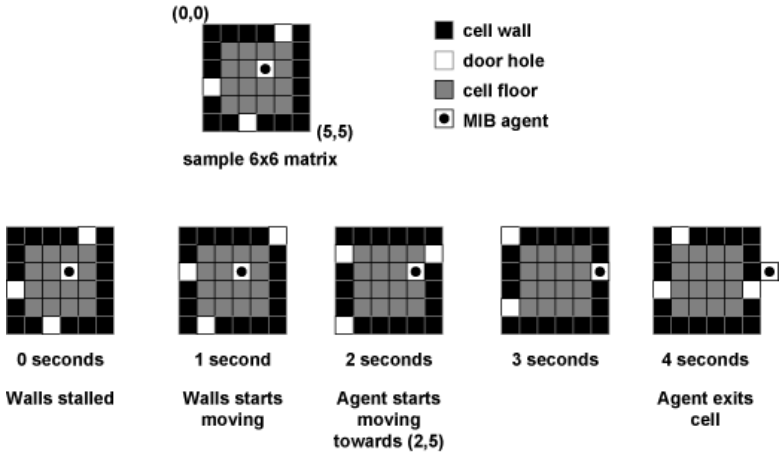
Problem E
Escape

Source file name: `escape.c`, `escape.cpp` *or* `escape.java`

Suppose you are a MIB (Men in Black) agent who has been just abducted by aliens. They have locked you in a very strange prison cell. The cell is completely rectangular and it seems to have no roof. You are located somewhere inside the cell, and for some reason you cannot walk around the cell, you are simply stuck at that position. A very strange detail is that the walls are moving constantly around you, in a periodic fashion, at a regular speed. Even stranger is that the walls have actually door holes which you could use to get out of the cell, but the holes are moving constantly with the walls and, anyway, you cannot move from where you are. It's all part of some bizarre alien torturing method.

Your fellow MIB agents are aware of your adduction and are trying to help you, using a secret communication device implanted in your skin. They have informed you that you cannot move because the floor has some kind of gravity control mechanism to hold you still. Your fellow MIB agents can disable the device remotely just for a little time, so you have only one chance to escape. In addition, they can only disable the device to allow you to move either horizontally or vertically. Once the device is disabled, you would have to run to the door and try to exit the cell through a door hole. Therefore, you have to calculate the exact moment a hole will pass in front of you. Once you start moving you cannot change your direction, or it might be too late to reach the door hole.

Since you only have one chance to escape, you have decided to run a simulation first. To simulate this process you are assuming the cell looks like a matrix when seem from the sky, and each matrix position could only be: part of the cell wall, part of the cell floor, a door hole or yourself. You are assuming the walls begin the simulation stalled, and then start moving clockwise, at one matrix position per second. You also move at one matrix position per second. The following image depicts a sample matrix of your simulation and the whole escaping process.



Given the size of the cell, the position of the door holes and your position inside the cell, your task is to write a program that simulates the situation and reports the minimum amount of time that would take you to exit the cell, counted from the moment the simulation starts, and the distance you have to run during your escape. This distance is represented by the number of matrix positions you have to pass through to reach the door hole, plus the one last step you need to actually exit the cell.

If you find two or more possible solutions with the same timing, then choose the solution with the lowest running distance.

Input

The input contains several test cases, each one corresponding to one simulation. The first line of a test case contains three integers M, N and H. The first two integers indicate the size of the cell ($3 \leq M, N \leq 1000$), corresponding to the number of rows and columns of the cell, respectively. The last integer indicates the number of door holes ($0 < H < (2 \times M) + (2 \times N) - 4$). Then, H lines follow, each one containing two integer values, corresponding to the matrix position of a door hole. These two values will be separated from each other by exactly one blank space. All door-hole-positions will be different from each other. Finally, there will be a single line with two integers, separated by a blank space, indicating your initial matrix position inside the cell.

The end of input is indicated by a test case with $M = 0$, $N = 0$ and $H = 0$.

The input must be read from standard input.

Output

For each test case specified in the input, your program must print a single line containing two integers, separated from each other by one blank space. The first integer should be the minimum amount of time that would take you to exit the cell, counted from the moment the simulation starts. The other integer should be the distance you have to run during your escape.

The output must be written to standard output.

Sample input	**Output for the sample input**
6 6 3 0 4 3 0 5 2 2 3 3 3 4 0 0 0 2 2 0 2 2 1 1 0 0 0	4 3 2 2