Problem A. Pieces of Parentheses

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

You are teaching a class in programming, and you want to cover balanced parentheses. You've got a great visual aid, a sign with a very long, balanced string of parentheses. But, alas, somehow, your visual aid has been broken into pieces, and some pieces may be missing! You've got to try to put it back together as best you can. Given the string of parentheses on each piece, what is the longest balanced string you can form by concatenating some of them in some order? Each piece may be used at most once, and the pieces cannot be reversed.

A balanced string of parentheses is defined as:

1. The empty string
2. AB where A and B are both balanced strings of parentheses
3. (A) where A is a balanced string of parentheses

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain a single integer n ($1 \le n \le 300$), which is the number of pieces.

Each of the next n lines will hold a single string s ($1 \le |s| \le 300$), which consists only of the characters '(' and ')'. This describes one of the pieces.

Output

Output a single integer, which is the length of the longest string of balanced parentheses you can form from the pieces. Note that the empty string is technically a balanced string of parentheses, so it is always possible to form a string of length at least 0 (although the empty string is not a very effective visual aid!).

Example

standard input	standard output
3 ()) ((())()	10
5)))))) (( ))(((	2

Problem B. Stars in a Can

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 mebibytes

Vera is an astronomer studying locality of nearby star systems. The star systems can be thought of as 3D points in space. Vera would like to place a can around the stars. In other words, she would like to know what is the smallest volume cylinder that encloses the stars. The cylinder can be oriented in any direction. At least one base of the cylinder must have at least three stars.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain a single integer n ($4 \le n \le 1\,000$), representing the number of stars.

Each of the next n lines will contain three integers x, y and z ($-1\,000 \le x, y, z \le 1\,000$), representing the position of a single star. No two stars will be at the same position. No four stars will be coplanar.

Output

Output a floating point number representing the smallest volume cylinder that can enclose all the stars. Your answer must be accurate within a relative tolerance of 10^{-6}.

Example

standard input	standard output

4
1 0 0
1 1 0
0 0 0
0 0 1
	1.57079633

4
-100 0 0
10 0 10
-10 -10 -10
0 0 0
	41938.65135885

7
10 20 30
0 0 0
-100 1000 -20
100 -20 33
8 -7 900
-100 -223 -23
3 0 3
	298192571.11934924

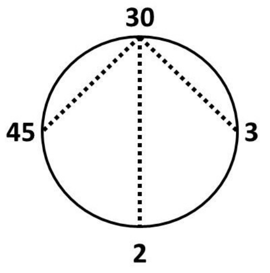
Problem C. Stretching Streamers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

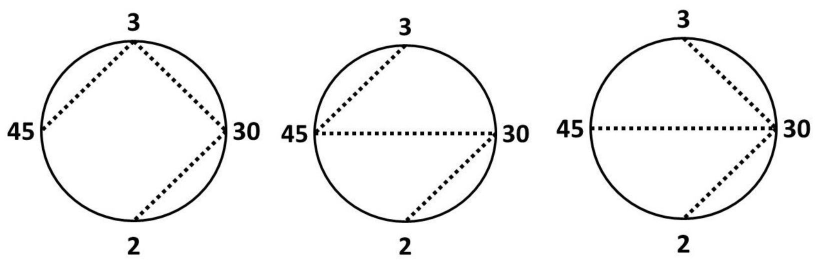
Ms. Hall wants to teach her class about common factors. She arranges her students in a circle and assigns each student an integer in the range $2..10^9$ inclusive. She also provides the students with crepe paper streamers. The students are to stretch these streamers between pairs of students, and pull them tight. But, there are some rules.

- Two students can stretch a streamer between them if and only if their assigned integers share a factor other than 1.
- There is exactly one path, going from streamer to streamer, between any two students in the circle.
- No streamers may cross.
- Any given student may hold an end of arbitrarily many streamers.

Suppose Ms. Hall has four students, and she gives them the numbers 2, 3, 30 and 45. In this arrangement, there is one way to stretch the streamers:



In this arrangement, there are three ways to stretch the streamers:



In how many ways can the students hold the streamers subject to Ms. Hall's rules? Two ways are different if and only if there is a streamer between two given students one way, but none between those two students the other way.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain a single integer n ($2 \le n \le 300$), which is the number of Ms. Hall's students.

Each of the next n lines will hold an integer x ($2 \le x \le 10^9$). These are the numbers held by the students, in order. Remember, the students are standing in a circle, so the last student is adjacent to the first student.

Output

Output a single integer, which is the number of ways Ms. Hall's students can satisfy her rules. Since this number may be very large, output it modulo $10^9 + 7$.

Example

standard input	standard output
4 30 3 2 45	1
4 3 30 2 45	3

Problem D. Heaps from Trees

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a rooted tree with n nodes. The nodes are labeled $1..n$, and node 1 is the root. Each node has a value v_i.

You would like to turn this tree into a heap. That is, you would like to choose the largest possible subset of nodes that satisfy this Heap Property: For every node pair i,j in the subset, if node i is an ancestor of node j in the tree, then $v_i > v_j$. Note that equality is not allowed.

Figure out the maximum number of nodes you can choose to form such a subset. The subset does not have to form a subtree.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain a single integer n ($1 \le n \le 2 \cdot 10^5$), which is the number of nodes in the tree. The nodes are numbered $1..n$.

Each of the next n lines will describe the nodes, in order. They will each contain two integers v_i and p_i, where v_i ($0 \le v_i \le 10^9$) is the value in the node, and p_i ($0 \le p_i < i$) is the index of its parent. Every node's index will be strictly greater than its parent node's index. Only node 1, the root, will have $p_1 = 0$, since it has no parent. For all other nodes ($i = 2..n$), $1 \le p_i < i$.

Output

Output a single integer representing the number of nodes in the largest subset satisfying the Heap Property.

Example

standard input	standard output
5	1
3 0	
3 1	
3 2	
3 3	
3 4	
5	5
4 0	
3 1	
2 2	
1 3	
0 4	
6	5
3 0	
1 1	
2 1	
3 1	
4 1	
5 1	
11	7
7 0	
8 1	
5 1	
5 2	
4 2	
3 2	
6 3	
6 3	
10 4	
9 4	
11 4	

Problem E. Blazing New Trails

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	512 mebibytes

Your state has just purchased a large, unspoiled tract of land, and wishes to turn it into a nature park with hiking trails. The land has n places of interest to which guests may wish to hike, and of these, k are very special. The state wishes to connect these places with hiking trails. There are m candidate hiking trails to choose from that directly connect two places of interest with various costs. There are some constraints for choosing the trails. First, there must be exactly one way to hike from any place to any other place. Second, exactly w of the trails must directly connect a special place with a regular place. Of course, the state wishes to minimize the cost of blazing these trails.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain four integers n, m, k and w, where n ($2 \le n \le 2 \cdot 10^5$) is the number of places, m ($1 \le m \le 5 \cdot 10^5$) is the number of potential direct trails between places, k ($1 \le k < n$) is the number of special places, and w ($1 \le w \le n - 1$) is the number of special-nonspecial direct trails the state wishes to blaze. The places are numbered $1..n$.

Each of the next k lines holds a single integer s ($1 \le s \le n$) indicating the special places. These values will be unique and will be in ascending order.

Each of the next m lines will describe a potential trail that the state could blaze. Each of these lines will consist of three integers, a, b and c, where the trail would go between places a and b ($1 \le a, b \le n, a \ne b$) and would cost c ($1 \le c \le 10^5$). No two places will have more than one potential trail between them, and a trail from a to b is the same as a trail from b to a.

Output

Output a single integer, which is the minimum total cost for the state to blaze trails in their new park subject to their constraints, or -1 if it isn't possible.

Example

standard input	standard output

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

`3 1 1 1`
`2`
`1 2 2`
	`-1`

Problem F. Incremental Double Free Strings

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

A string is called **double free** if no two adjacent letters are the same.

A string is called k-**incremental** if for all values of j in the range $[1, k]$, there exists exactly one character with j occurrences, and the string's length is $1 + 2 + 3 + \ldots + (k - 1) + k$. For example, if $k = 3$, then a **3-incremental** string should have one character appear once, another twice, another three times, in any order, for a total string length of 6.

A string is both k-**incremental** and **double free** if it meets both these criteria. Now consider examining all such strings of lowercase letters for a given k in alphabetical order. Consider the following examples.

$k = 2$: **aba**, **aca**, **ada**, . . . , **aya**, **aza**, **bab**, **bcb**, **bdb**, . . . , **zxz**, **zyz**

$k = 3$: **ababac**, **ababad**, . . . , **ababay**, **ababaz**, **ababca**, . . . , **zyzyzx**

What is the n^{th} string in an alphabetized list of all k-**incremental**, **double free** strings?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. There will be exactly one line of input. It will contain two integers, k and n $(1 \le k \le 26, 1 \le n \le 10^{18})$, which is asking for the n^{th} string in the alphabetically sorted list of all k-**incremental**, **double free** strings.

Output

Output the n^{th} k-**incremental**, **double free** string in the alphabetized list. If no such string exists, output -1.

Example

standard input	standard output
2 650	zyz
2 651	-1
5 12345678901234	yuzczuyuyuzuyci

Problem G. Apple Market

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

You are managing a market with some stores. The stores are arranged in an $n \times m$ grid. Each store sells apples. Apples cost exactly 1 Malaysian Ringgit per apple at every store.

There will be several customers who walk through this market. Each customer will only visit stores within a subrectangle of the market, and each customer has a fixed amount of money to spend. Also, each store has a limited inventory of apples, which will not be replenished between customers; the number available differs from store to store. Assuming you can control how many apples each store sells to each customer, what is the most money you can make?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain three space-separated integers n, m, and k, where the market has n rows and m columns ($1 \le n, m \le 50$), and there will be k ($1 \le k \le 10^5$) customers.

Each of the next n lines will have m integers a ($0 \le a \le 10^9$). This is a matrix in row major order, indicating the number of apples in the inventory of each store. $a[r, c]$ is the number of apples in the store in the r^{th} row, c^{th} column. The rows range from $1..n$ and the columns from $1..m$. The top left corner is $a[1, 1]$, and the bottom right corner is $a[n, m]$.

Each of the next k lines will describe a customer, with five integers: t, b ($1 \le t \le b \le n$), l, r ($1 \le l \le r \le m$), and x ($0 \le x \le 10^9$). The customer will only shop in the subrectangle from (t, l) to (b, r) inclusive (t=top, b=bottom, l=left, r=right). The customer has exactly x Malaysian Ringgits to spend.

Output

Output a single integer, representing the maximum amount of money to be made by controlling how many items each store sells to each customer.

Example

standard input	standard output

2 3 2
1 2 3
4 5 6
1 2 2 3 20
2 2 1 3 15
	20

Problem H. Maximum Color Clique

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

You found a complete, undirected graph with n nodes, labeled $1..n$. Each edge has a color. For simplicity, each color is identified by a number between 1 and 300 inclusive. Interestingly, you noticed that for each and every simple cycle in this graph, there are two adjacent edges on this cycle which have the same color.

For each non-empty subset of nodes in graph S, let $f(S)$ denote the size of the maximum subset of nodes you can choose from S such that all edges between the chosen nodes are the same color. Compute the sum of $f(S)$ over all non empty subsets S of nodes in the graph.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain a single integer n ($1 \leq n \leq 300$), which is the number of nodes in the graph.

The next n lines will each contain n integers c ($0 \leq c \leq 300$), which is a matrix representing the colors of the edges, where $c[x, y]$ is the color of the edge between node x and node y. It is guaranteed that the values on the diagonal will be 0 ($c[x, x] = 0$), since there is no edge from a node to itself. It is also guaranteed that the matrix is symmetric and the off-diagonal colors range from 1 to 300 ($1 \leq c[x, y] = c[y, x] \leq 300$ for $x \neq y$).

Output

Output a single integer, which is the sum of $f(S)$ over all non empty subsets S of nodes in the graph. Since this number may be very large, output it modulo $10^9 + 7$.

Example

standard input	standard output

4
0 1 1 1
1 0 2 2
1 2 0 3
1 2 3 0
	26

5
0 300 300 300 300
300 0 300 300 300
300 300 0 300 300
300 300 300 0 300
300 300 300 300 0
	80

Problem I. Ski Resort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 mebibytes

As an enterprising owner of a world-renowned ski resort in the US, you would like to increase your sales by stocking snack stands at key locations throughout your estate.

The ski resort is built on a mountain. A ski lift can take a guest to the top of the mountain. From there they can ski to a number of locations throughout the mountain.

There are n areas on the mountain. The areas are labeled $1..n$. The top of the mountain is area 1. Areas are connected with ski runs that are strictly downhill. In other words, it is not possible to return to an area after leaving it without taking the ski lift. Every area (including area 1) has exactly one snack stand.

As the owner of the resort, you want to know how effectively you can distribute snacks among snack stands to better serve your guests (and make more money). To this end you would like to run a survey, and analyze the result with a number of independent queries. Each guest in the survey has a favorite snack, and a list of favorite areas that they like to visit. You would like to know how to best stock your snack stands with their favorite snack.

Each query is a set of a guest's favorite areas, and a number k. You would like to know how many ways you can distribute this guest's favorite snack to exactly k snack stands on the mountain such that the snack stands meet a few conditions:

1. For each of this guest's favorite areas, over all sequences of ski runs to reach that area from the top of the mountain, there must be exactly one snack stand with the guest's favorite snack (In other words, they must not have a choice of more than one snack stand where their snack is available.)

2. Each of the k snack stands stocked with this guest's favorite snack must be on some sequence of ski runs from the top of the mountain to some area in the query set.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain three space-separated integers n, m, and q, where n ($1 \le n \le 10^5$) is the number of areas on the mountain, m ($1 \le m \le n + 50$) is the number of runs, and q ($1 \le q \le 10^5$) is the number of queries.

The next m lines each contain two integers x and y ($1 \le x, y \le n, x \neq y$). This represents a ski run from area x to area y. There will be at most one run between any two areas. It will be possible to reach each area from area 1 by some chain of ski runs.

The next q lines are each a sequence of space-separated integers, starting with k and a, which are followed by a integers i. Here, k ($1 \le k \le 4$) represents the number of snack stands to stock with this guest's favorite snack, a ($1 \le a \le n$) represents the number of areas in the query set, and the a integers i ($1 \le i \le n$) are the labels of the areas in the query set. In any given query, no integer i will be repeated.

The sum of all a's for all queries will not exceed 100 000.

Output

Output q integers, each on its own line with no blank lines in between. These represent the number of ways to select snack stands to stock for each query, in the order that they appear in the input. Two ways are considered different if an area is selected in one configuration but not the other.

Example

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

Problem J. Yin and Yang Stones

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

A mysterious circular arrangement of black stones and white stones has appeared. Ming has been tasked with balancing the stones so that only one black and one white stone remain.

Ming has two operations for balancing the stones:

1. Take some consecutive sequence of stones where there is exactly one more black stone than a white stone and replace the stones with a single black stone
2. Take some consecutive sequence of stones where there is exactly one more white stone than black stone and replace the stones with a single white stone

Given a circular arrangement, determine if it is possible for Ming to balance the stones.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The input will consist of a single string s ($1 \leq |s| \leq 10^5$), with only the characters capital 'B' and 'W'. The stones are arranged in a circle, so the first stone and the last stone are adjacent.

Output

Output 1 if it is possible for Ming to balance the stones with his rules. Otherwise, output 0.

Example

standard input	standard output
WWBWBB	1
WWWWBBW	0
WBBBBBWWBW	0

Problem K. Unbalanced Parentheses

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

Barry and Bruce are twin brothers. Bruce likes keeping his parenthetical sequences balanced. Barry would like to mess with Bruce by performing some operations on the sequence. Each operation is one of the following:

1. Change a single '`(`' to a '`)`' in the sequence.
2. Change a single '`)`' to a '`(`' in the sequence.

Bruce will attempt to rebalance the parenthetical sequence by performing the same operations. Bruce does not like tedium and will perform no more than k operations to balance the sequence.

A balanced parenthetical sequence is defined as:

1. The empty string
2. AB where A and B are both balanced parenthetical sequences
3. (A) where A is a balanced parenthetical sequence

Barry would like to disrupt the sequence to the point where it is impossible for Bruce to rebalance the sequence in k moves. Changing some position in the sequence requires effort and the amount of effort varies by position. Some positions are even delightful to switch and require negative effort. Each position can be changed at most once.

Barry hates effort and would like to compute the minimum sum of effort to ensure that Bruce cannot balance the sequence.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. The first line of input will contain two integers n and k, where n $(1 \le n \le 10^5)$ is the length of the sequence, and k $(0 \le k \le n)$ is the maximum number of moves for Bruce.

The next line contains a single string of length n consisting of only the characters '(' and ')'. This string is NOT required to be balanced.

The next n lines will each contain a single integer c $(-1\,000 \le c \le 1\,000)$, which is the cost of changing each parenthesis in order.

Output

Output a single integer, which is the minimum sum of effort of moves required to make the string impossible to be balanced by Bruce. If Bruce can always rebalance the string regardless of Barry's actions, print a single question mark ('?').

Example

standard input	standard output
4 1 ((() 480 617 -570 928	480
4 3)()(-532 870 617 905	?