//https://www.hackerrank.com/challenges/two-pluses

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

public class Celda

{

public int fila;

public int col;

public Celda() { }

public Celda(int fila, int col)

{

this.fila = fila;

this.col = col;

}

public override bool Equals(object obj)

{

//return base.Equals(obj);

if (this.fila == ((Celda)obj).fila && this.col == ((Celda)obj).col)

{

return true;

}

return false;

}

public override int GetHashCode()

{

return base.GetHashCode();

}

}

static int buscarMaxProd(string[] s)

{

/\* la idea es guardar en una matriz de celdas, cada celda con

cada posicion de Celda(fila, columna), PERO tengo que guadar

cada incremento actual del "mas" (de la cruz) por ejemplo

si la cruz es:

\* G

G

GGGGG

G

G

es una cruz de 9 (nueve) celdas, tengo que guardar

\* primero: “G”, luego:

\* G

\* GGG

\* G

\* ( o sea, le agrego las 4 celdas de cada punta

\* y la guardo como una nueva cruz)

\*

\* y luego guardo a

\* G

G

GGGGG

G

G

\* (las 9 celdas que forman el "mas" completo

\* esto lo hago porque pueden haber cruces que se superponen

\* entonces guardo a cada cruz en cada "etapa" de crecimiento

\* que crece con 4 celdas en cada extremo norte, sur, este, oeste

\*

\*

\*/

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

//bool[,] marcas = new bool[s.Length, s[0].Length];

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

{

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

{

int fila\_actual = i, col\_actual = j;

int arriba = i, abajo = i, izquierda = j, derecha = j;

if (s[i][j] == 'G')

{

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

cruz.Add(new Celda(i, j));

cruces.Add(cruz);

while (arriba - 1 >= 0 && abajo + 1 < s.Length

&& izquierda - 1 >= 0 && derecha + 1 < s[i].Length

&& s[arriba - 1][j] == 'G' && s[abajo + 1][j] == 'G'

&& s[i][izquierda - 1] == 'G' && s[i][derecha + 1] == 'G')

{

cruz.Add(new Celda(arriba - 1, j));

cruz.Add(new Celda(abajo + 1, j));

cruz.Add(new Celda(i, izquierda - 1));

cruz.Add(new Celda(i, derecha + 1));

//hago una nueva lista, porque sino me agrega

//abajo cruz en su etapa final, ej si la cruz es de 9

//me tiene que guardar una cruz de len 1, otra de len 5,

//y otra de len 9, porque les va agregando cada 4

//en los 4 extremos

List<Celda> aux = new List<Celda>(cruz);

cruces.Add(aux);

arriba--;

abajo++;

izquierda--;

derecha++;

}

// cruces.Add(cruz);

}

}

}

//foreach (List<Celda> lista in cruces)

//{

// //if (lista.Count == 9)

// {

// foreach (Celda unaCelda in lista)

// {

// Console.Write("(" + unaCelda.fila + " " + unaCelda.col + ") ");

// }

// Console.WriteLine();

// }

//}

//por ultimo comparo cada cruz, todas con todas,

//Y si 2 cruces comparten la misma celda, las descarto de la //solucion

int max\_len = 1;

int max\_prod = 1;

for (int i = 0; i < cruces.Count; i++)

{

for (int j = i+1; j < cruces.Count; j++)

{

List<Celda> a = cruces[i];

//me fijo si hay algun elemento en comun

List<Celda> b = cruces[j];

int k = 0;

for (k = 0; k < b.Count; k++)

{

if (a.Contains(b[k]))

{

break;

}

}

if (k == b.Count)

{

max\_prod = Math.Max(max\_prod, a.Count \* b.Count);

// Console.Write(max\_prod + " ");

}

max\_len = Math.Max(max\_len, a.Count);

}

}

if (cruces.Count == 1)

{

max\_prod = cruces[0].Count;

}

if (max\_prod == 1)

{

return max\_len;

}

// Console.ReadLine();

return max\_prod;

}

static void Main(string[] args)

{

//string[] s =

//{

// "GGGGGGGG",

// "GBGBGGBG",

// "GBGBGGBG",

// "GGGGGGGG",

// "GBGBGGBG",

// "GGGGGGGG",

// "GBGBGGBG",

// "GGGGGGGG"

//};//81

//string[] s =

//{

// "BBBBBGGBGG",

// "GGGGGGGGGG",

// "GGGGGGGGGG",

// "BBBBBGGBGG",

// "BBBBBGGBGG",

// "GGGGGGGGGG",

// "BBBBBGGBGG",

// "GGGGGGGGGG",

// "BBBBBGGBGG",

// "GGGGGGGGGG"

//}; //85

//string[] s =

//{

// "GGGGGGGGGGGG",

// "GBGGBBBBBBBG",

// "GBGGBBBBBBBG",

// "GGGGGGGGGGGG",

// "GGGGGGGGGGGG",

// "GGGGGGGGGGGG",

// "GGGGGGGGGGGG",

// "GBGGBBBBBBBG",

// "GBGGBBBBBBBG",

// "GBGGBBBBBBBG",

// "GGGGGGGGGGGG",

// "GBGGBBBBBBBG"

//}; //81

//string[] s =

//{

// "BBBBGBBBBB",

// "BBBBGBBBBB",

// "BBGGGGGBBB",

// "BBBBGBBBBB",

// "BBBBGBBBBB",

// "BBBBBBBBBB",

//};

//string[] s =

//{

// "GGGGGGGGGG",

// "GGGGGGGGGG",

// "GGGGGGGGGG",

// "GGGGGGGGGG",

// "GGGGGGGGGG",

// "GGGGGGGGGG",

//};

string[] input = Console.ReadLine().Split(' ');

int n = int.Parse(input[0]);

int m = int.Parse(input[1]);

string[] s = new string[n];

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

{

s[i] = Console.ReadLine();

}

int max = 0;

max = buscarMaxProd(s);

Console.WriteLine(max);

// Console.ReadLine();

}

}

}