## 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

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  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 .

**Test Case Generation**   
Each individual test case has been generated by first specifying the size ( and ) 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  contains the rectangular pattern. The evaluation will be case sensitive.

Solution :

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int t = in.nextInt();

for(int a0 = 0; a0 < t; a0++){

int R = in.nextInt();

int C = in.nextInt();

String G[] = new String[R];

for(int G\_i=0; G\_i < R; G\_i++){

G[G\_i] = in.next();

}

int r = in.nextInt();

int c = in.nextInt();

String P[] = new String[r];

for(int P\_i=0; P\_i < r; P\_i++)

{

P[P\_i] = in.next();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

boolean flag = false;

int count=0,end=0,start=0;

for(int m=0;m<R;m++)

{

if(G[m].contains(P[0]))

{

for(int k=0;k<=R-r;k++)

{

end=c;

start=0;

while(end<=C)

{

if(G[k].substring(start,end).equals(P[0])){

for(int i=1;i<r;i++)

{

if(G[k+i].substring(start,end).equals(P[i]))

count++;

if(count==r-1)flag=true;

}

}

count=0;

end++;

start++;

}

}

if(flag){

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

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

break;

}

}

}

}

}