Problem A. Artistic Swimming

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Tomi is participating in a Dog's artistic swimming contest. This is a very special competition, because dogs know how to swim very well. They have to move between different designated points in a pool (they must move between pairs of designated points, not to or from other place), in the minimum amount of time. However, there is a twist. The jury will ask every participant to wait for x seconds on each designated point in the pool before they can move to the next one, and also wait x seconds in the final designated point to finish the race.

Tomi and his coach Diego know Tomi's abilities really well, they know exactly how many seconds it takes for Tomi to swim from one designated point to another (Tomi might not be able to swim directly from one designated point to another). Tomi and Diego are resting today, so they need your help to simulate a few scenarios and determine how many seconds does it take for Tomi to go from a given designated point to another, while waiting x seconds on each designated point. The referee (Osman) is unpredictable, and is uncertain what value of x he will use in the competition.

Input

The first line will contain three integers n, m, q ($2 \leq n \leq 500$, $1 \leq m \leq 2n$ and $1 \leq q \leq 10^5$) which represent the number of designated points, the number of Tomi's times Diego knows and the number of queries, respectively. Next, there will be m lines with three integers each: u, v and w ($1 \leq u, v \leq n$ and $1 \leq w \leq 500$), which represents that Tomi swims from designated point u to designated point v in w seconds. You are guaranteed that every ordered pair (u, v) will be different. Finally, there will be q lines, each line will have 3 integers u, v and x, which represent a query to find the time to go from designated point u to designated point v waiting x ($0 \leq x \leq 50000$) seconds on each node.

Output

For each query, print in a single line the minimum number of seconds it will take Tomi to go from u to v. If there is no way to go from u to v, print -1.

Examples

standard input	standard output
4 6 3 1 4 1 1 2 8 4 3 9 2 3 1 4 2 2 3 1 4 1 3 0 1 3 10 2 1 1	4 39 8
2 1 3 1 2 100 1 1 100 2 2 100 1 2 100	100 100 300

Problem B. Blockchain

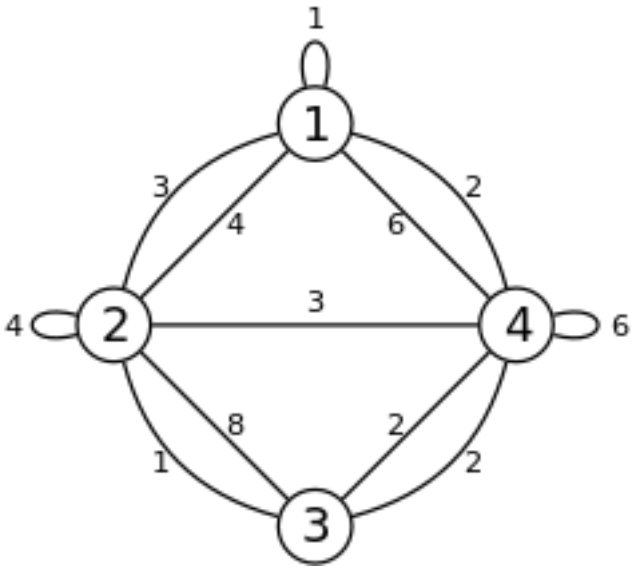
Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	32 megabytes

Proof-of-work (PoW in short) systems are used in blockchains to add valid blocks into a distributed ledger (the block chain). A block is assumed to be valid if a miner (the first, out of multiple miners) solves a difficult cryptographic puzzle, hence "proving" they have done a significant amount of computation (proof of work) before a new block is added.

Elon had a dream about a new cryptographic puzzle the other day and has decided to set up a new blockchain system for a new social network, Dogebook. On Dogebook, each block added to the blockchain consists of an undirected graph representing users, and edges representing how much they like each other (possibly multiple edges between the same pair of nodes). The puzzle Elon dreamt of is as follows: given a graph G composed of edges E and a function $w(e)$ which represents the weight of an edge $e \in E$, calculate the following function:

$$H(E, w) = \max_{E' \subseteq E} \prod_{e \in E'} w(e) \times p(w(e))$$

where $p(x) = 1$ if $x = 2k + 1$ for some non negative integer k, otherwise $p(x) = 0$.



$H(E, w) == (3 \times 1) \times (3 \times 1) \times (1 \times 1) = 9$

In the example above, one possible E' that maximizes $\prod_{e \in E'} w(e) \times p(w(e))$ consist of the subset with the edge of weight 3 between 1 and 2, the edge of weight 3 between 2 and 4 and the edge of weight 1 between 1 and 1.

Elon is convinced this system is foolproof and will make a lot of money. Tomi, a very resourceful dog-hacker, disagrees and thinks it is a waste of time. Tomi is convinced he can hack this system by solving the puzzle very fast, rendering the PoW meaningless. However, Tomi is busy running around with Eli and Rafa (see problem H), can you help Tomi to break the puzzle?

Input

The input will contain several graphs. Each graphs begins with n and m ($1 \leq n, m \leq 10^5$) separated by a whitespace, representing the number of nodes and the number of edges, respectively. Next, there will be m lines with three integers w, u and v ($1 \leq u, v \leq 10^5$, $1 \leq w \leq 10^9$), representing an edge between u and v with weight w.

Output

For each graph, you must print a line with a single integer, the value of $H(G, w)$. It is guaranteed that each answer fits on a 32-bit signed integer. Print an end of line character after each answer.

Example

standard input	standard output
4 12 1 1 1 4 2 2 6 4 4 4 2 1 6 1 4 8 2 3 2 4 3 3 2 4 3 1 2 2 1 4 2 3 4 1 2 3	9

Problem C. Corona

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 megabytes

Özlem is researching variants of the covid-19 virus. She has figured out a way to measure how transmissible a variant is by analyzing its genome sequence. In general, any genome sequence is highly transmissible if it has a large prefix which is also its own suffix. For example: for GACTCGA, GA is the longest prefix which is also a suffix (note that the full string is not a valid prefix/suffix). They call the *level* of a sequence the length of the longest prefix which is also a suffix, so the level of GACTCGA is 2. Covid-19 is a very special virus, so its transmissibility is equivalent to the sum of all substrings' level.

Özlem has asked you to compute the transmissibility for several genome sequences of variants collected all around the world. There is a lot of variants out there, so you need to develop a solution that is very fast so that Özlem and you can analyze variants in real-time.

Input

There will be several genome sequences (at most 50), one on each line. Each sequence is a string consisting of lower-case letters only. The length of each sequence is at most 5000.

Output

For each sequence, print its transmissibility on a different line.

Example

standard input	standard output
agcagct tomi acmicpc universidadnacional yvanehtnioj	10 0 3 12 1

Problem D. Derivative of polynomial

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Note: This problem is rendered better in HTML than in PDF form.

As the problem title suggests, you have to write a program to find the derivative of a polynomial, a polynomial is a string of characters that matches the following recursive definition:

- A *polynomial* is any of:
  - "0"
  - "*term*"
  - "$-$*term*"
  - "*polynomial* $+$ *term*"
  - "*polynomial* $-$ *term*"
- A *term* is any of:
  - "*positiveInt*"
  - "*coefficient x exponent*"
- A *coefficient* is any of:
  - "*nonOnePositiveInt*"
  - "" (empty string)
- An *exponent* is any of
  - "^*nonOnePositiveInt*"
  - "^$-$*positiveInt*"
  - "" (empty string)
- A *nonOnePositiveInt* is an integer number greater 1, with no leading zeroes.
- A *positiveInt* is an integer number greater than 0, with no leading zeroes.

The following are valid *nonOnePositiveInt* strings: "2", "10", "94" and "100". But these are invalid *nonOnePositiveInt* strings: "$-$3", "+32", "0" and "1".

The following are valid *positiveInt* strings: "1", "10", "94" and "100". But these are invalid *positiveInt* strings: "$-$1", "3.14", "0" and "+32".

The following are valid *coefficient* strings: "", "3" and "10". But these are invalid *coefficient* strings: "$-$10", "1" and "0".

The following are valid *exponent* strings: "", "^3" and "^$-$ 10". But these are invalid *exponent* strings: "^1", "^+ 30" and "^0".

The following are valid *term* strings: "x", "x^3", "4x^$-$ 2" and "123". But these are invalid *term* strings: "0x^2", "$-$10x^+ 32".

The following are valid *polynomial* strings: "3x^3 $-$ 2x^2 + 4" and "4x $-$ 23 + 1 + 3 $-$ 2 $-$ 5x $-$ 10x^$-$ 100". But these are invalid *polynomial* strings: "+10x^2" and "3x^3 + $-$2x^4".

The only characters in a valid *polynomial* string are: digits (0123456789), plus sign(+), minus sign ($-$), a lowercase letter x (x) and caret (^, this is ASCII 94 in input and output). There are no spaces even if the statement font make it seem like there are.

Input

A single line with a valid polynomial, the polynomial will consist of at most 100000 characters and you are guaranteed that every coefficient and exponent will be at most 100 in absolute value.

Output

A single line containing a valid polynomial, the derivative of the input polynomial. Additionally, you must:

- Combine all terms that can be combined (Output "$2x\hat{}3$" instead of "$10x\hat{}3 - 8x\hat{}3$")
- Order the terms from smallest exponent to biggest exponent

The above restrictions guarantees the answer is unique.

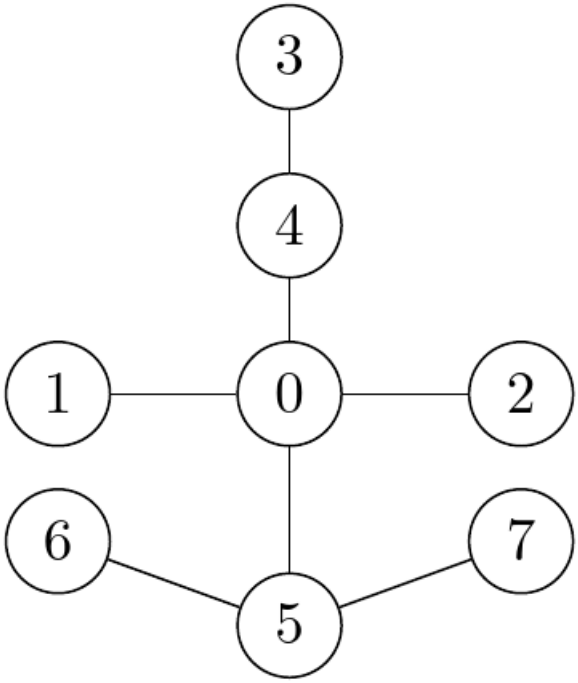
Example

standard input	standard output
4x-23+1+3-2-5x+10x^4-10x^-23+23+32x^-23	-506x^-24-1+40x^3

Problem E. Molecules

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Tomi is a famous scientific working with open-chain compounds, which generally speaking are molecules which do not form cycles between their atoms. Tomi is very interested on solving the famous protein folding problem, and for that purpose, he is developing a method to organize an open-chain compound into a list which is as balanced as possible. A single atom is balanced if exactly half of its neighbors appear to its left on the list, and the other half to the right (atoms with an odd number of neighbors will never be balanced). A list is *maximally balanced* if it contains as many balanced nodes as possible. For example:



A possible solution: $7, 6, 5, 3, 4, 0, 2, 1$

In the solution above, nodes 4 and 0 are balanced since 3 appears before 4 and 0 appears after. Similarly, 1 and 2 appear after 0, and 5 and 4 appears before. Tomi needs your help to develop an efficient algorithm to compute a maximally balanced list representation of a very big molecule.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 5 * 10^5$), the number of atoms. Next, there will be $n-1$ lines, with two integers u and v, which represent a link from atom u to atom v ($0 \leq u, v < n$).

Output

Print exactly n integers separated by space, which represent the maximally balanced list.

Example

standard input	standard output
8 0 1 0 2 0 4 0 5 4 3 5 6 5 7	3 4 5 0 1 2 6 7

Problem F. Flow of binary matrix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

A binary matrix is a matrix that has either 0 or 1 in all of its cells. The flow of a binary matrix is the number of rows contain only 1's plus the number of columns that contain only 1's, for example, here is a matrix with flow 3:

1	0	1	1
1	1	1	1
1	1	0	1
1	1	0	1

The flow is 3 because row 2 and columns 1 and 4 are full of 1s.

You are given a binary matrix of size $n \times n$, and you have to write a program to perform several operations, and for each operation your program must output the flow of the matrix after applying the operation. There are two types of operations:

- 1 i j b — updates the value in the i-th row and the j-th column to be equal to b ($1 \leq i, j \leq n$ and b is either 0 or 1).
- 2 b — inserts the value b (b is either 0 or 1) in the matrix at position $(1, 1)$, this moves all values of the first row 1 column to the right, the last element of the first row is removed from the first row and inserted as the first element of the second row, causing a cascade behavior that ends up in the last row having $n + 1$ elements. The last value of the n-th row is discarded and the matrix will still have dimensions $n \times n$.

Input

The first line of input contains two space separated integers: n and q ($2 \leq n \leq 5000$ and $1 \leq q \leq 5000$).

The next n lines describe the binary matrix. The j-th character of the i-th line is the value of the binary matrix at position (i, j).

The next q lines each contain an operation. The operations follow the format described in the statement.

Output

q lines, the i-th line must contain the flow of the binary matrix after applying the i-th operation.

Example

standard input	standard output
4 4 1011 1111 1101 1101 1 3 2 0 2 0 2 0 1 2 3 0	3 2 2 0

Note

This is the initial matrix:

1	0	1	1
1	1	1	1
1	1	0	1
1	1	0	1

After applying the first operation, we get the following matrix with flow 3:

1	0	1	1
1	1	1	1
1	0	0	1
1	1	0	1

After applying the second operation, we get the following matrix with flow 2:

0	1	0	1
1	1	1	1
1	1	0	0
1	1	1	0

After applying the third operation, we get the following matrix with flow 2:

0	0	1	0
1	1	1	1
1	1	1	0
0	1	1	1

After applying the fourth operation, we get the following matrix with flow 0:

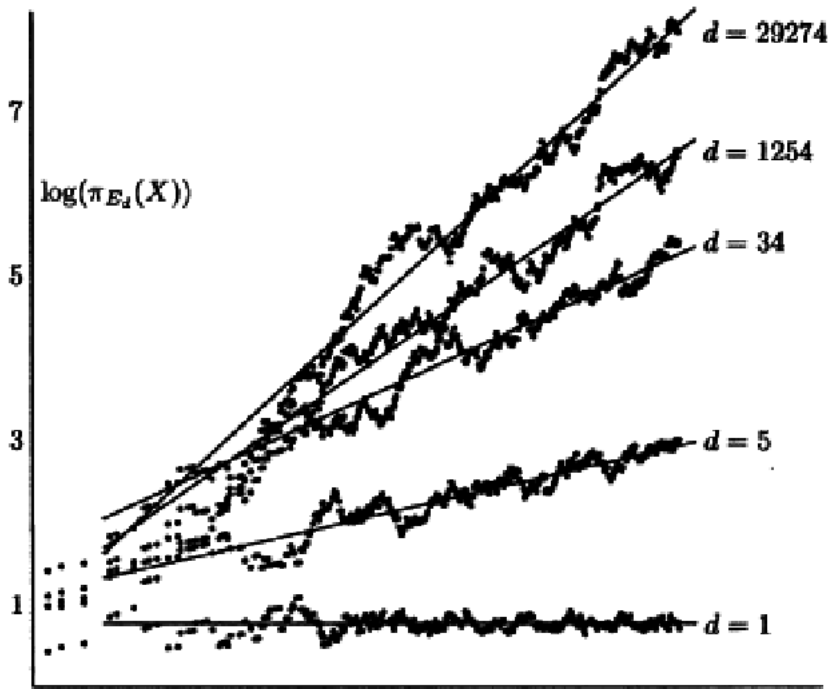
0	0	1	0
1	1	0	1
1	1	1	0
0	1	1	1

Problem G. Matematical Transformation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

When Diego tells Osman a problem about maths, Osman always tries to turn it into a graph problem. This is because it is easier for him to think about graph problems.

On this occasion Diego asked Osman about the Birch and Swinnerton-Dyer conjecture, this conjecture says that an elliptic curve (a type of cubic curve, or algebraic curve of order three, confined to a region known as a torus) has either an infinite number of rational points (solutions) or a finite number of rational points, according to whether an associated function is equal to zero or not equal to zero, respectively. In the following image we can see a related scientific diagram:



Quickly Osman turned this statement into a graph challenge. More specifically he thinks about an undirected graph rooted at node 1 in which every pair of nodes has a unique path. Each node will have a value associated, which initially is 0. And he will have two types of queries he needs to perform to the tree.

- The first query is simple: get the sum of the values in the nodes in the path between two given nodes.
- The second query is even simpler. You will be given three integers u, v and k and you have to add $v + k \times d$ to each node in the subtree of u, where d is the distance of each node to u.

Can you solve the new version of the original problem?

Input

The first line contains an integer n ($1 \leq n \leq 3 \times 10^5$) — The number of nodes in the tree.

The following $n - 1$ lines contains two integers u, v ($1 \leq u, v \leq n$) that means there is an edge between nodes u and v.

The next line will contain integer q ($1 \leq q \leq 3 \times 10^5$) — The number of queries to process.

The following q lines of input will describe the information of each query. The i-th line starts with the type of query $type_i \in \{0, 1\}$, then the rest of the line goes as follows:

- if $type_i = 0$, it will be followed by two integers u and v ($1 \leq u, v \leq n$), meaning that you want to compute the sum of the values in the nodes in the path from u to v
- if $type_i = 1$, it will be followed by three integers u, v and k ($1 \leq u \leq n$, $1 \leq v, k \leq 5 \times 10^5$), meaning that you want to add $v + k \times d$ to each node in the subtree of u, where d is the distance of each node to u.

Output

For each query of type 0 print the corresponding answer. The answer for each query should be in a separate line.

Examples

standard input	standard output
6 6 1 6 2 6 3 6 4 6 5 6 1 6 1 1 0 3 5 1 3 4 9 0 2 1 1 3 4 3 0 3 2	5 3 13
6 6 3 1 6 2 5 5 3 2 4 6 0 5 2 1 3 4 10 0 2 4 1 3 5 9 0 5 6 1 6 0 9	0 58 37
3 1 2 2 3 3 1 3 7 4 1 1 0 5 0 2 3	22
4 1 2 2 3 3 4 4 1 4 5 6 0 2 4 1 3 4 9 0 3 2	5 4

Problem H. Hiking trip

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Tomi is a playful Dog. He is on a hiking trip with his humans: Eli and Rafa.

At time 0, Eli is at position 0. She walks at a constant speed of v_0 units per minute, until she reaches position d, where she will stop. So after $t \geq 0$ minutes, Eli will be at position $min(v_0 \times t, d)$.

At time 0, Rafa is at position 1. He walks at a constant speed of $v_1 > v_0$ units per minute, until he reaches position d, where he will stop. So after $t \geq 0$ minutes, Rafa will be at position $min(1 + v_1 \times t, d)$.

At time 0, Tomi is at position 0. He will move at a constant speed of $v_2 > v_1$, until he reaches Rafa's position, then he turns 180 degrees and moves at the same speed v_2 until he reaches Eli. He continues to go back and forth between Rafa and Eli until all 3 of them are at position d, where he will stop.

Your task is to determine Tomi's position after t minutes.

Input

The input consists of a single line containing 5 space separated integers: d, $v0$, $v1$, $v2$ and t ($1 \leq d \leq 100$, $0 \leq t \leq 100$, and $1 \leq v0 < v1 < v2 \leq 100$).

Output

Output a single number, the position of Tomi after t minutes. Your will be judged as correct if it has an absolute or relative error of at most 10^{-6}.

Examples

standard input	standard output
10 1 2 3 1	3.0000000000
10 1 2 3 10	10.0000000000

Problem I. Impossible problems

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Former programming competitors at UNAL are putting together a new contest this year to define which teams will compete in the national finals. This year, they defined n problem topics and have exactly n problem setters. Each problem setter has more experience on some topics, so they have collected m data points about how much time it takes one problem setter to prepare a problem about a given topic. They have been doing this for a while, so they are very creative when they work together. Sometimes, too much. It turns out that a single problem setter can easily prepare problems about various topics, similarly, a problem topic can be prepared by many problem setters. However, when two or more problem setters coincide on a topic, and at least one of them prepares problems about some other topic, they get *too creative* and will start conflating the topics and they come up with extremely hard problems, which can not be solved within the normal time of official competitions, we can *not* allow this to happen.

Problem setter	Topic	Hours
Osman	Treaps	2
Felipe	Trees	2
Diego	Treaps	6
Osman	Math	1
Diego	Trees	3

The smallest total time to develop a contest is 8.

The former UNAL competitors want to prepare the problems spending as little time as possible, so they can go and play with Tomi. That is why they need your help to assign problem setters to topics, in such a way that all topics are covered, and all problem setters work at least with one topic. It is possible that the final problem set has more than one problem per topic. In the example above, if each problem setter chooses a different topic, the time to develop the contest would be 9, whereas if Felipe and Diego share Trees, and Osman chooses Treaps and Math, the total time to develop the contest is 8. Furthermore, if Felipe and Diego choose Trees, Osman and Diego choose Treaps, and Osman chooses Math, they will get too creative and create a problem about Trees, Treaps and Math, which will be very extremely hard.

Input

The first line contains two integers n and m ($1 \leq n \leq 200$, $1 \leq m \leq n^2$). The next m lines, will contain three integers k, t, h separated by a space, which represents that the problem setter k needs h hours to prepare a problem about topic t ($0 \leq k, t < n$, $1 \leq h \leq 500$).

Output

Print a single integer, the minimum total number of hours required to prepare the problem set. Print "Impossible" if it is not possible to develop the problem set with all the topics, with every problem setter working on at least one topic and without problem setters getting *too creative*.

Example

standard input	standard output
3 5 0 0 2 1 0 3 1 1 6 2 1 2 2 2 1	8

Problem J. Just enough squares

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Ivan has a grid paper of size $w \times h$ ($1 \leq w, h \leq 10^5$). So it contains $w \times h$ squares of size 1×1. We can use integer coordinates (x, y) to denote any corner of any of the 1×1 squares ($0 \leq y \leq w$ and $0 \leq y \leq h$).

In this grid paper, Tomi has drawn a *simple* polygon of n vertices ($3 \leq n \leq 50$). The i-th vertex has integer coordinates (x_i, y_i) ($0 \leq x_i \leq w$ and $0 \leq y_i \leq h$).

Ivan thinks Tomi's polygon is so good it can be sold to the local art museum. So he will cut the polygon out of the paper in such a way that the whole polygon will be cut out and every 1×1 square is either completely cut out or completely left in the original grid paper. Additionally, he will cut out as few 1×1 squares as possible.

Input

The first line of input contains 3 space separated integers: n, w and h.

The i-th of the next n lines contains 2 spaces separated integers: x_i and y_i. You are guaranteed that every ordered pair (x_i, y_i) will be different.

Output

Output a single integer: the number of 1×1 squares Ivan will cut.

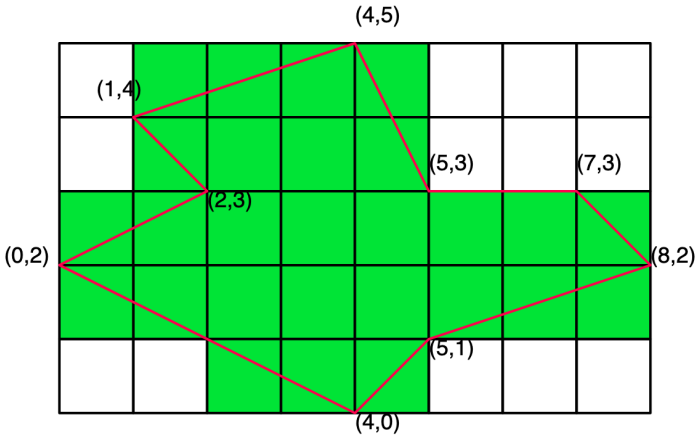
Example

standard input	standard output
9 8 5 1 4 4 5 5 3 7 3 8 2 5 1 4 0 0 2 2 3	27

Note

A *simple* polygon is a polygon that does not intersect itself and has no holes.

The green squares are cut out in the solution for the sample case:


(4,5)
(1,4)
(5,3)
(7,3)
(2,3)
(0,2)
(8,2)
(5,1)
(4,0)

Problem K. Walking Tiles

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

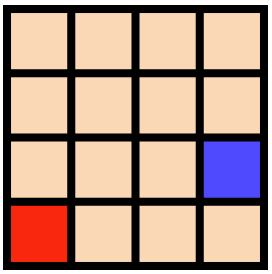
Some tiles of the UNAL entrance are just a mess. Sometimes when you are walking and you step a tile with your right foot you get your left foot dirty because of the mud under the tiles. It literally splashes out and this is awful, you know? Just when you are wearing you best suit you get dirty because of this. Osman is fed up with this. That's why he is trying to solve this problem so walking at the UNAL can be better.

He achieved to map every tile of UNAL in a 2D map. Unfortunately, he has only information about a few of them. Some of the tiles are loose and others are fixed while the others are unknown.

You can step on a fixed tile and be sure you won't get dirty and similarly you can step on a loose tile and be sure that you will get dirty. For the other tiles, you can't be sure if you can step on without getting dirty.

So in order to get a better understanding of this map Osman needs you to help with calculating distances. Suppose you start in a loose tile, then, of course, you want to go the closest fixed tile and for that, you can only move in one of the 4 directions (front, left, right, back).

For example in the below image can walk 3 steps to the right and 1 front (the blue is fixed and the red is loose while the others are unknown). So the distance is 4.



You need to compute the sum of closest distances for each loose tile to its closest fixed tile.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 2 \times 10^5$) — The number of loose and fixed tiles respectively.

The following n lines contain a pair of numbers ($-10^9 \leq x, y \leq 10^9$) which indicated the row and column of each loose tile will contain (note that rows and columns can be negative).

The following m lines contain a pair of numbers ($-10^9 \leq x, y \leq 10^9$) which indicated the row and column of each fixed tile will contain (note that rows and columns can be negative).

It's guaranteed that a tile can't be in both lists. Each tile can be either fixed, loose, or unknown.

Output

Print one integer which is the sum of closest distances for each loose tile to any closest fixed tile.

Examples

standard input	standard output
1 1 0 0 3 1	4
5 5 3 1 3 9 0 6 5 0 8 9 10 7 3 2 4 10 1 6 6 1	10
3 3 0 2 4 6 0 0 1 0 6 4 7 1	8

Problem L. Convert to heap

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

You have a tree with n vertices ($1 \leq n \leq 10^5$), numbered from 1 to n and rooted at node 1. Every node i has an integer value a_i ($0 \leq a_i \leq 10^9$).

You can update the values of the nodes q times ($1 \leq q \leq 1000$). In the i-th update, you can choose a subset of nodes and increase the value of all of them by x_i ($1 \leq x_i \leq 1000$).

You task is to use these updates to convert the tree into a heap. The tree is considered a heap if and only if every non-root node has a value less than or equal to the value of its parent. Since there can be multiple ways to do this, you must choose the one that minimizes $\sum_{i=1}^{n} a_i$.

Input

The first line of input contains two space separated integers: n and q.

The second line contains n space separated integers: a_i.

Each of the next $n-1$ lines describe the edges of the tree, each contains two space separated integers: u and v (Meaning there is an edge between u and v).

The next and final line of input contains q space separated integers: x_i.

Output

If there is a way to convert the tree into a heap, output the minimum $\sum_{i=1}^{n} a_i$ after converting it to a heap. Otherwise output -1.

Examples

standard input	standard output
5 2 40 20 20 20 50 1 2 2 3 2 4 3 5 10 20	220
5 2 40 20 20 20 51 1 2 2 3 2 4 3 5 10 20	-1

Problem M. Classroom Reordering

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Have you heard the saying that goes "when the cat's away, the mice will play"? Well, that was what happened in classroom 205 of the CyT building. They were in class and suddenly the professor had to leave the classroom for some time. At this moment the n chairs in the classroom were organized forming a circumference, each chair back to the other.

As soon as the party started, the students rearranged the chairs in some random way. After a while, somebody noticed the professor was coming back to the classroom and they were really afraid of the professor's reaction. That is why they need your help, since you have perfect memory and you remember how the chairs were arranged when the professor left. But you do not have much time, so you advise your classmates to arrange the chairs in such a way that they form a circumference and it is as similar as possible to the way the chairs are currently arranged.

More formally, there are n chairs, numbered from 1 to n. The current arrangement is given by an array a of n integers ($1 \leq a_i \leq n$). If $a_i = i$ then there is no chair in front of i-th chair, otherwise a_i is the chair in front of the i-th chair.

You have to find an arrangement b, which is an array of n *distinct* integers ($1 \leq b_i \leq n$), which is *as similar as possible* to a. Additionally, b must be a circumference. A circumference has the property that if I start at any chair, and go to the chair in front n consecutive times, I will visit each of the n chairs exactly once.

Let $l(x, y)$ be the longest common prefix of arrangements x and y, that is, the maximum k such that $x_1 = y_1, x_2 = y_2, \ldots x_k = y_k$. We consider an arrangement x more similar to a than arrangement y if $l(x, a) > l(y, a)$, or if $l(x, a) = l(y, a)$ and x is lexicographically smaller than y.

Input

The first line of input contains one integer: n ($1 \leq n \leq 5 \times 10^5$).

The second line contains n space separated integers: $a_1, a_2, \ldots, a_n$.

Output

Print a single line with n space separated numbers: $b_1, b_2, \ldots, b_n$.

Examples

standard input	standard output
7 3 4 6 5 7 1 2	3 4 6 5 7 2 1
8 3 3 3 1 1 5 7 3	3 1 4 5 6 7 8 2