

JOB

leetcode.com

Mail - Sikun Chen

github push folder t...

Sikun275 (SikunC)

(15) Sikun Chen | Lin...

New & Trending Co...

Amazon - LeetCode

Word Search - Leet...

Start Page

Amazon

Description

Editorial

Solutions

Submissions

79. Word Search

Medium Topics Companies

Given an $m \times n$ 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:

A	B	C	E
S	F	C	S
A	D	E	E

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

Example 2:

A	B	C	E
S	F	C	S
A	D	E	E

16.8K 279 307 Online

Code

Python3 Auto

```

1 class Solution:
2     def exist(self, board: List[List[str]], word: str) -> bool:
3         m, n = len(board), len(board[0])
4         def findBegin(c):
5             res = []
6             for i in range(m):
7                 for j in range(n):
8                     if board[i][j] == c:
9                         res.append((i,j))
10
11         return res
12     begin = findBegin(word[0])
13     if not begin:
14         return False
15     directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
16     for r, c in begin:
17         queue = deque()
18         pos = 1
19         queue.append((r, c))
20         while queue:
21             rr, cc = queue.popleft()
22             for dr, dc in directions:
23                 nr, nc = rr + dr, cc + dc
24                 if 0 <= nr < m and 0 <= nc < n:
25
26                     if board[nr][nc] == word[pos]:
27                         queue.append((nr, nc))
28                         pos += 1
29                         if pos == len(word) - 1:
30                             return True
31             else:
32                 break
33     return False
34
35
36
37
38

```

Ln 32, Col 34 Saved

Run Submit

Testcase Test Result

first version is not correct

All Submissions

Accepted 88 / 88 testcases passed

Editorial Solution

A... submitted at Jun 18, 2025 14:42

Runtime

2947 ms | Beats 90.25%

Analyze Complexity

Memory

17.85 MB | Beats 65.16%

Python3 Auto

```

1 class Solution:
2     def exist(self, board: List[List[str]], word: str) -> bool:
3         m, n = len(board), len(board[0])
4         l = len(word)
5
6         def dfs(r, c, i):
7             if i == l:
8                 return True
9
10            if r < 0 or r >= m or c < 0 or c >= n or board[r][c] != word[i]:
11                return False
12
13            temp, board[r][c] = board[r][c], '#'
14
15            res = (dfs(r+1, c, i+1) or
16                  dfs(r-1, c, i+1) or
17                  dfs(r, c+1, i+1) or
18                  dfs(r, c-1, i+1))
19            board[r][c] = temp
20            return res
21
22        for i in range(m):
23            for j in range(n):
24                if board[i][j] == word[0] and dfs(i, j, 0):
25                    return True
26
27        return False
28

```

the way to handle path, is not backtrack, but use or to make sure at least one correct answer