Ema's Supercomputer

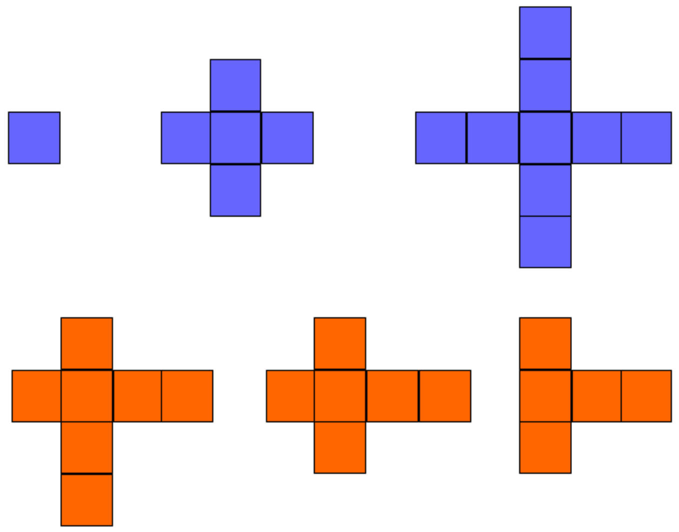
by nikasvanidze

Ema built a quantum computer! Help her test its capabilities by solving the problem below.

Given a grid of size $N \times M$, each cell in the grid is either $good$ or bad.

A *valid* plus is defined here as the crossing of two segments (horizontal and vertical) of equal lengths. These lengths must be odd, and the middle cell of its horizontal segment must cross the middle cell of its vertical segment.

In the diagram below, the blue pluses are *valid* and the orange ones are *not valid*.



Find the 2 *valid* pluses that can be drawn on $good$ cells in the grid, and print an integer denoting the maximum product of their areas.

Note: The two pluses *cannot* overlap, and the product of their areas should be maximal.

Input Format

The first line contains two space-separated integers, N and M.
The N subsequent lines contains M characters, where each character is either **G** ($good$) or **B** (bad). If the y^{th} character in the x^{th} line is **G**, then (x, y) is a $good$ cell (otherwise it's a bad cell).

Constraints

- $2 \leq N \leq 15$
- $2 \leq M \leq 15$

Output Format

Find 2 pluses that can be drawn on $good$ cells of the grid, and print an integer denoting the maximum product of their areas.

Sample Input 0

```
5 6
GGGGGG
GBBBGB
GGGGGG
GGBBGB
GGGGGG
```

Sample Output 0

```
5
```

Sample Input 1

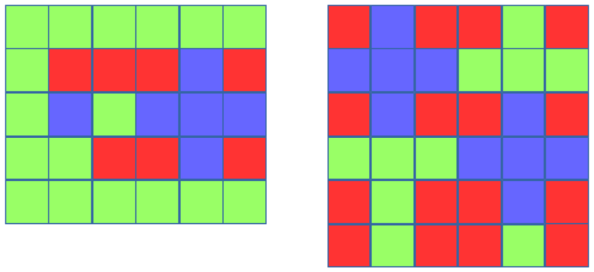
```
6 6
BGBBGB
GGGGGG
BGBBGB
GGGGGG
BGBBGB
BGBBGB
```

Sample Output 1

```
25
```

Explanation

Here are two *possible solutions* for **Sample 1** (left) and **Sample 2** (right):



Explanation Key:

- *Green*: $good$ cell
- *Red*: bad cell
- *Blue*: possible $pluses$.

For the explanation below, we will refer to a plus of length i as P_i.

Sample 0
There is enough good space to color one P_3 plus and one P_1 plus. $Area(P_3) = 5\ units$, and $Area(P_1) = 1\ unit$. The product of their areas is $5 \times 1 = 5$, so we print 5.

Sample 1
There is enough good space to color two P_3 pluses. $Area(P_3) = 5\ units$. The product of the areas of our two P_3 pluses is $5 \times 5 = 25$, so we print 25.

Submissions: 1604
Max Score: 40
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)

C#

```
using System;
using System.Collections.Generic;
using System.IO;
class Solution {


    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)
        {
            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));

                            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();
            //    }
            //}

            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();
        }



}
```

Line: 217 Col: 1

Upload Code as File   Test against custom input   Run Code   Submit Code

Congrats, you solved this challenge!

✓ Test Case #0	✓ Test Case #1	✓ Test Case #2
✓ Test Case #3	✓ Test Case #4	✓ Test Case #5
✓ Test Case #6	✓ Test Case #7	✓ Test Case #8
✓ Test Case #9	✓ Test Case #10	✓ Test Case #11
✓ Test Case #12	✓ Test Case #13	✓ Test Case #14
✓ Test Case #15	✓ Test Case #16	✓ Test Case #17
✓ Test Case #18	✓ Test Case #19	✓ Test Case #20

✓ Test Case #21

Next Challenge