Problem G

Go Make It Complete

Andi has a network G with N machines and M links; each link connects exactly two different machines. Due to some circumstances, Andi has to make his network "complete", i.e. every machine is directly linked to every other machine. This means Andy has to add a new link between any pair of machines which are not already directly linked.

In order to accomplish his goal, Andi outsources the work to a company, Go. Go accepts any work order on a network G with a requested integer k, i.e. $go(G, k)$. The way Go works: First, it creates a list L containing **all** pair of machines which are **not** yet directly linked. Then, Go evaluates each pair of machines (a, b) from L and create a new link between them if $\delta_a + \delta_b \geq k$. Note that δ_a is the degree of machine a, i.e. the number of links a has at the time of evaluation. Similarly, δ_b is for machine b.

The problem with Go's procedure is it evaluates each pair of machines in the order appeared in L. For example, let G be a network with $N = 4$ machines (machine 1, 2, 3, and 4) and $M = 3$ links; the links are $\{(1, 2), (2, 3), (3, 4)\}$. The degree of each machine before a work order is requested is: $\delta_1 = 1$, $\delta_2 = 2$, $\delta_3 = 2$, $\delta_4 = 1$, or simply can be written as $[1, 2, 2, 1]$. Let say a work order with $k = 3$ is requested ($go(G, 3)$).

Consider these two lists:

- $L_1 = ((1, 4), (1, 3), (2, 4))$.
  - × Evaluate $(1, 4)$ and don't create a new link since $\delta_1 + \delta_4 = 1 + 1 = 2$. The degree is still $[1, 2, 2, 1]$.
  - ✓ Evaluate $(1, 3)$ and create a new link since $\delta_1 + \delta_3 = 1 + 2 = 3$. The degree becomes $[2, 2, 3, 1]$.
  - ✓ Evaluate $(2, 4)$ and create a new link since $\delta_2 + \delta_4 = 2 + 1 = 3$. The degree becomes $[2, 3, 3, 2]$.

  The final result is a network with 5 links, which is not complete as it misses the $(1, 4)$ link.
- $L_2 = ((2, 4), (1, 3), (1, 4))$.
  - ✓ Evaluate $(2, 4)$ and create a new link since $\delta_2 + \delta_4 = 2 + 1 = 3$. The degree becomes $[1, 3, 2, 2]$.
  - ✓ Evaluate $(1, 3)$ and create a new link since $\delta_1 + \delta_3 = 1 + 2 = 3$. The degree becomes $[2, 3, 3, 2]$.
  - ✓ Evaluate $(1, 4)$ and create a new link since $\delta_1 + \delta_4 = 2 + 2 = 4$. The degree becomes $[3, 3, 3, 3]$.

  The final result is a network with 6 links and complete.

As we can see, L_2 produces a complete network, while L_1 is not.

Ordering a work to Go can be very expensive with lower k, thus, Andi has to make k as high as possible while maintaining the possibility that a complete network can be achieved with k. In other words, Andi wants the highest k such that there exists L such that $go(G, k)$ can produce a complete network, and for any j ($j > k$), there is no valid L for $go(G, j)$ which can produce a complete network. Your task in this problem is to find such k.

Input

Input begins with a line containing two integers: N M ($2 \leq N \leq 500$; $0 \leq M < \frac{N \times (N-1)}{2}$) representing the number of machines and the number existing links, respectively. The machines are numbered from 1 to N. The next M lines, each contains two integers: a_i b_i ($1 \leq a_i < b_i \leq N$) representing an existing link connecting machine a_i and b_i. You are guaranteed that each pair (a_i, b_i) will appear at most once in the input.

Output

Output contains an integer k in a line, as requested.

Sample Input #1

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

Sample Output #1

```
3
```

Sample Input #2

```
5 0
```

Sample Output #2

```
0
```

Explanation for the sample input/output #2

Andi's network has no link at all at the beginning. To make his network complete, Andi has to order $go(G, 0)$.

Sample Input #3

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

Sample Output #3

```
2
```