Double Move

Problem name	Double Move
Input file	standard input
Output file	standard output
Time limit	5 seconds
Memory limit	256 megabytes

Alice and Bob are playing a game, and Claire is helping them. There are n stones, numbered from 1 to n. The game consists of three phases.

In the first phase, Alice and Bob make alternating moves. Alice moves first. In each move, a player declares their intention to take a stone, but instead of saying exactly which one, they name two options. It is possible for the two options to be the same. It is also possible to name stones that were already named in the previous moves. No stones are actually taken during the first phase — the players merely declare their intentions for the second phase. The first phase ends when $n+1$ declarations have been made.

Here is an example first phase for $n = 3$:

1. Alice: "I will take either stone 1 or stone 3"
2. Bob: "I will take either stone 2 or stone 2"
3. Alice: "I will take either stone 3 or stone 2"
4. Bob: "I will take either stone 1 or stone 3"

In the second phase, for each of the $n+1$ declarations that have been made, Claire picks one of the two options by saying "first" or "second". We will call each sequence of $n+1$ choices made by Claire a *scenario*. Note that there are exactly $2 \cdot 2 \cdot 2 \cdot \cdots \cdot 2 = 2^{n+1}$ possible scenarios. (Even if, in some declaration, the first and the second option are the same, we consider picking the "first" or the "second" option to result in different scenarios.)

Here's one of the 16 scenarios Claire could choose in the above example:

1. "First": Alice will take stone 1
2. "First": Bob will take stone 2
3. "Second": Alice will take stone 2

4. "First": Bob will take stone 1

Finally, in the third phase, Alice and Bob actually start taking stones according to Claire's decisions. The first player who cannot make the required move — because the corresponding stone has already been taken — loses the game. Note that since there are n stones and $n+1$ moves, one of the players must eventually lose the game.

In the above example, Alice starts by actually taking stone 1. Bob continues by taking stone 2. Alice wants to continue by taking stone 2, but it was already taken, so Alice loses the game and therefore Bob wins.

You are given the number n, and the state of the game at some point during the first phase: a sequence of k declarations that have already been made. These declarations can be completely arbitrary.

From this point on, Alice and Bob will play the game optimally, as described in the following paragraph.

Regardless of how Alice and Bob play, Claire is equally likely to choose each of the 2^{n+1} possible scenarios. Alice and Bob know this and therefore when playing optimally, they are both trying to minimize the number of scenarios in which they lose.

Assume Alice and Bob will play the rest of the game as described above. For each of the two players find the number of scenarios in which they win the game.

Input

The first line of input contains two space-separated integers n and k ($1 \leq n \leq 35$, $0 \leq k \leq n+1$): the number of stones and the number of declarations that have already been made.

The rest of the input consists of k lines, each describing one declaration, in the order in which they were made. Each of these lines contains two space separated integers: the numbers of two stones (both between 1 and n, inclusive, and not necessarily distinct).

Note that when $k < n+1$, the next player to make a declaration depends on the parity of k.

Output

Output a single line with two space-separated integers: the number of scenarios in which Alice wins and the number of scenarios in which Bob wins, assuming both players play the rest of the game as described in the statement.

Note that the sum of the two numbers you print must be equal to 2^{n+1}.

Scoring

Subtask 1 (15 points): $n \leq 4$.

Subtask 2 (34 points): $n \leq 10$.

Subtask 3 (20 points): $n \leq 25$.

Subtask 4 (10 points): $k = 0$.

Subtask 5 (21 points): no additional restrictions.

Examples

standard input	standard output
3 4 1 3 2 2 3 2 1 3	4 12
2 0	4 4

Note

The first example corresponds to the example from the problem statement. There are no more declarations to be made, so we just need to see how many of Claire's possible decisions lead to Alice's win, and how many of them lead to Bob's win. Alice will win if Claire picks stone 1 for her on the first move, and stone 3 for her on the second move, and will lose in all other cases.

In the second example, if Alice starts by declaring "1 1" , Bob will continue "2 2", and no matter what Alice declares on the third move, she will lose, as Claire will have to pick stone 1 for the first move, and stone 2 for the second move, and there will be no stones left for Alice in the third move. However, this is not the optimal first move for Alice: instead, she should start by declaring "1 2". Then, no matter what Bob does on the second move and what Alice does on the third move, each of them will win in 4 cases out of 8.