The Grid Search

locked

by PRASHANTB1984

Problem | Submissions | Leaderboard | Discussions | Editorial

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

```
1234567890
09876543 21
1111111111
1111111111
2222222222
```

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

```
876543
111111
111111
```

The 2D pattern begins at the second row and the third column of the grid. The pattern is said to be *present* in the grid.

Function Description

Complete the *gridSearch* function in the editor below. It should return `YES` if the pattern exists in the grid, or `NO` otherwise.

gridSearch has the following parameter(s):

- *G*: the grid to search, an array of strings
- *P*: the pattern to search for, an array of strings

Input Format

The first line contains an integer t, the number of test cases.

Each of the t test cases is represented as follows:
The first line contains two space-separated integers R and C, indicating the number of rows and columns in the grid G.
This is followed by R lines, each with a string of C digits representing 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 representing 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 P is present in G.

Sample Input

```
2
10 10
7283455864
6731158619
8988242643
```

```
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 pattern is present in the larger grid, as marked in bold below.

```
7283455864
6731158619
8988242643
3830589324
222**9505**813
563**3845**374
647**3530**293
7053106601
0834282956
4607924137
```

The second test in the input file is:

```
15 15
400453592126560
114213133098692
474386082879648
522356951189169
887109450487496
252802633388782
```

```
650629270887160
2 2
99
99
```

The search pattern is:

```
99
99
```

This cannot be found in the larger grid.

Submissions: 126
Max Score: 10

Rate This Challeng

More

Current Buffer (saved locally, editable)

C

```c
#include <assert.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char** split_string(char*);

// Complete the gridSearch function below.

// Please either make the string static or allocate on the heap. For example,
// static char str[] = "hello world";
// return str;
//
// OR
//
// char* str = "hello world";
// return str;
//
char* gridSearch(int G_count, char** G, int P_count, char** P) {


}

int main()
{
    FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

    char* t_endptr;
    char* t_str = readline();
    int t = strtol(t_str, &t_endptr, 10);

    if (t_endptr == t_str || *t_endptr != '\0') { exit(EXIT_FAILURE); }

    for (int t_itr = 0; t_itr < t; t_itr++) {
        char** RC = split_string(readline());

        char* R_endptr;
        char* R_str = RC[0];
        int R = strtol(R_str, &R_endptr, 10);

        if (R_endptr == R_str || *R_endptr != '\0') { exit(EXIT_FAILURE); }

        char* C_endptr;
        char* C_str = RC[1];
        int C = strtol(C_str, &C_endptr, 10);

        if (C_endptr == C_str || *C_endptr != '\0') { exit(EXIT_FAILURE); }

        char** G = malloc(R * sizeof(char*));
```