STEP 1

	1	2	3	4
1	A	B	C	E
2	S	F	C	S
3	A	D	E	E

Search word = ABCCED

Step 1:

Start vertically and horizontally to search

A ✓	B ✓	C ✓	E ✗
S ✗			
A ✗			

Cells 1:1, 1:2, 1:3 match search word. Cell 1:4 does not.

Step 2:

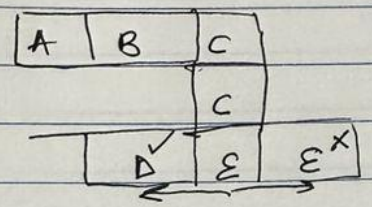
Continue search from the last cell with a positive match

A	B	C
		C ✓
		E ✓

Cells 2:3, 3:3 match search word

Step 3:

Continue search from the last cell with a positive match



Was the search word found? TRUE

STEP 2

backtracking .py - week3 assy - Visual Studio Code

.py | stairclimb.py | BF.py | backtracking .py | dijkstra.py | Kruskalconcities.py

backtracking .py > Solution > getWords

```python
class Solution(object):
    def exist(self, board, word) -> bool:
        """
        :type board: List[List[str]]
        :type word: str
        :rtype: bool
        """
        visited: dict = {}

        for i in range(len(board)):
            for j in range(len(board[0])):
                if self.getWords(board, word, i, j, visited):
                    return True

        return False

    def getWords(self, board, word, i, j, visited, pos = 0) -> Any | bool:
        if pos == len(word):
            return True

        if i < 0 or i == len(board) or j < 0 or j == len(board[0]) or visited.get((
            return False

        visited[(i, j)] = True
        res: Any = self.getWords(board, word, i, j + 1, visited, pos + 1) \
                or self.getWords(board, word, i, j - 1, visited, pos + 1) \
                or self.getWords(board, word, i + 1, j, visited, pos + 1) \
                or self.getWords(board, word, i - 1, j, visited, pos + 1)
        visited[(i, j)] = False

        return res

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

ob1: Solution = Solution()
print("Leetcode's first case: ", ob1.exist(board
 = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"))

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


Ln 21, Col 68 | Spaces: 4 | UTF-8 | CRLF | Python | 3.10.10 64-bit (microsoft store)

54°F | ENG | 1:37 PM 3/8/2023

backtracking .py - week3 assy - Visual Studio Code

.py | stairclimb.py | BF.py | backtracking .py | dijkstra.py | Kruskalconcities.py

backtracking .py > Solution > getWords

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

ob1: Solution = Solution()
print("Leetcode's first case: ", ob1.exist(board
 = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"))

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

ob1: Solution = Solution()
print("Leetcode's second case: ", ob1.exist(board
 = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"))

#Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "AB
#Output: false

ob1: Solution = Solution()
print("Leetcode's third case: ", ob1.exist(board
 = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"))
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | Python

```
PS C:\Users\melan\Desktop\Algorithms\week3 assy> cd "c:/Users/melan/Desktop/Algorithms/week3 assy"
PS C:\Users\melan\Desktop\Algorithms\week3 assy> & C:/Users/melan/AppData/Local/Microsoft/WindowsApps/python3
.10.exe "c:/Users/melan/Desktop/Algorithms/week3 assy/backtracking .py"
Leetcode's first case:  True
Leetcode's second case:  True
Leetcode's third case:  False
PS C:\Users\melan\Desktop\Algorithms\week3 assy>
```


⚠ Error running mypy in c:\Users\melan\Desktop\Algorithms\week3 assy...

⊗ 0 ⚠ 0 | Ln 21, Col 68 | Spaces: 4 | UTF-8 | CRLF | Python | 3.10.10 64-bit (microsoft store)

54°F | ENG | 1:37 PM 3/8/2023

CODE

```
class Solution(object):
    def exist(self, board, word):
        """
        :type board: List[List[str]]
        :type word: str
        :rtype: bool
        """
        visited = {}

        for i in range(len(board)):
            for j in range(len(board[0])):
                if self.getWords(board, word, i, j, visited):
                    return True

        return False

    def getWords(self, board, word, i, j, visited, pos = 0):
        if pos == len(word):
            return True

        if i < 0 or i == len(board) or j < 0 or j == len(board[0]) or visited.get((i, j)) or word[pos]
!= board[i][j]:
            return False

        visited[(i, j)] = True
        res = self.getWords(board, word, i, j + 1, visited, pos + 1) \
                or self.getWords(board, word, i, j - 1, visited, pos + 1) \
                or self.getWords(board, word, i + 1, j, visited, pos + 1) \
                or self.getWords(board, word, i - 1, j, visited, pos + 1)
        visited[(i, j)] = False

        return res

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

ob1 = Solution()
print("Leetcode's first case: ", ob1.exist(board
 = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"))

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

```
ob1 = Solution()
print("Leetcode's second case: ", ob1.exist(board
 = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"))

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

ob1 = Solution()
print("Leetcode's third case: ", ob1.exist(board
 = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"))
```