C. Connected Components

time limit per test: 4.0 s
memory limit per test: 512 MB
input: standard input
output: standard output

You are given an undirected graph consisting of n vertices and m edges. For each vertex u you have to calculate the size of the connected component containing the vertex u.

Input

The first line contains two integer numbers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — number of vertices and edges.

The following m lines contains edges: edge i is given as a pair of vertices v_i, u_i ($1 \le v_i, u_i \le n$, $u_i \ne v_i$). There is no multiple edges in the given graph, i.e. for each pair (v_i, u_i) there no other pairs (v_i, u_i) and (u_i, v_i) in the list of edges.

Output

Print n integer numbers: number i must be equal to the size of the connected component containing the vertex i.

Examples

input Copy

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

output Copy

```
2 2 2 2 1
```

input Copy

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

output Copy

```
3 3 3
```

input Copy

```
3 0
```

output Copy

```
1 1 1
```

input Copy

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

output Copy

```
3 3 3 2 2
```

D. Labyrinth analysis

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a labyrinth on a matrix $n \times m$. Each cell is either empty (denoted by a dot), or contains an obstacle (denoted by an asterisk).

A robot can only move through empty cells, and can move only to any of four cells that share an edge with current one.

Print the total number of connected components of empty cells and their sizes, in non-decreasing order.

Input

The first line contains two integers n and m ($1 \le n, m \le 1000$) — the size of the labyrinth.

n lines follow, each containing m characters, describing the labyrinth.

Output

Print the number of components in the first line. In the second line print components' sizes, in non-decreasing order.

Examples

input Copy

```
7 10
..**......
....*.....
******....
......****
.....*..**
.....*****
......**..
```

output Copy

```
5
2 2 6 15 22
```

input Copy

```
2 2
**
**
```

output Copy

```
0
```

E. Bipartite Checking

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected graph consisting of n vertices and m edges. You have to check that the given graph is bipartite, i.e. that it is possible to divide its vertices into two sets in such a way, that in each set there is no pair of adjacent vertices.

Input

The first line contains two integer numbers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — number of vertices and edges.

The following m lines contains the edges: edge i is given as a pair of the vertices v_i, u_i ($1 \le v_i, u_i \le n$, $u_i \ne v_i$). There is no multiple edges in the given graph, i.e. for each pair (v_i, u_i) there no other pairs (v_i, u_i) and (u_i, v_i) in the list of edges.

Output

If the given graph is bipartite, print "YES" (without quotes), otherwise print "NO".

Examples

input Copy

```
4 4
1 2
2 3
3 4
4 1
```

output Copy

```
YES
```

input Copy

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

output Copy

```
NO
```

J. Topological Sort

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a directed graph of n nodes and m vertices. A topological sort of a directed graph is a permutation of vertex indices such that each edge of the graph goes from a vertex that appears strictly earlier on the list to a vertex that appears later.

Check if there is a topological sort of the given graph. If there is, output any valid topological sort.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 10^5$) — the number of vertices and the number of edges.

The next m lines contain edges, one per line. Each line contains two integers a and b ($0 \leq a, b < n$) — the startpoint and the endpoint of an edge. Loops and multiple edges are allowed.

Output

If there is no topological sort for the given graph, output "NO" (without quotes).

Otherwise, output "YES" in the first line. In the second line, output a permutation of indices of the graph (integers from 0 to $n - 1$), such that for each edge the startpoint appears in the permutation strictly earlier than the endpoint.

Examples

input Copy

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

output Copy

```
YES
0 2 1 3
```

input Copy

```
3 3
0 1
1 2
2 0
```

output Copy

```
NO
```

input Copy

```
2 2
0 1
1 1
```

output Copy

```
NO
```

M. Bridges

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected graph consisting of n vertices and m edges. You have to find all edges, which are bridges. I.e. the edges, removing which leads to increasing the number of connected components.

Input

The first line contains two integer numbers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — number of vertices and edges.

The following m lines contains edges: edge i is given as a pair of vertices v_i, u_i ($1 \le v_i, u_i \le n$, $u_i \ne v_i$). There is no multiple edges in the given graph, i.e. for each pair (v_i, u_i) there no other pairs (v_i, u_i) and (u_i, v_i) in the list of edges.

Output

In the first line of output print the number of bridges. In the second line, print indices of edges which are bridges in **any** order.

Examples

input Copy

```
7 8
1 2
2 3
1 3
3 4
2 5
5 6
6 7
5 7
```

output Copy

```
2
4 5
```

input Copy

```
4 4
1 2
2 3
3 4
4 1
```

output Copy

```
0

```

input Copy

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

output Copy

```
4
1 4 3 2
```

A. Depth-First Search

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected graph consisting of n vertices.

You have to find the path from the vertex a to the vertex b using depth-first search. Use an implementation that iterates over adjacent vertices of current vertex in ascending order of indices. Only paths that are produced by this version of the algorithm will be accepted.

Input

The first line contains three integer numbers n, a, b ($1 \le n \le 500$, $1 \le a, b \le n$) — number of vertices and pair of vertices between which you have to find the path.

The following n lines contain square binary matrix g — adjacency matrix of the given graph. Each row consists of n space-separated integer numbers 0 or 1. Element $g_{i,j}$ equals to 1, if there is an edge between vertices i and j, or 0 otherwise.

It is guaranteed that there are no loops (i.e. $g_{i,i} = 0$ for each i) and that adjacency matrix is symmetric (i.e. $g_{i,j} = g_{j,i}$ for any pair i and j).

Output

If there is no path from a to b, print only one integer "-1" (without quotes).

Otherwise in the first line of the output print path length (number of edges in the path). In the second line print the path itself, beginning with the vertex a and ending with the vertex b.

Examples

input Copy

```
4 1 4
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0
```

output Copy

```
3
1 2 3 4
```

input Copy

```
4 1 4
0 1 1 0
1 0 1 0
1 1 0 0
0 0 0 0
```

output Copy

```
-1
```

B. Tree?

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Tree is a *connected* acyclic graph. You are given an undirected unweighted graph. You are to check if it is a tree or not.

Input

The first line contains a single integer N ($1 \leq N \leq 100$) — the number of vertices in the graph.

The next N lines contain N integers each — the adjacency matrix of the graph: the j-th integer of th i-th line is 1, if vertices i and j are connected with an edge, and 0, if there is no edge between them. The main diagonal contains zeros only. The matrix is symmetrical in respect to the main diagonal.

Output

Print "YES" if the graph is a tree, and "NO" otherwise.

Examples

input Copy

```
6
0 1 1 0 0 0
1 0 1 0 0 0
1 1 0 0 0 0
0 0 0 0 1 0
0 0 0 1 0 0
0 0 0 0 0 0
```

output Copy

```
NO
```

input Copy

```
3
0 1 0
1 0 1
0 1 0
```

output Copy

```
YES
```