[LeetCode](https://leetcode.com/problems/word-search/?envType=daily-question&envId=2024-04-03)

<https://github.com/AdityaKonda6/-50DaysOfCoding>

<https://leetcode.com/problems/word-search/?envType=daily-question&envId=2024-04-03>

<https://www.linkedin.com/in/aditya-adi-konda/>

Day 44 of [#50dayscodingchallenge](https://www.linkedin.com/feed/hashtag/?keywords=50dayscodingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633):  
[#leetcode](https://www.linkedin.com/feed/hashtag/?keywords=leetcode&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodechallenge](https://www.linkedin.com/feed/hashtag/?keywords=leetcodechallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodestreak](https://www.linkedin.com/feed/hashtag/?keywords=leetcodestreak&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode2024](https://www.linkedin.com/feed/hashtag/?keywords=leetcode2024&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode50day](https://www.linkedin.com/feed/hashtag/?keywords=leetcode50day&highlightedUpdateUrns=urn:li:activity:7166316239483461633)  
   
Ventured further into my coding journey today, tackling the engaging LeetCode Problem "Successfully solved LeetCode Problem ���"

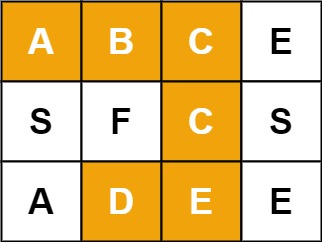
“79. Word Search.”

   
✨ Task: Given an m x n grid of characters board and a string word, return true if word exists in the grid.

The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.

Examples:

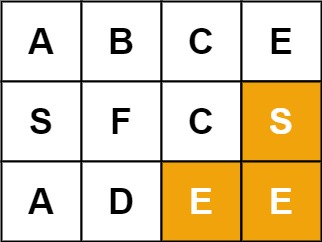
Example 1:



Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"

Output: true

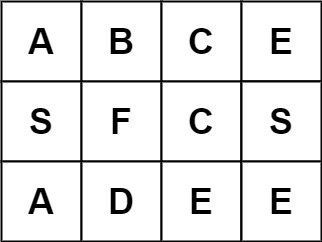
Example 2:



Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"

Output: true

Example 3:



Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"

Output: false

Let's Connect:

If you find this problem intriguing or have insights to share, let's connect! I'm passionate about problem-solving, algorithmic thinking, and collaborative learning. Feel free to comment or reach out for engaging discussions and knowledge exchange.Unravel the mystery using your coding skills!

[#CodingChallenge](https://www.linkedin.com/feed/hashtag/?keywords=codingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#Algorithm](https://www.linkedin.com/feed/hashtag/?keywords=algorithm&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#LinkedInPost](https://www.linkedin.com/feed/hashtag/?keywords=linkedinpost&highlightedUpdateUrns=urn:li:activity:7166316239483461633) #Algorithm #Optimization #DataStructures #CodingChallenge  
  
Excited about the progress and challenges ahead!  
   
Make Sure You Follow My GitHub For Solutions: <https://github.com/AdityaKonda6/-50DaysOfCoding>  
  
  
Happy coding!

**Solution:-**

class Solution {

  public boolean exist(char[][] board, String word) {

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

      for (int j = 0; j < board[0].length; ++j)

        if (dfs(board, word, i, j, 0))

          return true;

    return false;

  }

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

    if (i < 0 || i == board.length || j < 0 || j == board[0].length)

      return false;

    if (board[i][j] != word.charAt(s) || board[i][j] == '\*')

      return false;

    if (s == word.length() - 1)

      return true;

    final char cache = board[i][j];

    board[i][j] = '\*';

    final boolean isExist = dfs(board, word, i + 1, j, s + 1) || //

                            dfs(board, word, i - 1, j, s + 1) || //

                            dfs(board, word, i, j + 1, s + 1) || //

                            dfs(board, word, i, j - 1, s + 1);

    board[i][j] = cache;

    return isExist;

  }

}

