Search a Word in a 2D Grid of characters

Tuesday, November 30, 2021 11:34 PM

Search a Word in a 2D Grid of characters

- Difficulty Level : Medium
- Last Updated : 22 Jul, 2021

Given a 2D grid of characters and a word, find all occurrences of the given word in the grid. A word can be matched in all 8 directions at any point. Word is said to be found in a direction if all characters match in this direction (not in zig-zag form).

The 8 directions are, Horizontally Left, Horizontally Right, Vertically Up, Vertically Down and 4 Diagonal directions.

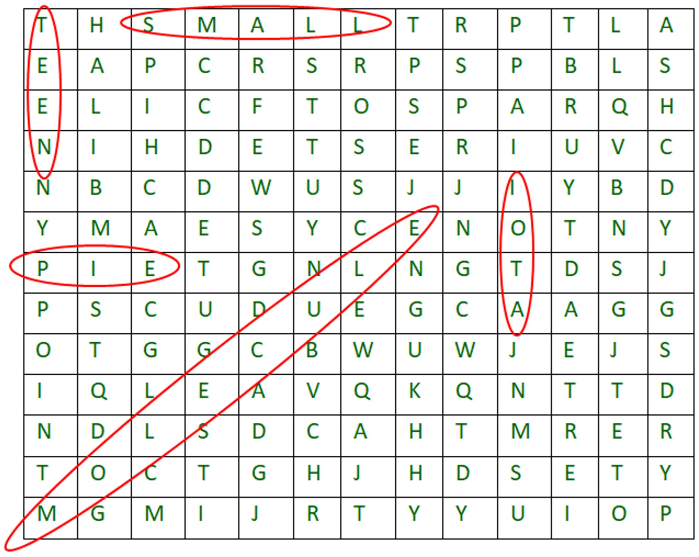
Example:

```
Input:  grid[][] = {"GEEKSFORGEEKS",
                    "GEEKSQUIZGEEK",
                    "IDEQAPRACTICE"};
        word = "GEEKS"
Output: pattern found at 0, 0
        pattern found at 0, 8
        pattern found at 1, 0
Explanation: 'GEEKS' can be found as prefix of
1st 2 rows and suffix of first row
```

```
Input:  grid[][] = {"GEEKSFORGEEKS",
                    "GEEKSQUIZGEEK",
                    "IDEQAPRACTICE"};
        word = "EEE"

Output: pattern found at 0, 2
        pattern found at 0, 10
        pattern found at 2, 2
        pattern found at 2, 12
Explanation: EEE can be found in first row
twice at index 2 and index 10
and in second row at 2 and 12
```

Below diagram shows a bigger grid and presence of different words in it.



T	H	S	M	A	L	L	T	R	P	T	L	A
E	A	P	C	R	S	R	P	S	P	B	L	S
E	L	I	C	F	T	O	S	P	A	R	Q	H
N	I	H	D	E	T	S	E	R	I	U	V	C
N	B	C	D	W	U	S	J	J	I	Y	B	D
Y	M	A	E	S	Y	C	E	N	O	T	N	Y
P	I	E	T	G	N	L	N	G	T	D	S	J
P	S	C	U	D	U	E	G	C	A	A	G	G
O	T	G	G	C	B	W	U	W	J	E	J	S
I	Q	L	E	A	V	Q	K	Q	N	T	T	D
N	D	L	S	D	C	A	H	T	M	R	E	R
T	O	C	T	G	H	J	H	D	S	E	T	Y
M	G	M	I	J	R	T	Y	Y	U	I	O	P

Source: Microsoft Interview Question.

Recommended: Please solve it on “***PRACTICE***” first, before moving on to the solution.

Approach: The idea used here is simple, we check every cell. If cell has first character, then we one by one try all 8 directions from that cell for a match. Implementation is interesting though. We use two arrays x[] and y[] to find next move in all 8 directions.

Below are implementation of the same:

- C++
- Java
- Python3
- C#
- Javascript

```java
// Java program to search
// a word in a 2D grid
import java.io.*;
import java.util.*;

class GFG {

    // Rows and columns in the given grid
    static int R, C;

    // For searching in all 8 direction
    static int[] x = { -1, -1, -1, 0, 0, 1, 1, 1 };
    static int[] y = { -1, 0, 1, -1, 1, -1, 0, 1 };
```

```java
// This function searches in all
// 8-direction from point
// (row, col) in grid[][]
static boolean search2D(char[][] grid, int row,
                        int col, String word)
{
    // If first character of word
    // doesn't match with
    // given starting point in grid.
    if (grid[row][col] != word.charAt(0))
        return false;

    int len = word.length();

    // Search word in all 8 directions
    // starting from (row, col)
    for (int dir = 0; dir < 8; dir++) {
        // Initialize starting point
        // for current direction
        int k, rd = row + x[dir], cd = col + y[dir];

        // First character is already checked,
        // match remaining characters
        for (k = 1; k < len; k++) {
            // If out of bound break
            if (rd >= R || rd < 0 || cd >= C || cd < 0)
                break;

            // If not matched, break
            if (grid[rd][cd] != word.charAt(k))
                break;

            // Moving in particular direction
            rd += x[dir];
            cd += y[dir];
        }

        // If all character matched,
        // then value of must
        // be equal to length of word
        if (k == len)
            return true;
    }
    return false;
}

// Searches given word in a given
// matrix in all 8 directions
static void patternSearch(
    char[][] grid,
    String word)
{
    // Consider every point as starting
    // point and search given word
    for (int row = 0; row < R; row++) {
            for (int col = 0; col < C; col++) {
                if (search2D(grid, row, col, word))
                    System.out.println(
                        "pattern found at " + row + ", " + col);
            }
        }
    }

    // Driver code
    public static void main(String args[])
    {
        R = 3;
        C = 13;
        char[][] grid = { { 'G', 'E', 'E', 'K', 'S', 'F', 'O', 'R', 'G', 'E',
'E', 'K', 'S' },
                          { 'G', 'E', 'E', 'K', 'S', 'Q', 'U', 'I', 'Z', 'G',
'E', 'E', 'K' },
                          { 'I', 'D', 'E', 'Q', 'A', 'P', 'R', 'A', 'C', 'T',
'I', 'C', 'E' } };
        patternSearch(grid, "GEEKS");
        System.out.println();
        patternSearch(grid, "EEE");
    }
}

// This code is contributed by rachana soma
```

Output:

```
pattern found at 0, 0
pattern found at 0, 8
pattern found at 1, 0
pattern found at 0, 2
pattern found at 0, 10
pattern found at 2, 2
pattern found at 2, 12
```

Complexity Analysis:

- **Time complexity:** O(R*C*8*len(str)).
  All the cells will be visited and traversed in all 8 directions, where R and C is side of matrix so time complexity is O(R*C).
- **Auxiliary Space:** O(1).
  As no extra space is needed.

From <https://www.geeksforgeeks.org/search-a-word-in-a-2d-grid-of-characters/>

```
packagedsaProblems;

publicclassWord_in_grid{
publicstaticvoidsearch2Dword(Stringword,chargrid[][])
{
```

```
intm=grid.length;
intn=grid[0].length;
for(inti=0;i<m;i++)
{
for(intj=0;j<n;j++)
{
if(search2D(i,j,word,grid))
System.out.println(
"patternfoundat"+i+","+j);
}
}
}
publicstaticbooleansearch2D(inti,intj,Stringword,chargrid[][])
{
if(grid[i][j]!=word.charAt(0))
returnfalse;
intlen=word.length();

intx[]={-1,-1,-1,0,0,1,1,1};
inty[]={-1,0,1,-1,1,-1,0,1};
for(intdir=0;dir<8;dir++)
{
intnewrow=i+x[dir];
intnewcol=j+y[dir];
intk;


for(k=1;k<word.length();k++)
{
if(newrow>=grid.length||newrow<0||newcol>=grid[0].length||newcol<0)
break;;
if(word.charAt(k)!=grid[newrow][newcol])
break;
newrow+=x[dir];
newcol+=y[dir];

}
if(k==len)
returntrue;

}
returnfalse;

}
```

```
publicstaticvoidmain(String[]args){
//char[][]grid={
//{'a','b','c'},
//{'d','e','f'},
//{'g','h','i'}
//};
char[][]grid={{'G','E','E','K','S','F','O','R','G','E','E','K','S'},
{'G','E','E','K','S','Q','U','I','Z','G','E','E','K'},
{'I','D','E','Q','A','P','R','A','C','T','I','C','E'}};
Stringword="GEEKS";
search2Dword(word,grid);



}

}
```