You have a seaMap of part of the Caribbean Sea, represented as a 2D array. The map shows the positions of dangerous cliffs that a wise sea dog should steer clear of.

As the captain of the most notorious pirate ship on the high seas, you're not afraid of these little rocks and are willing to navigate your ship through them. However, you're mildly curious about the number of *cliff lines* in the area. You assume that cliffs form a *line* if they are *connected* to each other, i.e. if each cliff has another cliff directly to the left, to the right, above, or below it.

Looking at the seaMap, calculate the number of *cliff lines* on it.

**Example**

For

seaMap = [[0,1,1],

[0,1,1],

[0,0,0]]

the output should be  
countCliffs(seaMap) = 1.

There is only one *cliff line* on the seaMap, which consists of 4 cliffs and forms a square.

**Input/Output**

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

A rectangular array representing your map of the Caribbean. seaMap[i][j] = 1 if there's a cliff at (i, j), and seaMap[i][j] = 0 otherwise.

*Guaranteed constraints:*  
3 ≤ seaMap.length ≤ 100,  
3 ≤ seaMap[0].length ≤ 100,  
0 ≤ seaMap[i][j] ≤ 1.

* **[output] integer**

The number of *cliff lines*.

<https://codefights.com/challenge/6eeBErC6dHrTQnyhj>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

public class Celda

{

public int Fila;

public int Columna;

public int ColorCelda;

public Celda(int fila, int columna)

{

this.Fila = fila;

this.Columna = columna;

}

public override bool Equals(object obj)

{

Celda c = (Celda)obj;

if (this.Fila == c.Fila && this.Columna == c.Columna)

{

return true;

}

return false;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

}

//static void mostrar(Celda[,] matriz, int filas, int columnas)

//{

// for (int i = 0; i < filas; i++)

// {

// for (int j = 0; j < columnas; j++)

// {

// Console.Write(matriz[i, j].ColorCelda + " ");

// //Console.Write(matriz[i, j].notacion + " ");

// }

// Console.WriteLine();

// }

//}

public static List<Celda> FloodFill(Celda[,] matriz, int filas, int columnas, Celda nodo, int viejo, int reemplazo)

{

Stack<Celda> pila = new Stack<Celda>();

if (matriz[nodo.Fila, nodo.Columna].ColorCelda != viejo)

return new List<Celda>();

pila.Push(nodo);

List<Celda> grupoSeleccionado = new List<Celda>();

grupoSeleccionado.Add(nodo);

while (pila.Count > 0)

{

Celda c = pila.Pop();

matriz[c.Fila, c.Columna].ColorCelda = reemplazo;

if (!grupoSeleccionado.Contains(matriz[c.Fila, c.Columna]))

{

grupoSeleccionado.Add(matriz[c.Fila, c.Columna]);

}

if (c.Fila > 0)

{

if (matriz[c.Fila - 1, c.Columna].ColorCelda == viejo)

{

pila.Push(new Celda(c.Fila - 1, c.Columna));

}

}

if (c.Fila < filas - 1)

{

if (matriz[c.Fila + 1, c.Columna].ColorCelda == viejo)

pila.Push(new Celda(c.Fila + 1, c.Columna));

}

if (c.Columna > 0)

{

if (matriz[c.Fila, c.Columna - 1].ColorCelda == viejo)

pila.Push(new Celda(c.Fila, c.Columna - 1));

}

if (c.Columna < columnas - 1)

{

if (matriz[c.Fila, c.Columna + 1].ColorCelda == viejo)

pila.Push(new Celda(c.Fila, c.Columna + 1));

}

//----------diagonales------------

//if (c.Fila - 1 >= 0 && c.Columna - 1 >= 0)

//{

// if (matriz[c.Fila - 1, c.Columna - 1].ColorCelda == viejo)

// pila.Push(new Celda(c.Fila - 1, c.Columna - 1));

//}

//if (c.Fila - 1 >= 0 && c.Columna + 1 < columnas)

//{

// if (matriz[c.Fila - 1, c.Columna + 1].ColorCelda == viejo)

// pila.Push(new Celda(c.Fila - 1, c.Columna + 1));

//}

//if (c.Fila + 1 < filas && c.Columna + 1 < columnas)

//{

// if (matriz[c.Fila + 1, c.Columna + 1].ColorCelda == viejo)

// pila.Push(new Celda(c.Fila + 1, c.Columna + 1));

//}

//if (c.Fila + 1 < filas && c.Columna - 1 >= 0)

//{

// if (matriz[c.Fila + 1, c.Columna - 1].ColorCelda == viejo)

// pila.Push(new Celda(c.Fila + 1, c.Columna - 1));

//}

}

return grupoSeleccionado;

}

static int countCliffs(int[][] seaMap)

{

Celda[,] matriz = new Celda[seaMap.Length, seaMap[0].Length];

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

{

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

{

matriz[i,j] = new Celda(i, j);

matriz[i, j].ColorCelda = seaMap[i][j];

}

}

int \_filas = seaMap.Length;

int \_columnas = seaMap[0].Length;

int grupos = 0;

for (int i = 0; i < matriz.GetLength(0); i++)

{

for (int j = 0; j < matriz.GetLength(1); j++)

{

if (matriz[i, j].ColorCelda == 1)

{

List<Celda> sel = FloodFill(matriz, \_filas, \_columnas, new Celda(i, j), 1, 2);

//max = Math.Max(sel.Count, max);

grupos++;

}

}

}

//Console.WriteLine(grupos);

return grupos;

}

static void Main()

{

// int[][] seaMap =

//{new int[]{0,1,1},

//new int[] {0,1,1},

//new int []{0,0,0}};

int[][] seaMap= {new int[]{1,1,1,1},

new int[]{0,1,0,0},

new int[]{0,0,1,0}};

Console.WriteLine(countCliffs(seaMap));

Console.ReadLine();

}

}

}