Problem: 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:

1234567890

09**876543**21

11**111111**11

11**111111**11

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

**Input Format**

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

**Constraints**

   
 

**Output Format**

Display 'YES' or 'NO', depending on whether (or not) you find that the larger grid  contains the rectangular pattern . 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

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

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.

Solution

public class Solution

{

public static int check(String grid[], String pgrid[],int row, int pcols, int cols)

{

int flag=0, index=0, end=0;

for(index=0; index+pcols<=cols && flag!=1; index++)

{

int counter=0;

for(int i=0; i<pgrid.length && flag!=1 && row+i<grid.length; i++)

{

if(grid[row+i].substring(index,index+pcols).equals(pgrid[i]))

{

counter+=1;

if(counter==pgrid.length)

{

flag=1;

return 1;

}

}

}

}

return 0;

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

int testCases, rows, cols, prows, pcols;

testCases=sc.nextInt();

for(int k=0; k<testCases; k++)

{

rows=sc.nextInt(); cols=sc.nextInt(); //---Input Grid

String grid[]=new String[rows];

for(int i=0; i<rows; i++)

{grid[i]=sc.next();}

prows=sc.nextInt(); pcols=sc.nextInt(); //---Input pGrid

String pgrid[]=new String[prows];

for(int i=0; i<prows; i++)

{pgrid[i]=sc.next();}

//---Processing the Grid

int counter=0;

for(int i=0; i<rows; i++)

{

if( grid[i].contains(pgrid[0]) ) //---Prospective case found

{ counter+=check(grid,pgrid,i,pcols,cols);

if(counter==1){break;}

}

}

if(counter>0)

{System.out.println("YES");}

else

{System.out.println("NO");}

} //End of testCase loop

}

}

* `’Anshul AgGarwal