



All Domains > Algorithms > Implementation > Ema's Supercomputer

Badge Progress (Details)

Points: 1057.09 Rank: 19490

Ema's Supercomputer



by nikasvanidze

Problem

Submissions

Leaderboard

Discussions

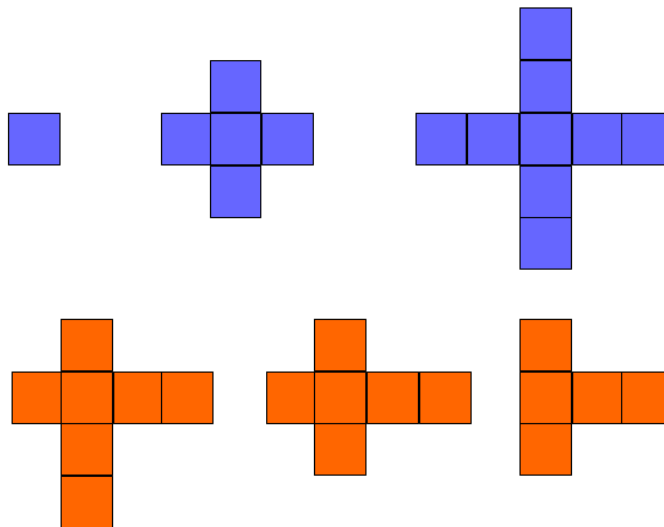
Editorial

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
```

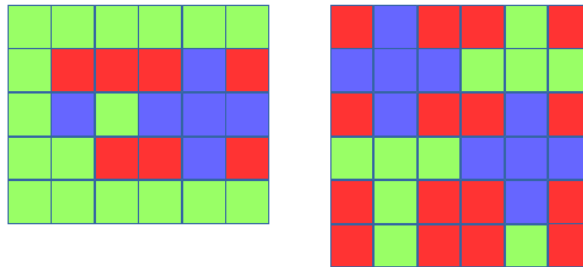
Sample Output 1

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. $Area(P_3) = 5 \text{ units}$, and $Area(P_1) = 1 \text{ unit}$. 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. $Area(P_3) = 5 \text{ units}$. The product of the areas of our two P_3 pluses is $5 \times 5 = 25$, so we print 25.

f t in

Submissions: 1604

Max Score: 40

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)

Current Buffer (saved locally, editable)

C#



```
1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4 class Solution {
5
6
7     public class Celda
8     {
9         public int fila;
```

```

10     public int col;
11
12     public Celda() { }
13
14     public Celda(int fila, int col)
15     {
16         this.fila = fila;
17         this.col = col;
18     }
19     public override bool Equals(object obj)
20     {
21         //return base.Equals(obj);
22         if (this.fila == ((Celda)obj).fila && this.col == ((Celda)obj).col)
23         {
24             return true;
25         }
26         return false;
27     }
28     public override int GetHashCode()
29     {
30         return base.GetHashCode();
31     }
32 }
33
34 static int buscarMaxProd(string[] s)
35 {
36     List<List<Celda>> cruces = new List<List<Celda>>();
37     //bool[,] marcas = new bool[s.Length, s[0].Length];
38
39     for (int i = 0; i < s.Length; i++)
40     {
41         for (int j = 0; j < s[i].Length; j++)
42         {
43             int fila_actual = i, col_actual = j;
44
45             int arriba = i, abajo = i, izquierda = j, derecha = j;
46
47             if (s[i][j] == 'G')
48             {
49                 List<Celda> cruz = new List<Celda>();
50                 cruz.Add(new Celda(i, j));
51
52                 cruces.Add(cruz);
53
54                 while (arriba - 1 >= 0 && abajo + 1 < s.Length
55                     && izquierda - 1 >= 0 && derecha + 1 < s[i].Length
56                     && s[arriba - 1][j] == 'G' && s[abajo + 1][j] == 'G'
57                     && s[i][izquierda - 1] == 'G' && s[i][derecha + 1] == 'G')
58                 {
59                     cruz.Add(new Celda(arriba - 1, j));
60                     cruz.Add(new Celda(abajo + 1, j));
61                     cruz.Add(new Celda(i, izquierda - 1));
62                     cruz.Add(new Celda(i, derecha + 1));
63
64                     List<Celda> aux = new List<Celda>(cruz);
65
66                     cruces.Add(aux);
67
68                     arriba--;
69                     abajo++;
70                     izquierda--;
71                     derecha++;
72                 }
73                 // cruces.Add(cruz);
74             }
75         }
76     }
77
78     //foreach (List<Celda> lista in cruces)
79     //{
80     //    //if (lista.Count == 9)
81     //    //{
82     //        foreach (Celda unaCelda in lista)
83     //        {
84     //            Console.WriteLine("(" + unaCelda.fila + " " + unaCelda.col + ") ");
85     //        }
86     //    }
87     //    Console.WriteLine();
88     //}
89 }
90
91
92

```



```

92
93     int max_len = 1;
94     int max_prod = 1;
95     for (int i = 0; i < cruces.Count; i++)
96     {
97         for (int j = i+1; j < cruces.Count; j++)
98         {
99             List<Celda> a = cruces[i];
100             //me fijo si hay algun elemento en comun
101             List<Celda> b = cruces[j];
102
103             int k = 0;
104             for (k = 0; k < b.Count; k++)
105             {
106                 if (a.Contains(b[k]))
107                 {
108                     break;
109                 }
110             }
111             if (k == b.Count)
112             {
113                 max_prod = Math.Max(max_prod, a.Count * b.Count);
114                 // Console.Write(max_prod + " ");
115             }
116
117             max_len = Math.Max(max_len, a.Count);
118         }
119     }
120
121     if (cruces.Count == 1)
122     {
123         max_prod = cruces[0].Count;
124     }
125     if (max_prod == 1)
126     {
127         return max_len;
128     }
129
130     // Console.ReadLine();
131     return max_prod;
132
133 }
134
135 static void Main(string[] args)
136 {
137
138
139
140     //string[] s =
141     //{
142     //    "GGGGGGGG",
143     //    "GBGBGBGB",
144     //    "GBGBGBGB",
145     //    "GGGGGGGG",
146     //    "GBGBGBGB",
147     //    "GGGGGGGG",
148     //    "GBGBGBGB",
149     //    "GGGGGGGG"
150     //}; //81
151
152     //string[] s =
153     //{
154     //    "BBBBBGBGBG",
155     //    "GGGGGGGGGG",
156     //    "GGGGGGGGGG",
157     //    "BBBBBGBGBG",
158     //    "BBBBBGBGBG",
159     //    "GGGGGGGGGG",
160     //    "BBBBBGBGBG",
161     //    "GGGGGGGGGG",
162     //    "BBBBBGBGBG",
163     //    "GGGGGGGGGG"
164     //}; //85
165
166     //string[] s =
167     //{
168     //    "GGGGGGGGGGGG",
169     //    "GBGBBBBBBBBG",
170     //    "GBGBBBBBBBBG",
171     //    "GGGGGGGGGGGG",
172     //    "GGGGGGGGGGGG",
173     //    "GGGGGGGGGGGG",

```



```
174         //      "GGGGGGGGGGG",
175         //      "GBGGBBBBBBBG",
176         //      "GBGGBBBBBBBG",
177         //      "GBGGBBBBBBBG",
178         //      "GGGGGGGGGGG",
179         //      "GBGGBBBBBBBG"
180     //}; //81
181
182     //string[] s =
183     //{
184     //    "BBBBGBBBBB",
185     //    "BBBBGBBBBB",
186     //    "BBGGGGGGBB",
187     //    "BBBBGBBBBB",
188     //    "BBBBGBBBBB",
189     //    "BBBBGBBBBB",
190
191     //};
192
193
194     //string[] s =
195     //{
196     //    "GGGGGGGGGG",
197     //    "GGGGGGGGGG",
198     //    "GGGGGGGGGG",
199     //    "GGGGGGGGGG",
200     //    "GGGGGGGGGG",
201     //    "GGGGGGGGGG",
202     //};
203
204
205
206
207     string[] input = Console.ReadLine().Split(' ');
208     int n = int.Parse(input[0]);
209     int m = int.Parse(input[1]);
210
211     string[] s = new string[n];
212
213     for (int i = 0; i < n; i++)
214     {
215         s[i] = Console.ReadLine();
216     }
217
218     int max = 0;
219
220
221     max = buscarMaxProd(s);
222
223     Console.WriteLine(max);
224
225     // Console.ReadLine();
226 }
227
228
229
230 }
```

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 #3
✓ Test Case #6
✓ Test Case #9
✓ Test Case #12
✓ Test Case #15
✓ Test Case #18

✓ Test Case #1
✓ Test Case #4
✓ Test Case #7
✓ Test Case #10
✓ Test Case #13
✓ Test Case #16
✓ Test Case #19

✓ Test Case #2
✓ Test Case #5
✓ Test Case #8
✓ Test Case #11
✓ Test Case #14
✓ Test Case #17
✓ Test Case #20

✓ Test Case #21

Next Challenge

Copyright © 2016 HackerRank. All Rights Reserved

Join us on IRC at [#hackerrank](#) on freenode for hugs or bugs.

[Contest Calendar](#) | [Interview Prep](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) | [Request a Feature](#)