Task: OSI
Zones

XXVI OI, Stage II, Day two. Source file `osi.*` **Available memory: 256 MB.** *14.02.2019*

Bitown is rife with traffic accidents. The mayor believes that the complexity of the street network is to blame: n intersections are interconnected by m bidirectional streets. To simplify the network (and hopefully also navigating it), he intends to make every street unidirectional.

The mayor would like to choose a direction for each street so as to minimize the overall number of zones. A **zone** is a maximal (i.e., non-extensible) set of intersections such that every intersection in the set can be reached from every other intersection in the set while observing street directions.

Your task is to write a program that will determine the minimum number of zones, together with the street directions that produce this many zones.

Input

The first line of the standard input contains two integers n and m ($2 \le n \le 1\,000\,000$, $1 \le m \le 1\,000\,000$), separated by a single space, which specify the number of intersections and of streets in Bitown respectively. The intersections are numbered with integers from 1 to n.

The m lines that follow describe the street network; the i-th such line holds two integers a_i, b_i ($1 \le a_i$, $b_i \le n$, $a_i \ne b_i$), separated by a single space, which indicate that the i-th street directly links the intersections no. a_i and b_i. There can be more than one street directly linking any pair of intersections.

Output

A single integer should be printed to the first line of the standard output: the minimum number of zones that that results from an appropriate choice of street directions. To the second line a single string of length m that encodes those directions should be printed; the i-th character of the string corresponds to the direction of the i-th street, in the input order. The character `>` stands for direction from intersection a_i to intersection b_i, whereas `<` stands for direction from intersection b_i to intersection a_i. No other characters are allowed. Your program can report an arbitrary optimal choice of directions if more than one exists.

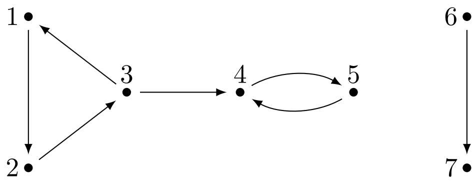
Example

For the following input data:

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

the correct answer is:

```
4
><>>><<
```



Explanation for the example: The figure depicts an optimal choice of street directions, which produces four zones, encompassing the following intersections: $\{1, 2, 3\}$, $\{4, 5\}$, $\{6\}$, and $\{7\}$. Notice that even though reaching intersection 4 from intersection 3 is possible, they are not in a single zone, because it is not possible to reach intersection 3 from intersection 4.

Sample grading tests:

`1ocen`: $n = 7$, $m = 10$ – a small correctness test;

`2ocen`: $n = 5000$, $m = 4999$ – a path (for each $i \in \{1, \ldots, n-1\}$, there is a street linking intersections i and $i+1$);

`3ocen`: $n = 2000$, $m = 20\,000$ – a "tenfold" cycle (for each i, there are exactly ten streets linking intersections i and $(i+1) \bmod n$);

`4ocen`: $n = 500\,000$, $m = 999\,998$ – for each $i \in \{1, \ldots, n-2\}$, there are streets linking intersections i and $i+1$ as well as i and $i+2$; in addition, there are two streets linking intersections $n-1$ and n.

Grading

The set of tests consists of the following subsets. Within each subset, there may be several unit tests.

Time limits for each subset are published in SIO.

Subset	**Condition**	**Score**
1	n, $m \leq 5000$	16
2	$n \leq 2000$, $m \leq 20\,000$	12
3	$n \leq 5000$	20
4	no further conditions	52

For every test where only one of the lines output by your program is correct, it will be awarded 50% of that test's score. To have the second output line accepted this way, your program has to print both lines, and the first one has to contain a 32-bit integer (of `int` type).