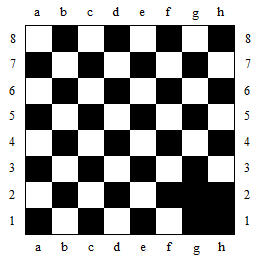
Author

[sir\_ementaler](https://codefights.com/profile/sir_ementaler)

https://codefights.com/img/coins_new.png3000

A bored student printed out a regular 8 × 8 chessboard and then used a black marker to completely fill in some of the white squares. Knowing which squares he filled in, determine the area (i.e. the number of the chessboard squares) of the largest enclosed black region on the chessboard after his actions.

**Example**

* ChessboardShapes(["g2", "h1"]) = 6.  
    
  Coloring "g2" black creates a plus-shaped black region around it consisting of 5 squares ("f2", "g1", "g2", "g3", "h2"). When "h1" is colored black, it connects to that region, but because it's in a corner of the chessboard, it doesn't add any new squares besides itself, thus the returned value is 5 + 1 = 6.
* ChessboardShapes([]) = 1.  
  If no squares are recolored, the largest black region is just any of the originally black squares.
* **[input] array.string squares**

White squares colored black, expressed in standard algebraic chess notation (all letters are lowercase, "a1" is a black square).  
0 ≤ squares.length ≤ 32.

* **[output] integer**

The area of the largest enclosed black region.

<https://codefights.com/challenge/YPo4BxN4Mu95MksRG>

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 string notacion;

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;

}

}

static Celda[,] matriz = new Celda[8, 8];

public static List<Celda> FloodFill(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 < 8 - 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 < 8- 1)

{

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

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

}

}

return grupoSeleccionado;

}

static void mostrar(Celda[,] matriz)

{

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

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

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

}

Console.WriteLine();

}

}

static void Main(string[] args)

{

int[,] m =

{

{0,1,0,1,0,1,0,1},

{1,0,1,0,1,0,1,0},

{0,1,0,1,0,1,0,1},

{1,0,1,0,1,0,1,0},

{0,1,0,1,0,1,0,1},

{1,0,1,0,1,0,1,0},

{0,1,0,1,0,1,0,1},

{1,0,1,0,1,0,1,0},

};

//int[,] m =

//{

// {0,1,0,1,0,1,0,1},

// {1,0,1,0,1,0,1,0},

// {0,1,1,1,0,1,0,1},

// {1,0,1,1,1,0,1,0},

// {0,1,0,1,0,1,0,1},

// {1,0,1,0,1,0,1,0},

// {0,1,0,1,0,1,0,1},

// {1,0,1,0,1,0,1,0},

//};

Console.WriteLine(ChessboardShapes(new string[] { "g2", "h1" }));

Console.ReadLine();

}

static int ChessboardShapes(string[] squares)

{

int[,] m =

{

{0,1,0,1,0,1,0,1},

{1,0,1,0,1,0,1,0},

{0,1,0,1,0,1,0,1},

{1,0,1,0,1,0,1,0},

{0,1,0,1,0,1,0,1},

{1,0,1,0,1,0,1,0},

{0,1,0,1,0,1,0,1},

{1,0,1,0,1,0,1,0},

};

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

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

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

}

}

int f = 0, c = 0;

for (int i = 8; i >= 1; i--)

{

for (int j = 'a'; j <= 'h'; j++)

{

matriz[f, c].notacion = (char)j + i.ToString();

c++;

}

f++;

c = 0;

}

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

if (squares.Contains(matriz[i, j].notacion))

{

matriz[i, j].ColorCelda = 1;

}

}

}

int max = 0;

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

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

{

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

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

}

}

}

return max;

}

}

}