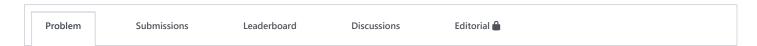


# The Grid Search





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:

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**

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

#### **Test Case Generation**

Each individual test case has been generated by first specifying the size ( $\mathbf{R}$  and  $\mathbf{C}$ ) of the large 2D matrix, and then randomly generating the digits in it. A limited number of digits in the larger matrix may be changed by the problem setter (no more than 5% of the total number of digits in the matrix). So the larger 2D matrix is almost-random. The pattern matrix has been manually-curated by the problem setter.

#### **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

#### 6/23/2016

### **Sample Output**

YES NO

# **Explanation**

The first test in the input file is:

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

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.

in 💆 f

Submissions: 22434
Max Score: 30
Difficulty: Moderate

More

```
Current Buffer (saved locally, editable) & 40
                                                                                          C#
                                                                                                                           Ö
 1 using System;
   using System.Collections.Generic;
   using System.IO;
   using System.Linq;
 5 ▼ class Solution {
 6
 7 ▼
        static void Main(String[] args) {
 8
            int t = Convert.ToInt32(Console.ReadLine());
 9 ▼
            for(int a0 = 0; a0 < t; a0++){
                string[] tokens_R = Console.ReadLine().Split(' ');
10
                int R = Convert.ToInt32(tokens_R[0]);
11
                int C = Convert.ToInt32(tokens_R[1]);
12
13
                string[] G = new string[R];
14 ▼
                for(int G_i = 0; G_i < R; G_i++){
15
                   G[G_i] = Console.ReadLine();
16
                string[] tokens_r = Console.ReadLine().Split(' ');
17
                int r = Convert.ToInt32(tokens_r[0]);
18
19
                int c = Convert.ToInt32(tokens_r[1]);
                string[] P = new string[r];
20
21 ▼
                for(int P_i = 0; P_i < r; P_{i++}){
22
                   P[P_i] = Console.ReadLine();
23
24
            }
25
        }
26
    }
27
                                                                                                                   Line: 1 Col: 1
```

**1** Upload Code as File

☐ Test against custom input

Run Code

Submit Code

Copyright © 2016 HackerRank. All Rights Reserved

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature