You are given an n × m matrix, which contains all the integers from 1to n \* m, inclusive, each exactly once.

Initially you are standing in the cell containing the number 1. On each turn you are allowed to move to an adjacent cell, i.e. to a cell that shares a common side with the one you are standing on now. It is prohibited to visit any cell more than once.

Check if it is possible to visit all the cells of the matrix in the order of increasing numbers in the cells, i.e. get to the cell with the number 2 on the first turn, then move to 3, etc.

**Example**

* For
* matrix = [[1, 4, 5],
* [2, 3, 6]]

the output should be  
findPath(matrix) = true;

* For
* matrix = [[1, 3, 6],
* [2, 4, 5]]

the output should be  
findPath(matrix) = false.

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] array.array.integer matrix**

A non-empty 2-dimensional array of integers representing a rectangular matrix.

*Guaranteed constraints:*  
1 ≤ matrix.length ≤ 5,  
1 ≤ matrix[0].length ≤ 10,  
1 ≤ matrix[i][j] ≤ 25.

* **[output] boolean**

<https://codefights.com/challenge/n927Ns3HD2q3EMeXB/solutions>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static bool findPath(int[][] matrix)

{

int f = 0, c = 0;

for (int i = 0; i < matrix.Length; i++)

{

for (int j = 0; j < matrix[i].Length; j++)

{

if (matrix[i][j] == 1)

{

f = i;

c = j;

goto e;

}

}

}

e:

//Console.WriteLine("hola");

//int i =

int v =1;

//int max = matrix.Length \* matrix[0].Length;

int n = matrix.Length;

int m = matrix[0].Length;

bool[][] marcas = new bool[n][];

for (int i = 0; i < n; i++)

{

marcas[i] = new bool[m];

for (int j = 0; j < m; j++)

{

marcas[i][j] = false;

}

}

marcas[f][c] = true;

while (true)

{

if (f - 1 >= 0 && matrix[f - 1][c] == v + 1)

{

marcas[f - 1][c] = true;

f--;

v++;

}

else if (f + 1 < matrix.Length && matrix[f + 1][c] == v + 1)

{

marcas[f + 1][c] = true;

f++;

v++;

}

else if (c - 1 >= 0 && matrix[f][c - 1] == v + 1)

{

marcas[f][c - 1] = true;

c--;

v++;

}

else if (c + 1 < matrix[0].Length && matrix[f][c + 1] == v + 1)

{

marcas[f][c + 1] = true;

c++;

v++;

}

else

{

break;

}

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

if (!marcas[i][j])

{

return false;

}

}

}

return true;

}

static void Main(string[] args)

{

//int[][] matrix= {new int[]{1,3,6},

// new int[]{2,4,5}};

int[][] matrix = {

new int[]{1, 2, 3,4,5},

new int[]{14,13,12,7,6},

new int[]{15,16,11,8,25},

new int[]{18,17,10,9,24},

new int[]{19,20,21,22,23}};

Console.WriteLine(findPath(matrix));

Console.ReadLine();

}

}

}