Problem A. Perfect Bombing

Input file:	`bombing.in`
Output file:	`bombing.out`
Time limit:	1 second
Memory limit:	64 megabytes

The government of one terrorism-supporting countries has decided to destroy its main enemy — the United States of Antarctica. To do it, they prepared n nuclear bombs — to destroy the n main cities of the United States. But besides the devastating effect of destroying the main cities, the terrorists want people around the country to panic. The government expects to blow up one bomb a week. To ensure the panic spreads around the country, they want news to spread as fast as possible after each explosion.

The terrorists know that the United States of Antarctica have a good system of communication channels between different cities. When the city is destroyed by the bomb, all cities connected to the destroyed city via communication channels quickly get the horrible information, and the news is spread there. But the terrorists want more — they want people from different cities to exchange their opinions. So the neighbouring cities must in turn be connected by communication channels to each other.

The government calls the bombing order *perfect*, if each time after destroying a city, all of its still existing neighbours are directly connected to each other by communication channels. The Minister of Offence have prepared the order in which the cities must be destroyed, but the Prime Minister does not trust him completely. So he asked you, the Minister of Information Technology, to check whether the given order is indeed the perfect bombing order.

Input

The first line of the input file contains n — the number of cities to destroy, and m — the number of communication channels ($1 \le n \le 5000$, $0 \le m \le 500\,000$). The second line contains n different integer numbers — the order in which cities are supposed to be destroyed. The following m lines describe communication channels. Each channel is specified with the cities it connects. Any two cities are connected by at most one channel, no channel connects a city to itself. All channels are bidirectional.

Output

If the given order is indeed a perfect bombing order, print "`YES`" on the first line of the output file. In the other case print "`NO`".

Example

bombing.in	bombing.out
5 7 3 2 1 4 5 1 2 1 3 1 4 1 5 2 3 2 5 4 5	YES
4 4 1 2 3 4 1 2 1 4 2 3 3 4	NO

Problem B. Fixed Point

Input file:	`fixed.in`
Output file:	`fixed.out`
Time limit:	1 second
Memory limit:	64 megabytes

Consider a finite alphabet Σ. A function $\varphi : \Sigma \to \Sigma^*$ is called a *morphism*. For example, if $\Sigma = \{a, b, c\}$, a function $\varphi(a) = ab$, $\varphi(b) = \varepsilon$, $\varphi(c) = bca$ is a morphism. Here ε denotes an empty string.

Morphism can be extended to a set Σ^* of strings in a natural way:

$$\varphi(c_1 c_2 \dots c_l) = \varphi(c_1)\varphi(c_2)\dots\varphi(c_l).$$

For example, if we consider the morphism described above, $\varphi(abc) = abbca$.

Consider a set Σ^ω of one-side infinite strings. Such string can be interpreted as a function $f : \mathbb{N} \to \Sigma$ that for each position i identifies a character $c_i = f(i)$ that occurs at this position. For the purpose of this problem all infinite strings are one-side infinite, so we will omit "one-side" later on.

We can easily extend our morphism to a set of infinite strings in the following way:

$$\varphi(c_1 c_2 \dots c_l \dots) = \varphi(c_1)\varphi(c_2)\dots\varphi(c_l)\dots.$$

Considering again the sample morphism defined in the first paragraph, we have $\varphi(bababa\dots) = abababab\dots$. Note, that the value of φ for an infinite string may be finite, for example $\varphi(abbbbb\dots) = ab$.

We call a (finite or infinite) string s a *fixed point* of the morphism, if $\varphi(s) = s$. For example, ab, $ababa\dots$, $bcaababababab\dots$ are all fixed points of our sample morphism.

Given a morphism φ, and an integer k, find the number of its finite fixed points with length not exceeding k, and the number of its infinite fixed points.

Input

The first line of the input file contains n — the number of characters in Σ and k ($1 \le n \le 26$, $1 \le k \le 1\,000$). The following n lines contain values for $\varphi(c_i)$. We use first n small English letters as characters from Σ. Length of $\varphi(c_i)$ does not exceed 5 for any c_i.

Output

Output two numbers — the number finite fixed points of the given morphism with length not exceeding k, and the number of its infinite fixed points. Use -1 to denote infinity.

Example

fixed.in	fixed.out
3 10 ab bca	6 -1

In the given example the finite fixed points with length not exceeding 10 are: ε, ab, $abab$, $ababab$, $abababab$, and $ababababab$.

Problem C. Paragraph Formatting

Input file:	`formatting.in`
Output file:	`formatting.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Mocrosoft company is developing the new publishing software. You are responsible for the module that will do a paragraph layout. You must split the paragraph into lines so that it looked nice.

The paragraph layout problem is well studied, and there are many models that define the value to optimize. One of the most ugly artefacts of the text is the "road" of whitespaces that occurs if three or more whitespaces are one above the other. So the customers asked that the software must avoid such situations as much as possible. This is the problem you must now solve.

You are given one paragraph and must split it into lines to minimize the penalty. The paragraph consists of words. The words are separated by spaces and line feeds. You are not allowed to break words and use hyphenation. The penalty is defined as the sum of lengths of whitespace intervals, such that there is a whitespace immediately above it, and whitespace immediately below it.

All lines except the last one must contain at least two words. The sum of the lengths of the words must not exceed $p - c + 1$ where p is text width, and c is the number of words. The first word of the line is aligned to the left, the last word — to the right. All other words are located in such a way, that all whitespaces have equal width. Since this is the modern software, whitespaces may have a fractional length. However, we consider all characters to have the same width equal to one.

The last line may contain one or more words. Again, the sum of the lengths of the words must not exceed $p - c + 1$ where p is text width, and c is the number of words. The first word of the line is aligned to the left. All other words are located in such a way, that all whitespaces have the width equal to one. The whitespace in the end of the last line is not considered when calculating the penalty.

Input

The first line of the input file contains p — the text width ($6 \le p \le 80$). The rest of the file contains the paragraph text — a sequence of words separated by spaces and line feeds. All words consist of 1 to $\lfloor (p-1)/2 \rfloor$ characters with ASCII codes from 33 to 255. The total length of the text does not exceed 5000 characters. The number of words does not exceed 100. The number of short words does not exceed 50 (the word is called short if it contains at most three letters).

Output

Output one real number — the minimal penalty of the layed out text. You answer must be accurate up to 10^{-4}.

Example

formatting.in	formatting.out
28 From thousands of teams taking part in regional contests all around the world seventy eight teams will advance to the 2006 ACM International Collegiate Programming Contest World Finals that will take place in April, in San-Antonio, Texas.	1.75

Problem D. Young Hackers

Input file:	`hackers.in`
Output file:	`hackers.out`
Time limit:	1 second
Memory limit:	64 megabytes

Young hackers Andrew and Beth are trying to find some secret information in a long log of the chat conversation of their friend Chris. They know that the secret part of the conversation is related to the message to his friend Dan they intercepted before. But the log is too long to search by hand.

Therefore they decided to find the parts that are most similar to the message to analyze them later. They ask you to help them to write the corresponding program.

Let us represent the log as one string t with all spaces, line feeds, and punctuation marks deleted. We will also ignore the case of the letters. Thus, the string consists of capital letters of the English alphabet and digits. The message in turn is also converted to the same form, let us denote the corresponding string as p.

For each position i in t Andrew and Beth want to find the length of the maximal substring of p that starts in t at i-th position. Thus, for each i you have to find such $j = j(i)$ that $t[i \dots i+j-1] = p[k \dots k+j-1]$ for some k, and j is maximal possible.

Input

The input file contains two lines. The first line contains t, the second line contains p. The length of t does not exceed 100 000, the length of p does not exceed 5 000. Both strings are not empty.

Output

Let the length of t be n. Output n numbers — for each i output the corresponding $j(i)$.

Example

hackers.in	hackers.out
ABBAABABC ABAAB	2 1 4 3 3 2 2 1 0

Problem E. Mazes Exit Guide

Input file:	`mazes.in`
Output file:	`mazes.out`
Time limit:	1 second
Memory limit:	64 megabytes

Recently the famous national park of Flatland has opened a new attraction — two mazes located inside the famous Great Cave.

Each maze consists of several rooms connected by twisting passages. Initially the visitor of the maze enters the first room, and after that he walks along the passages, until he finally gets (or does not get...) to the last room where the exit is located.

To help visitors to walk inside the maze, some entrances to the passages are marked with various colors. Namely, for each passage exactly one of its ends is marked with some color, and for each room and each color there is at most one passage leading from this room, marked with this color.

It turned out that to get out of the maze it is enough to know the sequence of the colors of passages to choose in each room on your way. The guides of the national park have learned some such sequences, and now use them to show the maze to visitors. But since there are two mazes, they sometimes forget which sequence is required for which maze, and use the wrong one. After that they often get lost, and only the rescue team helps them to get out of the maze.

The manager of the park is not happy to see the bills from the rescue team, so he decided to find the sequence of colors that would lead to the exit in both mazes. He found the maps of the mazes in his desk, and started to build the sequence. But he soon got tired and called you. You have to find the sequence of colors that would lead to the exit in both mazes, or find out that no such sequence exists.

More of that, your manager is a minimalist, therefore he asks you to find the lexicographically smallest of such sequences. If there is no lexicographically smallest sequence you must report so.

Let us denote colors by integer numbers from 1 to n. The sequence $(a_1, a_2, \ldots, a_l)$ is lexicographically smaller than the sequence $(b_1, b_2, \ldots b_m)$ if there exists $i < m$ such that $a_1 = b_1$, $a_2 = b_2$, ..., $a_i = b_i$, and either $a_{i+1} < b_{i+1}$, or $i = l$.

Input

The first line of the input file contains n — the number of colors used to mark the passages ($1 \le n \le 100$). The rest of the input file describes the two mazes.

Each description starts with the line containing r and p — the number of rooms and passages in the maze, respectively ($2 \le r \le 100$, $1 \le p \le 10000$). The following p lines describe passages. Each passage is described with three integer numbers: a, b, and c. The triple a, b, c means that there is a passage that connects rooms a and b, and its entrance in the room a is marked with the color c.

Entrance to the maze is located in the room 1, exit from the maze is located in the room r. There can be several passages from one room to another, the passage can connect a room to itself.

Output

If there is no sequence of colors that leads to exit in both mazes, print "`no common solution`" on the first line of the output file. If a common solution exits, but there is no lexicographically smallest one, print "`no smallest solution`" on the first line of the output file.

In the other case output the solution. Print l — the length of the solution on the first line of the output file. On the second line print l integer numbers — the sequence of the colors to use to exit from both mazes.

Example

mazes.in	mazes.out
3 3 3 1 2 1 1 3 3 2 3 2 2 2 1 2 1 2 2 2	2 1 2
3 3 3 1 2 1 1 3 3 2 3 2 2 2 1 2 1 2 2 3	no common solution

Problem F. Nonequal Parts

Input file:	`nonequal.in`
Output file:	`nonequal.out`
Time limit:	1 second
Memory limit:	64 megabytes

Alice and Bob play the following game. They put a heap of n coins to the table, and after that make moves in turn. Alice makes her move first.

Each move the player takes any heap of coins from the table, and splits it into several unequal parts. For example, a heap of 7 coins can be split in the following ways:

$$7 \to 6, 1$$
$$7 \to 5, 2$$
$$7 \to 4, 3$$
$$7 \to 4, 2, 1$$

The resulting heaps are put back to the table and the game continues. The player who has no valid moves loses.

Help Alice to find out whether she can win regardless of Bob's moves, and if she can what is the first move she must make.

For example, if $n = 7$, Alice can win by splitting the initial heap into heaps containing 4, 2 and 1 coins. Bob is forced to split the heap of 4 coins into two, containing 3 and 1 coins respectively, and Alice wins by splitting 3 to 2 and 1. There are now two heaps with 2 coins and 3 heaps with 1 coin, and none of them can be split further. So Bob loses.

Input

Input file contains one number n ($3 \le n \le 300$).

Output

If Alice can win regardless of Bob's moves, print "`win`" on the first line of the output file. On the second line print the sizes of heaps she should split the initial heap to in order to win.

If Bob can force Alice to lose, print "`lose`" on the first line of the output file.

Example

nonequal.in	nonequal.out
7	win 1 2 4
8	lose

Problem G. Primitive Product

Input file:	`product.in`
Output file:	`product.out`
Time limit:	1 second
Memory limit:	64 megabytes

The number $x \in \mathbb{Z}$ is called a *unit* if there exists $y \in \mathbb{Z}$ such that $xy = 1$. The number $x \in \mathbb{Z}$ is called *irreducible* if whenever $x = yz$, either y, or z is a unit.

Consider $n \in \mathbb{Z}$. We call its representation as a product

$$n = a_1 \cdot a_2 \cdot \ldots \cdot a_k$$

primitive if each a_i is irreducible and is not unit.

Given n, you have to find all ways to represent it as a primitive product. Two representations that only differ by the order of the factors are considered the same. The factors in each product must be sorted in the nondecreasing order. The representations must be sorted by the first factor, then by the second factor, and so on.

Input

Input file contains n ($2 \le |n| \le 2 \cdot 10^9$).

Output

On the first line of the output file print t — the number of representations of n as a primitive product. The following t lines must contain representations, one on a line. Adhere to the format of sample output.

Example

product.in	product.out
30	4 -5 -3 2 -5 -2 3 -3 -2 5 2 3 5

Problem H. Rent A Car

Input file:	`rentacar.in`
Output file:	`rentacar.out`
Time limit:	1 second
Memory limit:	64 megabytes

Bill owns a number of car rent sites around the country. People are allowed to rent a car at any site and return it to any site, not necessarily the same.

To ensure that there is always enough cars for rent at any site, throughout the week Bill's employees drive extra cars from one site to another. They do it in such a way, that there is always the same number of cars at any site in the beginning of each week. So each week Bill has to decide which car should be driven to which site.

For simplicity, let us assume that all people return the car to some site the same week they have rent it. You are given the number of cars rented and returned at each site, and the information about roads between the sites. Find the shortest total path the cars must pass in the end of the week, to ensure that the number of cars at each site is the same as in the beginning of the week.

Input

The first line of the input file contains n and m — the number of cities and the number of roads in the country, respectively ($1 \le n \le 100$, $0 \le m \le 5000$). Bill has a car rent site in each city.

The following n lines decribe the sites. Each site is described by two numbers: t_i and r_i — the number of cars rented and returned at the site during the last week, respectively. The total number of rented cars is equal to the total number of returned cars, it does not exceed 10^4.

The next m lines describe roads. All roads are bidirectional. Each road is specified by the numbers of the cities it connects and its length. The lengths are positive integer numbers and do not exceed 10^3. No road connects a city to itself. There can be several roads between a pair of cities.

Output

Output one number — the shortest total path that the cars must pass in order to ensure that the number of cars at each site is the same as in the beginning of the week. If it is impossible to rearrange the cars in the required way, output -1.

Example

rentacar.in	rentacar.out
3 3 1 10 5 0 5 1 1 2 10 2 3 4 1 3 3	47

Problem I. Roundtrip

Input file:	`roundtrip.in`
Output file:	`roundtrip.out`
Time limit:	1 second
Memory limit:	64 megabytes

Once upon a time there lived the King, and he ruled the Kingdom. The King was wise and people in his Kingdom were happy. There were n towns in the Kingdom, some of them were connected by the roads. There were m roads in the Kingdom, and people were allowed to travel along them in both directions.

The King was so wise that he understood that making one of the towns a capital would be too dangerous. Should the enemy suddenly capture it, the Kingdom would be in serious danger. So, besides the capital there was another town that the King could rule the Kingdom from. That town was called a secret capital. In order to ensure that the King could safely get from the captital to the secret capital, the roads were designed in such a way that there were at least three different paths that the king could use to get from the capital to the secret capital, and no other town belonged to more than one of these paths.

Recently the King has decided to make a roundtrip around his Kingdom. He would like to start his trip in the capital, and visit two other towns — the secret capital, and the new town he has just built. In the end of the trip the King would like to return to the capital. Due to the security reasons the King wants to visit any town at most once in his trip.

Help the wizards of the King to prepare the plan of the roundtrip.

Input

The first line of the input file contains n and m ($4 \le n \le 10\,000$, $5 \le m \le 50\,000$). Let the towns be numbered from 1 to n. The second line contains three different integer numbers — a, b and c — the numbers of the capital, the secret capital, and the new town, respectively.

The following m lines describe the roads of the Kingdom — each road is specified with the numbers of the cities it connects. There is at most one road between each pair of cities, no road connects a city to itself.

Output

If there is no roundtrip satisfying all King's requirements, print -1 on the first line of the output file.

If there is such roundtrip, describe any one. On the first line of the output file print k — the number of roads in the trip. On the second line print $k+1$ number — the cities visited by the King in the roundtrip in the order he visits them. The trip must start and end in the capital.

Example

roundtrip.in	roundtrip.out
8 10 1 6 8 1 2 1 3 1 7 1 8 2 4 3 5 4 6 5 6 6 7 7 8	6 1 8 7 6 5 3 1

Problem J. Yet Another Minimal Triangle

Input file:	`triangle.in`
Output file:	`triangle.out`
Time limit:	2 seconds
Memory limit:	64 megabytes

Given n points on the plane, your task is to find the triangle of minimal area with vertices in the given points. The triangle must not be degenerate.

Input

The first line of the input file contains n — the number of points ($3 \le n \le 1000$). The following n lines contain two integer numbers each — the coordinates of the points. Points do not coincide. The absolute values of the coordinates do not exceed 10^4. There are three points that do not lie on the same line.

Output

Output three integer numbers — the numbers of the points to choose as the vertices of the triangle so that its area were minimal possible. If there are several solutions, output any one. The points are numbered in the order they are given in the input file, starting from 1.

Example

triangle.in	triangle.out
4 0 0 0 3 3 0 1 1	1 2 4