BEARs

The Infinite city is divided into unitary square blocks by an infinite number of south-north and west-east two-way streets. One of the south-north streets is labeled 0, and the street numbers increase to the east and decrease to west. Similarly, one of the west-east streets is labeled 0, and the numbers increase to the north and decrease to the south.

Every intersection is labeled by an ordered pair of numbers of the streets that intersect (the first being the number of south-north street). Some street sections are more important and are called main streets.

One day sheriff Wolf (the fiercest caretaker of the Infinite city) is patrolling the streets and at intersection (A, B) he notices a car with a few members of the well-known BEAR gang. Wolf has heard of the BEARs' plans to break into the city's Honey Warehouse, which is located near the intersection $(0, 0)$, and decides to stop them.

However, they haven't commited any crime so far and Wolf can't arrest them. But he has the authority to stop his car at any intersection and block exactly one of the four unit segments that meet at this intersection. However he can't block a unit segment that belongs to a main street.

So Wolf decides to pursue the BEARs and just before they reach an intersection, he may overtake their car and block one of the four unit segments at the intersection. The BEARs will be able to drive into the intersection, but they won't be able to exit the intersection to a segment blocked by the sheriff's car.

The sheriff wants to keep the BEARs as far away from the Honey Warehouse as possible. Find the maximum distance D, such that any intersection (x, y) the BEARs may reach satisfies the condition $\max(|x|, |y|) \geq D$.

Input. The first line of the file `bears.in` contains two integers: A and B ($|A| \leq 10^6$, $|B| \leq 10^6$), the starting point of the BEARs. The second line contains one integer N ($0 \leq N \leq 500$), the number of main streets. Each of the following N lines contains four integers: X_1, Y_1, X_2, Y_2 ($|X_i| \leq 10^6$, $|Y_i| \leq 10^6$), meaning that the street section between intersections (X_1, Y_1) and (X_2, Y_2) is a main street. Either $X_1 = X_2$ or $Y_1 = Y_2$ holds.

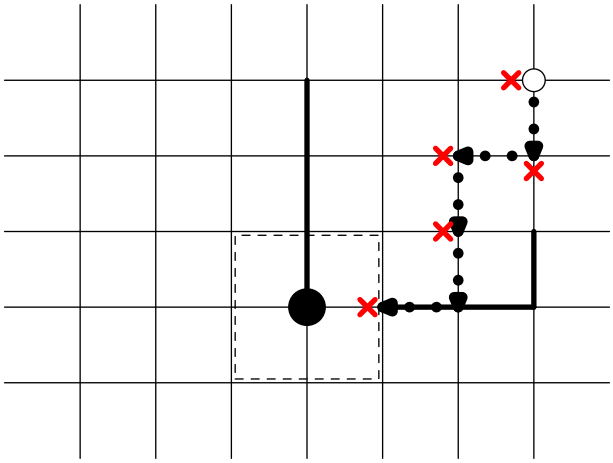
Output. The only line of the file `bears.out` must contain the maximum value of D.

Sample.

bears.in
```
3 3
3
1 0 3 0
0 0 0 3
3 0 3 1
```

bears.out
```
1
```

The following figure illustrates how the BEARs can get to within distance 1 of the warehouse:



Even though the BEARs may continue trying forever, the sheriff can prevent them from ever getting closer to the warehouse.

Lego

You are using Lego building blocks to train an artificial vision system. Write a program that, given pictures of a Lego construction taken from two angles, calculates in how many different ways it can be built.

In this task, there is only one kind of lego block (with 2×2 "knobs", see picture below), but it can have three different colors: white (W), gray (G) or black (B). All blocks exist in unlimited amounts. You use a quadratic base with 6×6 knobs. Every block must have its edges parallel to this base and no block may extend outside of it. Every block must rest upon at least one underlying block.

Left: An allowed way to place a block on top of another one. Center: An illegal way (the upper block hangs in the air). Right: Another illegal way (the upper block extends outside the base).

Input. The first line of the file `lego.in` contains H ($1 \leq H \leq 6$), the height of the construction. Then follow H lines with 6 characters on each line, giving a picture of the construction as seen from one side (marked A on the figure below). The jth character on the ith line specifies what you see looking at the jth column from the left on the ith row from above. Each character may be one of 'W', 'G', 'B' or '.', specifying a color ('W', 'G', or 'B') or a hole ('.'). Note that you cannot estimate the depth, so a color seen in a certain position may either belong to a block near the front edge, or further back, provided no other block is blocking the sight.

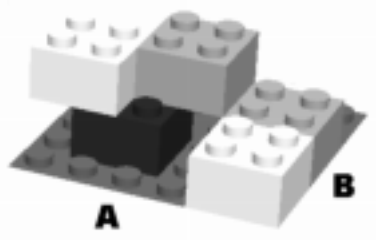
The first picture is followed by another set of H lines with the construction seen from an angle where the observer has moved 90 degrees counterclockwise around the construction (marked B on the figure below).

Output. The program should output one line to the `lego.out` output file, containing a single integer: the number of different Lego constructions that satisfy the pictures given in the input. Note that even if two different possible constructions could be obtained from each other by rotating or mirroring, they both should be counted. For the given input, the answer will always fit in a signed 64-bit integer.

Sample.

`lego.in`
```
2
WWGG..
.BB.WW
.WGG..
WWGG..
```

`lego.out`
```
6
```



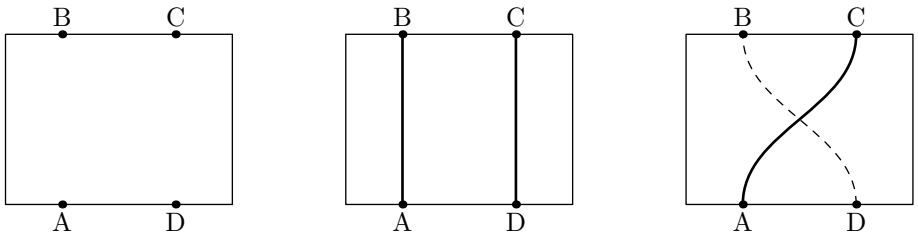
One of the possible constructions in the example.

Printed Circuit Board

In a printed circuit board, conductive wires are laid on a non-conductive board. Because the conductors in the same layer cannot cross without creating short-circuits, boards with conductors divided into several layers separated by non-conductive board material are used in more complex cases. However, boards with more layers are more expensive. So, manufacturers try to allocate the required conductors to layers in a way that minimizes the number of layers needed.

In this task we look at boards where each conductor is connecting two ports located on opposite edges of the board and seek to minimize the cost of such a board.

Consider, for example, the board shown on the left on the figure below. If one conductor has to connect A to B and another D to C, this could be achieved in a single layer, as shown in the middle on the figure. But a conductor connecting A to C and another connecting D to B could not be laid out in the same layer, as can be seen on the right on the figure.



Write a program that is given the locations of the endpoints of the N conductors on a $W \times H$ board and determines the minimal number of layers needed to accommodate all of them.

It may be assumed the width of the conductors is very small compared to the distances between the ports. That is, between any two conductors, there is always enough room for a third one.

Input. The first line of the text file `pcb.in` contains N ($1 \leq N \leq 10^5$), the number of connectors. Each of the following N lines contains two integers, X_{i1} and X_{i2} ($0 \leq X_{ij} \leq 10^6$), separated by a space, meaning that the i-th conductor has to connect the points $(X_{i1}, 0)$ and (X_{i2}, H). It may be assumed that all the $2 \cdot N$ endpoints given in the input are distinct.

Output. The first and only line of the text file `pcb.out` should contain a single integer, the minimal number of layers needed to accommodate all the required conductors.

Sample.

pcb.in	pcb.out
2 1 1 3 3	1

Sample.

pcb.in	pcb.out
2 1 3 3 1	2