

Search a Word in a 2D Grid of characters

Given a 2D grid of characters and a word, find all occurrences of given word in 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 and 4 Diagonal directions.

Example:

```
Input: grid[][] = {"GEEKSFORGEESK",
                  "GEEKSQUIZGEEK",
                  "IDEQAPRACTICE"};
word = "GEEKS"
```

```
Output: pattern found at 0, 0
        pattern found at 0, 8
        pattern found at 1, 0
```

```
Input: grid[][] = {"GEEKSFORGEESK",
                  "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
```

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

Source: Microsoft Interview Question

We strongly recommend you to minimize your browser and try this yourself first.

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 is C++ implementation of the same.

```
// C++ programs to search a word in a 2D grid
#include<bits/stdc++.h>
using namespace std;

// Rows and columns in given grid
#define R 3
#define C 14

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

// This function searches in all 8-direction from point
// (row, col) in grid[][]
bool search2D(char grid[R][C], 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[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[k])
                break;

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

        // If all character matched, then value of must
```

Popular Posts

- Top 10 Algorithms and Data Structures for Competitive Programming
- Top 10 algorithms in Interview Questions
- How to begin with Competitive Programming?
- Reflection in Java
- Memory Layout of C Programs
- Push Relabel Algorithm
- Heavy Light Decomposition
- Sorted Linked List to Balanced BST
- Generics in Java
- Aho-Corasick Algorithm for Pattern Searching
- Binary Search , QuickSort , MergeSort , HeapSort

- Common Interview Puzzles
- Interview Experiences
- Advanced Data Structures
- Design Patterns
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Geometric Algorithms
- Searching
- Sorting
- Hashing
- Analysis of Algorithms

```

// be equal to length of word
if (k == len)
    return true;
}
return false;
}

// Searches given word in a given matrix in all 8 directions
void patternSearch(char grid[R][C], 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))
                cout << "pattern found at " << row << ", "
                    << col << endl;
}

// Driver program
int main()
{
    char grid[R][C] = {"GEEKSFORGEEKS",
                       "GEEKSQUIZGEEK",
                       "IDEQAPRACTICE"
                      };

    patternSearch(grid, "GEEKS");
    cout << endl;
    patternSearch(grid, "EEE");
    return 0;
}

```

[Run on IDE](#)

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

```

Exercise: The above solution only print locations of word. Extend it to print the direction where word is present.

See [this](#) for solution of exercise.

This article is contributed by [Utkarsh Trivedi](#). Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

[Leave a comment](#) Category: [Pattern Searching](#)

Related Posts:

- Find all occurrences of a given word in a matrix
- Aho-Corasick Algorithm for Pattern Searching
- kasai's Algorithm for Construction of LCP array from Suffix Array
- Z algorithm (Linear time pattern searching Algorithm)
- Suffix Tree Application 6 – Longest Palindromic Substring
- Manacher's Algorithm – Linear Time Longest Palindromic Substring – Part 4
- Manacher's Algorithm – Linear Time Longest Palindromic Substring – Part 3
- Manacher's Algorithm – Linear Time Longest Palindromic Substring – Part 2

Previous post in category

Next post in category

(Login to Rate and Mark)

2.8


Average Difficulty : 2.8/5.0
Based on 8 vote(s)

[Add to TODO List](#)
[Mark as DONE](#)

Writing code in comment? Please use [code.geeksforgeeks.org](#), generate link and share the link here.

- Mathematical Algorithms
- Randomized Algorithms
- Recursion

Jumpstart **Raspberry Pi** projects with *Cayenne*



Accelerate your maker projects with the first drag-and-drop IoT builder.

Get it Free

Tags

[Adobe](#) [Advance Data Structures](#) [Advanced Data Structures](#) [Amazon](#) [array](#) [Backtracking](#) [Bit Magic](#) [C++](#) [CN](#) [c puzzle](#) [D-E-Shaw](#) [DBMS](#) [design](#) [Divide and Conquer](#) [Dynamic Programming](#) [Flipkart](#) [GATE](#) [GATE-CS-C-Language](#) [GATE-CS-DS-&Algo](#) [GATE-CS-Older](#) [GFacts](#) [Goldman Sachs](#) [Google](#) [Graph](#) [Greedy Algorithm](#) [Hashing](#) [Interview Experience](#) [Java](#) [MAQ](#) [Software](#) [MathematicalAlgo](#) [Matrix](#) [Microsoft](#) [Morgan Stanley](#) [Operating systems](#) [Oracle](#) [Pattern Searching](#) [puzzle](#) [Python](#) [Recursion](#) [Samsung](#) [SAP Labs](#) [SnapDeal](#) [stack](#) [Stack-Queue](#) [STL](#)

Subscribe and Never Miss an Article

Email Address

Subscribe

Recent Comments

All Categories

Select Category

- GeeksQuiz
- GeeksforGeeksIDE
- GeeksforGeeks Practice
- Data Structures
- Algorithms
- C Programming
- C++ Programming
- Java Programming
- Books
- Interview Experiences
- GATE CS
- GATE CS Forum
- Android App