**Word Search**

**Medium**

Given a 2D board of letters and a word. Check if the word exists in the board. The word can be constructed from letters of adjacent cells only. ie - horizontal or vertical neighbors. The same letter cell can not be used more than once.

**Example 1:**

**Input:** board = {{a,g,b,c},{q,e,e,l},{g,b,k,s}},

word = "geeks"

**Output:** 1

**Explanation:** The board is-

a g b c

q e e l

g b k s

The letters which are used to make the

"geeks" are colored.

**Example 2:**

**Input:** board = {{a,b,c,e},{s,f,c,s},{a,d,e,e}},

word = "sabfs"

**Output:** 0

**Explanation:** The board is-

a b c e

s f c s

a d e e

Same letter can not be used twice hence ans is 0

**Expected Time Complexity:**O(N \* M \* 4L) where N = No. of rows in board, M = No. of columns in board, L = Length of word  
**Expected Space Compelxity:**O(L), L is length of word.

**Constraints:**  
1 ≤ N, M ≤ 100  
1 ≤ L ≤ N\*M

**Company Tags**

[**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Intuit**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Intuit) [**Apple**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Apple)

//{ Driver Code Starts

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class CodingMaxima

{

public static void main(String[] args) throws IOException

{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int T = Integer.parseInt(br.readLine().trim());

while(T-->0)

{

String[] S = br.readLine().trim().split(" ");

int n = Integer.parseInt(S[0]);

int m = Integer.parseInt(S[1]);

char [][] board = new char[n][m];

for(int i = 0; i < n; i++){

String[] s = br.readLine().trim().split(" ");

for(int j = 0; j < m; j++){

board[i][j] = s[j].charAt(0);

}

}

String word = br.readLine().trim();

Solution obj = new Solution();

boolean ans = obj.isWordExist(board, word);

if(ans)

System.out.println("1");

else

System.out.println("0");

}

}

}

// } Driver Code Ends

class Solution

{

public boolean isWordExist(char[][] board, String word)

{

int n=board.length;

int m=board[0].length;

for(int i=0;i<n;i++){

for(int j=0;j<m;j++){

if(board[i][j]==word.charAt(0) && dfs(board, word, i, j,n, m, 0))

return true;

}

}

return false;

}

private boolean dfs(char[][] board, String word, int i, int j,int n, int m, int idx){

if(idx==word.length())

return true;

if(i<0 || j<0 || i==n || j==m || board[i][j]=='&' || board[i][j]!=word.charAt(idx))

return false;

char temp=board[i][j];

board[i][j]='&';

boolean x\_positive=dfs(board, word, i+1, j,n, m ,idx+1);

boolean x\_negative=dfs(board, word, i-1, j,n, m, idx+1);

boolean y\_positive=dfs(board, word, i, j+1,n, m, idx+1);

boolean y\_negative=dfs(board, word, i, j-1, n, m,idx+1);

board[i][j]=temp;

return x\_positive || x\_negative || y\_positive || y\_negative;

}

}