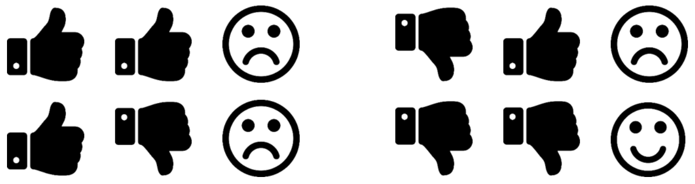
Unsatisfying



Usually a computer scientist tries to satisfy some constraints. This time you will try to make some logical statements unsatisfiable.

You will be given a list of logical statements of the following form:

$$\begin{aligned} p_1 &\vee p_2, \\ \neg p_2 &\vee p_3, \\ p_3 &\vee \neg p_4. \end{aligned}$$

Each of the p_i is a proposition that can be either `true` or `false`. The disjunction operator $\vee$ can be interpreted as logical OR. The $\neg$ symbol is negation, which negates the value of the subsequent proposition from `true` to `false`, and vice versa.

To satisfy a list of logical statements, you must assign either `true` or `false` to each of the propositions such that every disjunction in the list evaluates to `true`.

Your task is to add disjunctions to the list to make the list of statements unsatisfiable. But the additional disjunctions you add cannot use the negation symbol!

All disjunctions (both those given and ones you add) must have exactly 2 terms.

1 Input

The first line of input contains two space-separated integers n and m ($1 \le n, m \le 2{,}000$), with n representing the number of propositions and m representing the number of disjunctions.

Each of the next m lines contains two space-separated integers a_i and b_i ($1 \le |a_i|, |b_i| \le n$), which describes the two propositions of the ith disjunction in the given list. A positive value a_i, for example, represents the proposition p_{a_i}. On the other hand, if a_i is negative, it represents the negated proposition, namely $\neg p_{|a_i|}$.

The second sample input corresponds to the following list of logical statements:

$$\begin{aligned} p_1 &\vee p_2, \\ \neg p_1 &\vee \neg p_3, \\ \neg p_2 &\vee p_3, \\ p_3 &\vee \neg p_4, \\ \neg p_2 &\vee \neg p_3. \end{aligned}$$

2 Output

Output, on a single line, a single integer representing the minimum number of additional disjunctions necessary to make the list unsatisfiable. If it is not possible to make the list unsatisfiable, print `-1` instead. Each disjunction you add must have two (not necessarily distinct) propositions, and you may not use negated propositions.

In the second sample case, adding a disjunction $p_2 \vee p_2$ makes the list unsatisfiable.

3 Sample Input and Output

`2 1`
`1 2`
	`-1`

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