Lab 9 - Graph

#A - Matrix 裸邻接矩阵

Information

Time Limit	Memory Limit	Data Amount	Problem Type
2000ms	128MiB	10	Tradition

Description

Given a **directed weighted** graph, please print its adjacency matrix representation.

Input

The first line contains two integers *n*,*m*: number of vertexes, number of edges.

For the next *m* lines, each line contains three integers: *x*, *y*, *z*, representing an edge $x \to y$ with weight *z*.

Output

A *n* * *n* matrix.

If edge $x \to y$ does not exist, please set the corresponding value of the matrix to -1.

Sample Test Data

Input #1

```
4 6
1 3 1
2 1 2
4 2 3
3 3 4
1 2 5
3 4 6
```

Output #1

```
-1 5 1 -1
2 -1 -1 -1
-1 -1 4 6
-1 3 -1 -1
```

Tips

For 100% cases: 1<=n<=1000, 0<=m<=n * n, 1<=z<=10^9.

It's guaranteed that there are no duplicate edges in the graph.

#B - Tree?

Information

Time Limit	Memory Limit	Data Amount	Problem Type
2000ms	128MiB	10	Tradition

Description

Given an **undirected** graph, please determine if it is a tree or not.

Input

The first line contains two integers *n*, *m*: number of vertexes, number of edges.

For the next *m* lines, each line contains two integers: *x*, *y*, representing an edge.

Output

If it is a tree, print **"tree"** (Without quotes), otherwise print **"graph"** (Without quotes).

Sample Test Data

Input #1

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

Output #1

```
tree
```

Input #2

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

Output #2

```
graph
```

Data Limit

For 50% cases: 1<=*n*,*m*<=2000;

For 100% cases: 1<=n,m<=10^5, 1<=*vertex*<=*n*.

It's guaranteed that there are no duplicate edges and self loops in the graph.

#C - Graph traversal 反图来做?

Information

Time Limit	Memory Limit	Data Amount	Problem Type
2000ms	128MiB	10	Tradition

Description

Given a **directed** graph, for each vertex *v*, please calculate $P(v)$: starting at *v*, the maximum index of vertex which can be reached.

Input

The first line contains two integers *n*, *m*: number of vertexes, number of edges.

For the next m*m* lines, each line contains two integers: *x*, *y*, representing an edge $x \to y$.

Output

n integers: $P(1), \ldots, P(n)$.

Sample Test Data

Input #1

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

Output #1

```
4 4 3 4
```

Data Limit

For 50% cases: $1 <= n, m <= 2000$;

For 100% cases: $1 <= n, m <= 10^5$, 1<=*vertex*<=*n*.

#D - Asuka's maze

Information

Time Limit	Memory Limit	Data Amount	Problem Type
2000ms	128MiB	10	Tradition

Description

Asuka is trapped in a maze!

The maze can be thought of as infinite, and consists of infinite $N * M$ matrices repeated.

Some places in the matrix are roads, denoted by **'.'**; Some places are walls, denoted by **'#'**.

The positions of Asuka are denoted by **'S'**.

For a point (x, y) in the maze, if $(x \ mod \ N, y \ mod \ M)$ is '.' or 'S', then that place is the road;

If $(x \ mod \ N, y \ mod \ M)$ is '#', then this place is the wall. Asuka can move up, down, left, and right, but certainly not to the wall.

Please tell Asuka if she can get out of the maze (if she can get infinitely far from the starting point, she can get out). If not, Asuka will have to initiate the Self-destruct mode. Of course not unless she have to!

Input

The input contains several pieces of data. It is end with *N*=*M*=−1.

For each piece of data:

The first line contains two integers: *N*, *M*.

Next is a character matrix of size *N*, *M*, representing the unit matrix $(0, 0) - (N - 1, M - 1)$ in the maze.

Output

If Asuka can get out of the maze, print **"Yes"** (Without quotes), otherwise print **"No"** (Without quotes).

Sample Test Data

Input #1

```
5 4
##.#
##S#
#..#
#.##
#..#
5 4
##.#
##S#
#..#
..#.
#.##
-1 -1
```

Output #1

```
Yes
No
```

Data Limit

For 60% cases, $1 <= N, M <= 20$.

For 100% cases: $1 <= N, M <= 1500$.

There are no more than 10 pieces of data in one input.

#E - Transaction

Information

Time Limit	Memory Limit	Data Amount	Problem Type
2000ms	128MiB	4	Tradition

Description

There are some transactions to commit. Some transactions must be commit after committing other transactions, which we call the pre-transactions. A transaction *i* need to spend a_i time to commit. Transactions without dependencies can be committed at the same time.

Note that the pre-transactions of transaction *i* can only be a subset of $\{1, 2, \ldots, i-1\}$. At least one transaction have no pre-transaction. The transaction that can be done first is labeled transaction 1.

Now given all the transactions and their pre-transactions list, Please calculate the minimum time to commit all transactions.

Input

The first line contains an integers *n*: number of transactions.

For the next *m* lines, each line has several integers separated by spaces, respectively:

The number of transactions *i* (which is ordered in the input);

The time it takes t_i to commit the transaction;

The pre-transactions $\{p_{i_1}, p_{i_2}, \ldots p_{i_k}\}$, that are terminated by a number 0.

Output

One integer: the answer.

Sample Test Data

Input #1

```
7
1 5 0
2 2 1 0
3 3 2 0
4 6 1 0
5 1 2 4 0
6 8 2 4 0
7 4 3 5 6 0
```

Output #1

```
23
```

Data Limit

For 50% cases, $n <= 100$.

For 100% cases: $1 <= n <= 10000, 1 <= t_i <= 100, 0 <= k_i <= min(i - 1, 100)$. (*k* is the number of the pre-transactions of a transaction)