The Grid Search

by PRASHANTB1984

Given a 2D array of digits, try to find the occurrence of a given 2D pattern of digits. For example, consider the following 2D matrix:

```
1234567890
0987654321
1111111111
1111111111
2222222222
```

Assume we need to look for the following 2D pattern:

```
876543
111111
111111
```

If we scan through the original array, we observe that the 2D pattern begins at the second row and the third column of the larger grid (the 8 in the second row and third column of the larger grid is the top-left corner of the pattern we are searching for).

So, a 2D pattern of P digits is said to be present in a larger grid G, if the latter contains a contiguous, rectangular 2D grid of digits matching with the pattern P, similar to the example shown above.

Input Format

The first line contains an integer, T, which is the number of test cases. T test cases follow, each having a structure as described below:
The first line contains two space-separated integers, R and C, indicating the number of rows and columns in the grid G, respectively.
This is followed by R lines, each with a string of C digits, which represent the grid G.
The following line contains two space-separated integers, r and c, indicating the number of rows and columns in the pattern grid P.
This is followed by r lines, each with a string of c digits, which represent the pattern P.

Constraints

$1 \leq T \leq 5$
$1 \leq R, r, C, c \leq 1000$
$1 \leq r \leq R$
$1 \leq c \leq C$

Output Format

Display 'YES' or 'NO', depending on whether (or not) you find that the larger grid G contains the rectangular pattern P. The evaluation will be case sensitive.

Sample Input

```
2
10 10
7283455864
6731158619
8988242643
3830589324

2229505813
5633845374
6473530293
7053106601
0834282956
4607924137
3 4
9505
3845
3530
```

```
----
15 15
400453592126560
114213133098692
474386082879648
522356951189169
887109450487496
252802633388782
502771484966748
075975207693780
511799789562806
404007454272504
549043809916080
962410809534811
445893523733475
768705303214174
650629270887160
2 2
99
99
```

Sample Output

```
YES
NO
```

Explanation

The first test in the input file is:

```
10 10
7283455864
6731158619
8988242643
3830589324
2229505813
5633845374
6473530293
7053106601
0834282956
4607924137
3 4
9505
3845
3530
```

As one may see, the given 2D grid is indeed present in the larger grid, as marked in bold below.

```
7283455864
6731158619
8988242643
3830589324
2229505813
5633845374
6473530293
7053106601
0834282956
4607924137
```

The second test in the input file is:

```
15 15
400453592126560
114213133098692
474386082879648
522356951189169
887109450487496
252802633388782
502771484966748
075975207693780
511799789562806
404007454272504
549043809916080
962410809534811
445893523733475
768705303214174
650629270887160
2 2
99
99
```

The search pattern is:

```
99
99
```

This cannot be found in the larger grid.

Submissions: 25030
Max Score: 30
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)

C#

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

    static void Main(String[] args) {
        int t = Convert.ToInt32(Console.ReadLine());
        for(int a0 = 0; a0 < t; a0++){
            string[] tokens_R = Console.ReadLine().Split(' ');
            int R = Convert.ToInt32(tokens_R[0]);
            int C = Convert.ToInt32(tokens_R[1]);
            string[] G = new string[R];
            for(int G_i = 0; G_i < R; G_i++){
               G[G_i] = Console.ReadLine();
            }
            string[] tokens_r = Console.ReadLine().Split(' ');
            int r = Convert.ToInt32(tokens_r[0]);
            int c = Convert.ToInt32(tokens_r[1]);
            string[] P = new string[r];
            for(int P_i = 0; P_i < r; P_i++){
               P[P_i] = Console.ReadLine();
            }

            bool encontro = false;

            for (int i = 0; i < G.Length && !encontro; i++)
            {
                int indice_col = G[i].IndexOf(P[0]);
                if (indice_col > -1)
                {
                    while (indice_col > -1 && !encontro)
                    {
                        int j = 0;
                        for (j = 0; j < P.Length; j++)
                        {
                           // Console.WriteLine(G[i + j].Substring(indice_col, P[j].Length) + " " + P[j]);
                            if (G[i + j].Substring(indice_col, P[j].Length) != P[j])
                            {
                                break;
                            }
                        }
                        if (j == P.Length)
                        {
                             encontro = true;
                            Console.WriteLine("YES");
                            break;
                        }
                        indice_col = G[i].IndexOf(P[0], indice_col+1);

                       // Console.WriteLine();
                    }
                }
                if (!encontro && i + P.Length >= G.Length)
                {
                    Console.WriteLine("NO");
                    break;
                }
            }


            }
        }
    }

```

Line: 51 Col: 26

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

Next Challenge

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