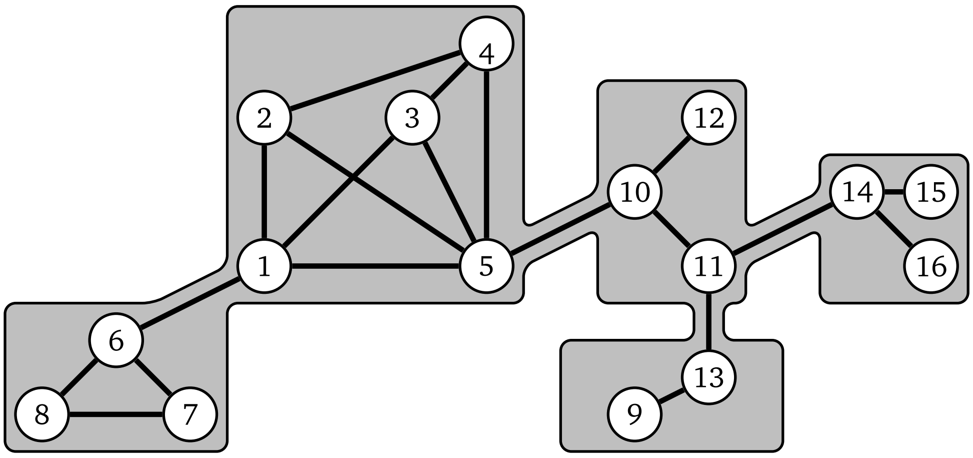


Figure 1: The castle layout of the example provided below.

In times of turmoil, the king monitors all corridors by strategically placing guards in halls. He asks you to determine the least number of guards required to monitor all corridors in the castle (as he wants to keep his personal guard as large as possible). Note that since the last fire, there are no doors in the castle, so we can safely assume that a guard placed in a hall can monitor all connecting corridors.

Input

The input contains a number of castle descriptions. Within a castle, each hall is identified by a unique number between 1 and 10000. Each castle is recursively defined, starting with a description of the main building:

1. A line containing three integers, representing the number of halls in this building ($2 \leq n \leq 10$), the number of corridors in this building ($1 \leq m \leq 45$), and the number of peripheral buildings that were later attached to this building ($0 \leq w \leq 10$).
2. For each of the m corridors:
   - A line containing two integers (each $\leq$ 10000), representing the two halls connected by this corridor. Both halls are located inside the current building.

3. For each of the w peripheral buildings:
    - A line containing two integers, describing the corridor that leads to this peripheral building. The first integer represents a hall in the current building, while the second integer represents a hall in the peripheral building.
    - The structure of the peripheral building and any newer buildings that were later attached to it, described by repeating rules 1 to 3.

The castle is fully connected: any hall is directly or indirectly reachable from any other hall. Corridors with the same start and end hall do not exist, and for every two halls there is at most one corridor between them.

Output

For each castle, print a single line containing a positive integer: the minimum number of guards to place in halls such that all corridors in the castle are monitored.

Example

input	output
5 8 2	8
1 2	
2 4	
3 4	
1 3	
1 5	
2 5	
3 5	
4 5	
1 6	
3 3 0	
6 7	
7 8	
8 6	
5 10	
3 2 2	
10 11	
10 12	
11 13	
2 1 0	
13 9	
11 14	
3 2 0	
14 15	
14 16	

Problem M: **Inconspicuous Hacking**

Karl is competing in the preliminary round of a talent show called North-Western European Idol (NWEI), and wants to advance to the next round: World Idol. In the talent show, each contestant gets 10 minutes to impress the judges. After all the contestants have performed, each of the judges will cast two distinct votes. A vote can be either in favour of a contestant (meaning this contestant should advance) or against a contestant (meaning this contestant should not advance). The number of contestants that advance to the next round is not known in advance; if there are only very bad contestants, then it is possible that nobody will advance, or if everybody is amazing, then everybody may advance.

Karl is afraid that the judges will not appreciate his programming talents, and hence wants to use his other talent to advance to the next round: hacking. Having gained access to the jury system, Karl is capable of overriding the regular process of counting votes, and instead selecting exactly which contestants advance to the next round. The only problem is, he has to be careful not to arouse suspicion.

Each judge expects that at least one of his (or her) own two votes corresponds to the outcome of the contest. If the outcome contradicts both votes, the judge becomes alarmed. As an example, assume judge Harry casts a vote in favour of Pete and a vote against Sally. If Sally advances and Pete does not, judge Harry will be alarmed and may discover Karl's tampering with the system.

Since Karl's programming talents are limited (otherwise he would not have needed his hacking talents), he needs you to make a program that finds out if there is a set of contestants, which includes himself, that he can select to advance to the next round by hacking the jury system, such that it does not alarm any of the judges.

Input

For each test case, the input is as follows:

- One line containing two positive integers: the number of contestants n ($2 \le n < 1000$) and the number of judges m ($1 \le m < 2000$).
- m lines containing the votes of each judge.
  Each of these line contains two integers: the numbers a ($1 \le |a| \le n$), and b ($1 \le |b| \le n$), the two votes of this judge ($|a| \neq |b|$).

  A vote $x < 0$ means that the vote is against advancement of contestant $|x|$.
  A vote $x > 0$ means that the vote is in favour of contestant $|x|$.

Contestants are numbered $1 \dots n$.
Karl is contestant 1.

Output

For each test case, print one line of output containing the word '`yes`' if there is a set of contestants that advances to the next round that includes Karl, and does not alarm any of the judges. If there is no such set of contestants, the line should contain '`no`'.

Example

input	**output**

`4 3`
`1 2`
`-2 -3`
`2 4`
`2 4`
`1 2`
`1 -2`
`-1 2`
`-1 -2`
	`yes`
	`no`

Problem N
Around the Track

In order to compare race tracks, we wish to compute their lengths. A racetrack is strictly two-dimensional (no elevation). It is described by two simple polygons, where one is completely contained inside the other. The track is the region between these two polygons. We define the length of the track as the absolute minimum distance that one needs to travel in order to complete a lap. This could involve traveling on the very edge of the track and arbitrarily sharp cornering (see Figure A.1).

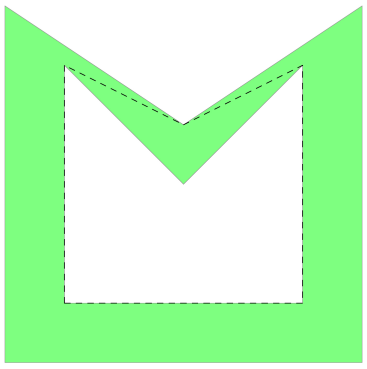


Figure A.1: Illustration of sample input number 3 together with the shortest route around the track (dashed).

Input

The input consists of:

- one line with one integer n ($3 \leq n \leq 50$), the number of vertices of the inner polygon;
- n lines, the ith of which contains two integers x_i and y_i ($-5\,000 \leq x_i, y_i \leq 5\,000$): the coordinates of the ith vertex of the inner polygon;
- one line with one integer m ($3 \leq m \leq 50$), the number of vertices of the outer polygon;
- m lines, the ith of which contains two integers x_i and y_i ($-5\,000 \leq x_i, y_i \leq 5\,000$): the coordinates of the ith vertex of the outer polygon.

For both polygons, the vertices are given in counterclockwise order. The borders of the two polygons do not intersect or touch each other.

Output

Output one line with one floating point number: the length of the race track. Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1

```
3
1 1
2 1
1 2
3
0 0
4 0
0 4
```

Sample Output 1

```
3.41421356237309
```

Sample Input 2

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

Sample Output 2

```
16
```

Sample Input 3

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

Sample Output 3

```
16.4721359549996
```