Documentation for the Word Search Solution

Problem Statement

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.

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

Constraints:

- m == board.length
- n == board[i].length
- 1 <= m, n <= 6
- 1 <= word.length <= 15
- board and word consist of only lowercase and uppercase English letters.

Followup:

- Could you use search pruning to make your solution faster with a larger board?

Solution

The provided solution uses DepthFirst Search (DFS) to check if the word exists in the given board. The algorithm explores all possible paths in the grid to find the word.

Explanation

1. **Dimensions of the Board:** The dimensions of the board are determined using m, n = len(board), len(board[0]).

2. **DFS Helper Function:** The dfs function is used to perform a depthfirst search from a given cell (x, y) and checks if the current index of the word can be found starting from this cell.

- If index equals the length of the word, it means all characters have been matched, and True is returned.
- The function checks if the current cell is out of bounds or does not match the current character of the word. If so, False is returned.
- The cell is temporarily marked as visited by setting board[x][y] to '#'.
- The function recursively checks all four neighboring cells (up, down, left, right).
- After exploring, the cell is restored to its original value.

3. **Iterating Over the Board:** The main function iterates over each cell in the board. If the cell's character matches the first character of the word, it starts the DFS search from that cell.

4. **Returning the Result:** If any path matches the word, True is returned. If no paths match, False is returned.

Complexity

Time Complexity: O(m * n * 4^L) in the worst case, where L is the length of the word.

Space Complexity: O(L) for the recursion stack, where L is the length of the word.