Contents

In each problem the are two `Time Limit` – for language family `C/C++` and for language family `Java`. All other languages use `TL` for `C/C++`. `Time Limit` for family `Java` in two times greater than `Time Limit` for family `C/C++`. In statement you see only `Time Limit` for `C/C++`. Jury do not guarantee ability to get `OK` using any other language except `C/C++/Java`.

Warm-up

Problem A. Just a flow [2 sec, 256 mb]

Tou are given a system of nodes and pipes. Water may flow throw pipes. For each pipe you know maximal speed of the water flow. Each moment of time for each node except source and sink amount of water flowing in the node is equal to amount of water flowing out the node

You have to find maximal amount of water per unit time which can flow between source and sink. Also you have to find speed of the flow along each pipe, which should be to acquire maximal amount.

All pipes can be used in both directions. Between pair of nodes can be more than one pipe.

Input

The first line contain integer N — number of nodes ($2 \leqslant N \leqslant 100$). All nodes are numbered from 1 to N. The source has number one, sink has number N. The first line contain integer M — number of pipes ($1 \leqslant M \leqslant 5\,000$). Next M lines describe pipes. Each pipe is described by three integers A_i, B_i, C_i, where A_i, B_i — indices of nodes, which the pipe connects, and C_i ($0 \leqslant C_i \leqslant 10^4$) — maximal speed of the water flow throw the pipe.

Output

In the first line output maximal amount of water per unit of time. i-th of the next M lines should contain speed throw the pipe number i. If direction of the flow throw the pipe differs from order of nodes in input, output the speed with sign minus; Output numbers with accuracy at least 3 digits after decimal point.

Example

flow.in	flow.out

2
2
1 2 1
2 1 3
	4.0000000
	1.0000000
	-3.0000000

Problem B. Разрез [0.5 sec, 256 mb]

You are given undirected graph. Find minimal cut between vertices 1 and n.

Input

The first line contain integers n ($1 \leqslant n \leqslant 100$) — number of vertices in the graph, and m ($0 \leqslant m \leqslant 400$) — number of edges in the graph. Next m lines describe the edges. Each edge is described by numbers of its ends and its capacity (positive integer no more than 10 000 000). There are no multiedges and loops.

Output

On the first line output number of edges going throw minimal cut and its summary capacity. Next line should contain increasing sequence of indecies of edges (edges are numbered by integers from 1 to m in order they are given).

Example

cut.in	cut.out

3 3
1 2 3
1 3 5
3 2 7
	2 8
	1 2

Problem C. Decomposition of flow [2 sec, 256 mb]

You are given directed graph. Each edge has integer capacity. You have to find max floe from vertex number 1 to vertex number n and decompose it as union of paths.

Input

The first line contains numbers n ($2 \leqslant n \leqslant 500$) and m ($1 \leqslant m \leqslant 10\,000$) — number of vertices and edges in teh graph ($1 \leqslant n, m \leqslant 25\,000$). i-th of next m lines contains three integers a_i, b_i, c_i means edge from a_i to b_i with capacity c_i ($0 \leqslant c_i \leqslant 10^9$).

Output

Output number of paths (equals to the value of max flow). Each of the next lines should contain description of pathes. Each path is described as $value\ k\ e_1, e_2, \ldots e_k$ (value of path, number of edges in the path, indices of the edges of the path). Edges are numbered from 1 to m in order they are given.

Examples

decomposition.in	decomposition.out

4 5
1 2 1
1 3 2
3 2 1
2 4 2
3 4 1
	3
	1 2 1 4
	1 3 2 3 4
	1 2 2 5

Problem D. Snails [0.5 sec, 256 mb]

Two snails Masha and Petya are on the lawn with apricots and want to go back to their home. Lawns are numbered by integers from 1 to n and are connected by roads. Two lawns may be connected by many roads. It may happen, a road connects a lawn with itself. Due to reasons of hygiene, if one snail already went along the road, another one can not use this road. Help to Petya and Masha to get back home.

Input

The first line contain four integer numbers — n, m, s и t (number of lawns, number of roads, index of lawn with apricots, index of lawn with the home). Next m lines describe roads. Each line contain pair of integers (x, y), means there is road from lawn number x to lawn number y. (due to the nature of snails and the space all roads are one-way).

Limitations: $2 \leqslant n \leqslant 10^5, 0 \leqslant m \leqslant 10^5, s \neq t$.

Output

If there is a solution output `YES` and two lines with sequences of indices of lawns in the paths. At first output the path for Masha, then output the path for Petya. If there is no solution output `NO`. If there are several solutions output any of them.

Example

snails.in	snails.out

3 3 1 3
1 2
1 3
2 3
	YES
	1 3
	1 2 3

Note

You are given oriented graph, find two disjoint by edges paths from s to t. Output vertices of desired paths.

Problems

Problem E. Full orientation [3 sec, 256 mb]

You are given undirected graph without loops and multiedges. You have to make this graph directed in such a way, that maximal outgoing degree is minimal possible.

Input

The first line contains numbers n and m — number of vertices and edges in teh graph ($1 \leqslant n, m \leqslant 25\,000$). Next m lines contain pairs of integers from 1 to n — edges of the graph.

Output

Output m integers from 0 to 1. If i-th edge is given as a_i, b_i then zero means it should go from a to b, and one means edge from b to a.

Examples

orient.in	orient.out

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

5 5
1 2
2 3
3 1
1 4
1 5
	1
	0 0 0 1 1

Problem F. Perspective [1 sec, 256 mb]

Breaking news! A Russian billionaire has bought a yet undisclosed NBA team. He's planning to invest huge effort and money into making that team the best. And in fact he's been very specific about the expected result: the first place.

Being his advisor, you need to determine whether it's possible for your team to finish first in its division or not.

More formally, the NBA regular season is organized as follows: all teams play some games, in each game one team wins and one team loses. Teams are grouped into divisions, some games are between the teams in the same division, and some are between the teams in different divisions.

Given the current score and the total number of remaining games for each team of your division, and the number of remaining games between each pair of teams in your division, determine if it's possible for your team to score at least as much wins as any other team in your division.

Input

The first line of the input file contains N ($2 \leqslant N \leqslant 20$) — the number of teams in your division. They are numbered from 1 to N, your team has number 1.

The second line of the input file contains N integers w_1, w_2, …, w_N, where w_i is the total number of games that i^{th} team has won to the moment.

The third line of the input file contains N integers $r_1, r_2, \ldots, r_N$, where r_i is the total number of remaining games for the i^{th} team (including the games inside the division).

The next N lines contain N integers each. The j^{th} integer in the i^{th} line of those contains a_{ij} — the number of games remaining between teams i and j. It is always true that $a_{ij} = a_{ji}$ and $a_{ii} = 0$, for all i $\sum_j a_{ij} \leqslant r_i$.

All the numbers in the input file are non-negative and don't exceed 10 000.

Output

On the only line of output, print "YES" (without quotes) if it's possible for the team 1 to score at least as much wins as any other team of its division, and "NO" (without quotes) otherwise.

Example

perspective.in	perspective.out

3
1 2 2
1 1 1
0 0 0
0 0 0
0 0 0
	YES

3
1 2 2
1 1 1
0 0 0
0 0 1
0 1 0
	NO

Problem G. Vertex cover [1 sec, 256 mb]

You are given weighted bipartitee graph. You have to find vertex cover of minimal weight. *Vertex cover* – set of vertices which covers all edges (for each edge at least one of two edges is in the set). Weight of vertex cover is summary weight of all vertices in the set.

Input

The first line contains number of tests T.

Then T tests are given.

Each test is described by N ($1 \leqslant N \leqslant 100$) — number of vertices in the first part; N numbers – weights of vertices in the first part; M ($1 \leqslant M \leqslant 100$) — number of vertices in the second part; M numbers – weights of vertices in the second part; number of edges E ($0 \leqslant E \leqslant 10^4$); E pairs of integers u, v ($0 \leqslant u \leqslant N-1, 0 \leqslant v \leqslant M-1$).

Limitations:
N, M, E do not exceed 30 000.
All weights are integers from 0 to 10^9.

Output

For each test output integers n and m – numbers of choosen vertices in the first and the second parts. Then output n indices of vertices in the first part. Then output m indices of vertices in the first part. If there are several optimal answers, you may output any.

Examples

controls.in	controls.out

1
2
1 178
3
178 178 1
4
0 0
0 1
1 1
1 2
	2 0
	0 1

Problem H. Матан [3 sec, 256 mb]

In university of city M. experiment takes place. Tutors decide what to tell during the course of lections.

One tutor specialized on mathematical analysis (Matan) valued all known to him themes in the course. As result each theme has a number (possible negative) — "— usefulness of the topic. Tutor wants to maximize summary usefulness of the themes, but it's not so easy. To make students understand theme, you need to tell them some other themes, beacause of some proofs are based on facts from other themes. But if there is cycle of dependencies, you may read all the themes of the cycle and finally all stdudents will understand all the themes from the cycle.

You have to find set of themes. This set should be clear for students and usefulness of the set should be maximized.

Input

The first line contains integer N ($1 \leqslant N \leqslant 200$) – number of themes. The second line contains N integers, not exceeding 1 000 by absolute value — usefulness of themes. Next N lines describe dependencies between themes. i-th line start with k_i — number of themes students should already know to understand i-th theme. Then k_i diffetent integers from 1 to N – indecies of the themes;

Summary number of dependencies do not exceed 1 800.

Output

Output the only number — maximal summary usefulness of choosen themes;

Examples

matan.in	matan.out

4
-1 1 -2 2
0
1 1
2 4 2
1 1
	2

3
2 -1 -2
2 2 3
0
0
	0

Some more problems

Problem I. Maximal Matching [3 sec, 256 mb]

You are given a bipartite graph. Each vertex has some weight. The weight of an edge is the sum of weights of its ends. The weight of a matching is the sum of weights of edges of the matching. You have to find a matching with maximal weight. Note that the matching may contain any number of edges, the only constraint is that its weight should be maximal possible.

Recall that a matching in a bipartite graph is a collection of edges of the graph such that no two edges share a common vertex.

Input

The first line contains the sizes of parts of the bipartite graph n and m ($1 \leqslant n, m \leqslant 5\,000$) and the number of edges e ($0 \leqslant e \leqslant 10\,000$). The second line contains n integer numbers from 0 to 10 000 — weights of vertices of the first part. The third line contains m integer numbers from 0 to 10 000 — weights of vertices of the second part. Next e lines contain the edges of the graph. Each edge is described by a pair of integer numbers a_i b_i, where $1 \leqslant a_i \leqslant n$ is the number of vertex in the first part and $1 \leqslant b_i \leqslant m$ is the number of vertex in the second part.

Output

On the first line output w — the maximal weight of matching. On the second line output k — the number of edges in a matching with maximal weight. Next line should contain k different numbers from 1 to e — numbers of edges from the matching. If there are several matchings with maximal weight, you may output any one of them.

Examples

matching.in	matching.out

4 3 3
2 0 9 9
1 0 9
1 2
2 1
1 1
	3
	1
	3

3 2 4
1 2 3
1 2
1 1
2 1
2 2
3 2
	8
	2
	4 2

Problem J. Looking for brides [0.3 sec, 256 mb]

Once upon a time King of Flatland Widely known in Flatland there are n cities, some of them are connected by roads. King lives in capital, its number is one. City with number n is famous by its brides.

So the King ordered each of his sons should reach city n from city 1 using the roads. There are a lot of brides, but only few of them are really beutiful, so sons fear of each other. So the want to reach the destination in such a way, that any two sons don't go along the same road (even in different moments of time). King loves his songs so average time his son spends in path is as minimal as possible.

Input

The first line contains integers n, m и k — number of cities, number of roads, number of sons. ($2 \leqslant n \leqslant 200$, $1 \leqslant m \leqslant 2000$, $1 \leqslant k \leqslant 100$). Next m lines contain by three positive inregers – cities, which are connected by the road and time to go along the road. Times don't exceed 10^6. Any road can be used in both directions. Two cities may be connected by several roads.

Output

If it's impossible to choose k such paths, output the only number -1. In other case output on the first line minimal possible average time (with accuracy at least 5 digits after floating point). Next k lines should contain choosen paths for the sons. To describe path output number of roads in it and the roads in traversal order. Roads are numbered from 1 to m as they are given in the input.

Example

brides.in	brides.out

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
	3 1 5 6
	3 2 7 8