**Word Boggle - II**

**Medium**

Given a dictionary of words and an M x N board where every cell has one character. Find all possible different words from the dictionary that can be formed by a sequence of adjacent characters on the board. We can move to any of 8 adjacent characters, but a word should not have multiple instances of the same cell.

**Example 1:**

**Input:**

N = 1

dictionary = {"CAT"}

R = 3, C = 3

board = {{C,A,P},{A,N,D},{T,I,E}}

**Output:**

CAT

**Explanation**:

C A P

A N D

T I E

Words we got is denoted using same color.

**Example 2:**

**Input:**

N = 4

dictionary = {"GEEKS","FOR","QUIZ","GO"}

R = 3, C = 3

board = {{G,I,Z},{U,E,K},{Q,S,E}}

**Output:**

GEEKS QUIZ

**Explanation**:

G I Z

U E K

Q S E

Words we got is denoted using same color.

**Expected Time Complexity:**O(4^(N^2))  
**Expected Auxiliary Space:**O(N^2)

**Constraints:**  
1 ≤ N ≤ 15  
1 ≤ R, C ≤ 50  
1 ≤ length of Word ≤ 60

All words of Dictionary and all characters of board are in uppercase letters from **'A'** to **'Z'**

**Company Tags**

[**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**MakeMyTrip**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MakeMyTrip) [**Directi**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Directi) [**Google**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Google) [**Facebook**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Facebook) [**Yahoo**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Yahoo) [**Nvidia**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Nvidia)

//{ Driver Code Starts

//Initial Template for Java

import java.io.\*;

import java.util.\*;

class CodingMaxima

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

while(t>0)

{

int N = sc.nextInt();

String[] dictionary = new String[N];

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

{

dictionary[i] = sc.next();

}

int R = Integer.parseInt(sc.next());

int C = Integer.parseInt(sc.next());

char board[][] = new char[R][C];

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

{

for(int j=0;j<C;j++)

{

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

}

}

Solution obj = new Solution();

String[] ans = obj.wordBoggle(board, dictionary);

if(ans.length == 0) System.out.println("-1");

else

{

Arrays.sort(ans);

for(int i=0;i<ans.length;i++)

{

System.out.print(ans[i] + " ");

}

System.out.println();

}

t--;

}

}

}

// } Driver Code Ends

class Solution

{

public String[] wordBoggle(char board[][], String[] dictionary)

{

Set<String> ans=new HashSet<>();

int n=board.length;

int m=board[0].length;

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

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

for(int k=0;k<dictionary.length;k++){

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

ans.add(dictionary[k]);

}

}

}

}

String[] array = new String[ans.size()];

ans.toArray(array);

return array;

}

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);

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

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

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

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

board[i][j]=temp;

return (x\_positive || x\_negative || y\_positive || y\_negative || a || b || c || d);

}

}