Problem A. Fastest Food

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

The campus of the Eastern University can be viewed as a graph with N vertices and M bidirectional edges (vertices are numbered from 0 to $N-1$). Each edge has the same length 1. Every day, there are K students walking to the dining-hall (vertex $N-1$) from the learning building (vertex 0) at lunch time. They all want to reach the dining-hall as soon as possible. However, each edge can only serve at most c_i students at any time. Can you make arrangements for students, so that the last student can reach the dining-hall as soon as possible? (It is assumed that the speed of the students is 1 edge per unit time.)

Input

There are several (about 350) test cases, please process till EOF.

The first line of each test case contains three integers: N ($2 \le N \le 2500$), M ($0 \le M \le 5000$), K ($0 \le K \le 10^9$). Then follow M lines, each line has three integers a_i, b_i, c_i ($0 \le c_i \le 20$), means there is an edge between vertices a_i and b_i with capacity c_i.

Output

For each test case, print an integer representing the minimum time. If the requirements can not be met, print "`No solution`" (without quotes) instead.

Example

standard input	standard output
5 6 4 0 1 2 0 3 1 1 2 1 2 3 1 1 4 1 3 4 2 3 3 10 0 1 1 1 2 1 0 2 1 2 0 1	3 6 No solution

Problem B. Hard Life

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

John is a Chief Executive Officer at a privately owned medium size company. The owner of the company has decided to make his son Scott a manager in the company. John fears that the owner will ultimately give CEO position to Scott if he does well on his new manager position, so he decided to make Scott's life as hard as possible by carefully selecting the team he is going to manage in the company. John knows which pairs of his people work poorly in the same team. John introduced a hardness factor of a team — it is a number of pairs of people from this team who work poorly in the same team divided by the total number of people in the team. The larger is the hardness factor, the harder is this team to manage. John wants to find a group of people in the company that are harderst to manage and make it Scotts team. Please, help him.

Input

The first line of the input file contains two integer numbers n and m ($1 \le n \le 100$, $0 \le m \le 1000$). Here n is a total number of people in the company (people are numbered from 1 to n), and m is the number of pairs of people who work poorly in the same team. Next m lines describe those pairs with two integer numbers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) on a line. The order of people in a pair is arbitrary and no pair is listed twice.

Output

Write to the output file an integer number k ($1 \le k \le n$) — the number of people in the hardest team, followed by k lines listing people from this team in ascending order. If there are multiple teams with the same hardness factor then write any one.

Examples

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

Note

In the first example, the hardest team consists of people 1, 2, 4 and 5. Among 4 of them 5 pairs work poorly in the same team, thus hardness factor is equal to 5/4. If we add person number 3 to the team then hardness factor decreases to 6/5.

In the last example any team has hardness factor of zero, and any non-empty list of people is a valid answer.

Problem C. Goat Ropes

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

A farmer has n goats. Coincidentally, he also has n fixed posts in a field where he wants the goats to graze. He wants to tie each goat to a post with a length of rope. He wants to give each goat as much leeway as possible — but, goat ropes are notorious for getting tangled, so he cannot allow any goat to be able to wander into another goat's territory. What is the maximum amount of rope he could possibly use?

Input

There will be multiple test cases in the input. Each test case will begin with an integer n ($2 \leq n \leq 50$), indicating the number of posts in the field. On each of the next n lines will be a pair of integers, x and y ($0 \leq x \leq 1000$, $0 \leq y \leq 1000$) which indicate the cartesian coordinates (in meters) of that post in the field. No two posts will be in the same position. You may assume that the field is large enough that the goats will never encounter its border. The input will end with a line with a single 0.

Output

For each test case, output a single floating point number, which indicates the maximum amount of rope that the farmer could possibly use, in meters. Output this value with exactly two decimal places, rounded. Output no spaces, and do not output a blank line between answers.

Example

standard input	standard output
2 250 250 250 750 3 250 250 500 500 250 750 0	500.0 603.55

Problem D. Flooding Fields

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

Farmer John has a problem: it's raining, and his pastures are at risk of flooding. Luckily he's invested in a state-of-the-art drain system that ensures that the water level is the same across the farm. He hasn't been as lucky with the terrain. Farmer John's cows can only stand on dry land: if the land becomes wet, any cow on that land drowns. Luckily, cows can move one grid sector each hour, giving them a chance to escape the water (the levels of which rise and fall each hour).

Farmer John has divided his field into grid sectors, which he indexes with a row and column. Each grid sector is small enough to only be able to hold one cow. Given a description of Farmer John's field, the starting locations of his cows, and the height of the water at each hour, and assuming that Farmer John's cows are extremely intelligent and prescient, and so can always make the best move, what is the maximum number of Farmer Johns cows that could survive?

Input

There will be several test cases in the input. Each test case will begin with a line with three integers, n ($1 \leq n \leq 100$), k ($0 \leq k \leq 100$), and h ($1 \leq h \leq 24$), where n represents the size of Farmer John's field (it's $n \times n$ grid sectors), k is the number of cows in the field, and h is the number of hours he needs to track.

Each of the next n lines will contain n integers each, which represent the height of each grid sector of Farmer John's field ($0 \leq height \leq 100$). The first row in the input is row 0, and the last is row $n - 1$. The first column in each row is column 0, and the last is column $n - 1$.

The following k lines will each contain a pair of integers, r followed by c ($0 \leq r, c < n$). Each line indicates the grid sector position of one cow at hour 0, where r is the row and c is the column. No two cows will be in the same position.

The next h lines will each contain a single integer, indicating the level of the flood at that hour ($0 \leq level \leq 100$). These lines are given in order: hour 1 first, then hour 2, and so on. Note that the times start at hour 1, and the cows' positions are given at hour 0, so the cows have an opportunity to move (or MOOOOve) before the first hour's flood. A grid square is considered to be flooded if its $height \leq level$ of the flood at a given hour. The input will end with a line with three 0s.

Output

For each test case, output a single integer denoting the maximum possible number of surviving cows. Do not output any spaces, and do not print any blank lines between answers.

Example

standard input	standard output
3 1 3 2 1 2 3 1 3 2 4 2 1 1 0 2 1 0 0 0	1

Problem E. Craftsman

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

Takeshi, a famous craftsman, accepts many offers from all over Japan. However, the tools which he is using now has become already too old. So he is planning to buy new tools and to replace the old ones before next use of the tools. Some offers may incur him monetary cost, if the offer requires the tools to be replaced. Thus, it is not necessarily best to accept all the orders he has received. Now, you are one of his disciples. Your task is to calculate the set of orders to be accepted, that maximizes his earning for a given list of orders and prices of tools. His earning may shift up and down due to sale income and replacement cost. He always purchases tools from his friends shop. The shop discounts prices for some pairs of items when the pair is purchased at the same time. You have to take the discount into account. The total price to pay may be not equal to the simple sum of individual prices. You may assume that all the tools at the shop are tough enough. Takeshi can complete all orders with replaced tools at this time. Thus you have to buy at most one tool for each kind of tool.

Input

The input conforms to the following format:

$N\ M\ P$

$X_1 K_1 I_{1,1} \cdots I_{1,K_1}$

$\cdots$

$X_N K_N I_{N,1} \cdots I_{N,K_N}$

Y_1

$\cdots$

Y_M

$J_{1,1} J_{1,2} D_1$

$\cdots$

$J_{P,1} J_{P,2} D_P$

where N, M, P are the numbers of orders, tools sold in the shop and pairs of discountable items, respectively. The following N lines specify the details of orders. X_i is an integer indicating the compensation for the i-th order, and K_i is the number of tools required to complete the order. The remaining part of each line describes the tools required for completing the order. Tools are specified by integers from 1 through M. The next M lines are the price list at the shop of Takeshis friend. An integer Y_i represents the price of the i-th tool. The last P lines of each test case represent the pairs of items to be discounted. When Takeshi buys the $J_{i,1}$-th and the $J_{i,2}$-th tool at the same time, he has to pay only D_i yen, instead of the sum of their individual prices. It is guaranteed that no tool appears more than once in the discount list, and that $max(Y_i, Y_j) < D_{i,j} < Y_i + Y_j$ for every discount prices, where $D_{i,j}$ is the discount price of i-th and j-th tools bought at the same time. Also it is guaranteed that $1 \leq N \leq 100, 2 \leq M \leq 100, 1 \leq Ki \leq 100, 1 \leq P \leq M/2$ and $1 \leq X_i, Y_i \leq 1000$.

Output

Output the maximum possible earning of Takeshi to the standard output.

Example

standard input	standard output
3 4 2 100 2 1 2 100 1 3 100 1 4 20 20 50 150 1 2 30 3 4 180	120
1 2 1 100 1 2 20 40 1 2 51	60

Problem F. Uniform Flow

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

You are given a graph with n vertices and m edges. You want to make the flow of water from source to sink as large as possible. Source is located at node 1, while sink is located at node n. For each edge you can pick the amount of water that will flow along this edge every second and the direction of this flow. After all values and directions are defined we can compute the value of the flow for any path. The *value* of the flow along the path is considered to be equal to the sum of values of the flow for all edges of this path. If the direction of the flow along the edge is opposite to the direction of the path, then this edge is counted with minus sign. The values and directions should be defined to satisfy the following conditions:

- For each vertex v except the source and the sink, the amount of water that comes to this vertex every second must be equal to the amount of water that flows away from this vertex every second.
- For each pair of vertices v and u, the value of any two paths connecting these vertices should be equal.
- The flow along each edge doesn't exceed the capacity of this edge.

All edges are bidirectional, each pair of vertices can be connected by more than one edge.

Input

The first line of the input contains a single integer n ($2 \le n \le 100$) — the number of vertices in the graph. The source is located at vertex 1, while the sink is located at vertex n. The second line contains a single integer m ($1 \le m \le 5000$) — the number of edges. Then follow m lines containing edges descriptions. Each edge is defined by three integers a_i, b_i and c_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $0 \le c_i \le 10\,000$) — indexes of vertices connected by this edge and its capacity function.

Output

The first line of the output should contain the maximum amount of water that can be delivered to sink every second. Then print m lines containing the amount of water that should be transported along the corresponding edge any second. If the direction should differ from the order vertices appear in the input (i.e. the water should flow from b_i to a_i), multiply this value by -1. Your answer should differ from optimal by no more than 10^{-3} by absolute error.

Example

standard input	standard output
2 1 1 2 1	1.0000000000 1.0000000000
3 2 1 2 1 2 3 2	1.0000000000 1.0000000000 1.0000000000

Problem G. Altitude

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

YT (the location of this NOI) is a very well planned city. The city is partitioned by horizontal and vertical roads into $n \times n$ regions. For simplicity, the city and each of the regions can be viewed as a square, and there are $(n + 1) \times (n + 1)$ road intersections and $2n \times (n + 1)$ bidirectional roads (which we'll denote by roads). Every road connects two neighboring intersection points. The following diagram is an example of a map when $n = 2$, the city is divided into 2×2 and contains 3×3 intersections and 12 bidirectional roads.

Little Z has tallied the number of people traveling in each direction of every road during the every day's rush hour. Every road intersection has a corresponding altitude, and climbing hills is a very tiring task. For every h units of height climbed, the energy exerted is h; but no energy is expended when traveling downhill. If the altitude of the destination of a road minus its starting point is h (note h can be negative), then the energy exerted is $max(0, h)$. Little Z also knows that the intersection at the northwest corner has height 0, and the intersection at the south east corner has height 1. He is interested in knowing in the best case scenario (all other heights can be anything), the least total amount of energy spent by all the people.

Input

The input contains one integer in the first line, n in the above description. $4n(n + 1)$ lines follow, each line contains a non-negative integer denoting the number of people traveling on some roads. The roads are given in the following order: $n(n + 1)$ numbers for each west to east road, $n(n + 1)$ numbers for each north to south road, $n(n + 1)$ roads for each east to west road and $n(n + 1)$ numbers for each south to north road. For each direction, the roads are ordered by their starting point, first in north to south order, then west to east order (see sample input for tie breaking rules).

Output

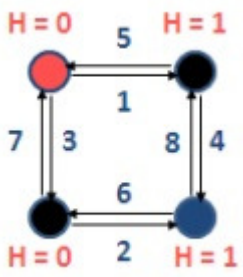
Output should contain only one integer, the total energy spent in the best case scenario rounded to an integer.

Examples

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

Note

The heights may not be integral. The picture below illustrates the sample:


H = 0
H = 1
5
1
7
3
8
4
6
H = 0
2
H = 1

Problem H. Looking for a Bride

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

Once upon a time, the king of Flatland decided to sent k of his sons to look for a bride. As you probably know, there are n cities in Flatland, and some of them are connected by roads. King lives in a capital, that has index 1, while city n is famous to have the best brides.

King ordered each of his sons to travel from city 1 to city n. Though there are many brides in city n, some of them are more beautiful than the others, so king's sons do not trust each other and would like to arrange the travelling in such a way that no two sons use the same road (even at different moment of times). King loves his sons so he would like to minimize the average length of their paths.

Input

The first line of the input contains three integers n, m and k ($2 \le n \le 200$, $1 \le m \le 2000$, $1 \le k \le 100$) — the number of cities and road in Flatland and the number of king's sons respectively.

Next m lines contain tree positive integers v_i, u_i and t_i each ($1 \le v_i, u_i \le n$, $0 \le t_i \le 10^6$) — the indexes of cities connected by the I-th road and the amount of time required to travel along this road. The road can be used in any direction and any pair of cities can be connected by more than one road.

Output

If there is no way to follow the king's order, print -1 on the single line of the output. Otherwise, first print the minimum average travelling time with at least 5 digits after decimal point. Next k lines should contain the descriptions of paths the sons should travel. Each path description should start with the number of cities along the path, followed by the number of roads in order they should be used. Roads are numbered starting with 1 in order they appear in the input.

Examples

standard input	standard output
5 8 2 1 2 1 1 3 1 1 4 3 2 5 5 2 3 1 3 5 1 3 4 1 5 4 1	3.00000 2 2 6 2 3 8

Problem I. Painting

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

You are given a rectangular painting consisting of black and white cells. To make the painting look smooth you have decided to paint gray any border that separates white and black cell.

Before the start of this task you found out that gray paint is very expensive. To save some money you can change the color of some cells (paint some white cells black, and some black cells white) and only then paint grey borders. Of course, black and white paint also costs some money.

Given the description of the initial painting and the cost of white, black and gray paint find out the minimum cost required to finish the painting.

Input

The first line of the input contains five integers n, m, w, b, g ($1 \le n, m \le 70$, $1 \le w, b, g \le 1000$) — height and width of the painting, cost of painting one cell while, painting one cell black and the cost of one gray border line, respectively. Then follow n lines each containing m characters. Character '`B`' stands for the black cell, while character '`W`' stands for the white.

Output

Print one integer — the minimum cost required to finish the painting.

Examples

standard input	standard output
3 2 10 12 1 BW WB BW	7

Problem J. Snails

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

Two snails Masha and Petya are located in the graph of size n and want to get home. Vertices of the graph are numbered from 1 to n. There are m unidirectional arcs, there may be multiple arcs between a pair of vertices and there may be an arc that connects vertex with itself. Masha and Petya dislike each other so they would never use the same arc. Help them get home.

Input

The first line of the input contains four integers n, m, a and h ($2 \le n \le 100\,000$, $0 \le m \le 100\,000$, $1 \le a, h \le n$, $a \ne h$) — the number of vertices and edges in the graph, the index of vertex snails are currently located at and the index of the vertex where the home is located.

Then follow m pairs of integers. Pair (x_i, y_i) means that there the i-th arc goes from vertex x_i to vertex y_i.

Output

If there exists a solution print "`YES`" on the first line of the output. Then print two lines containing the paths of the snails (Masha goes first as she is a lady). If there is no solution print "`NO`" on the only line of the output. If there are multiple solution, print any of them.

Examples

standard input	standard output
3 3 1 3 1 2 1 3 2 3	YES 1 3 1 2 3