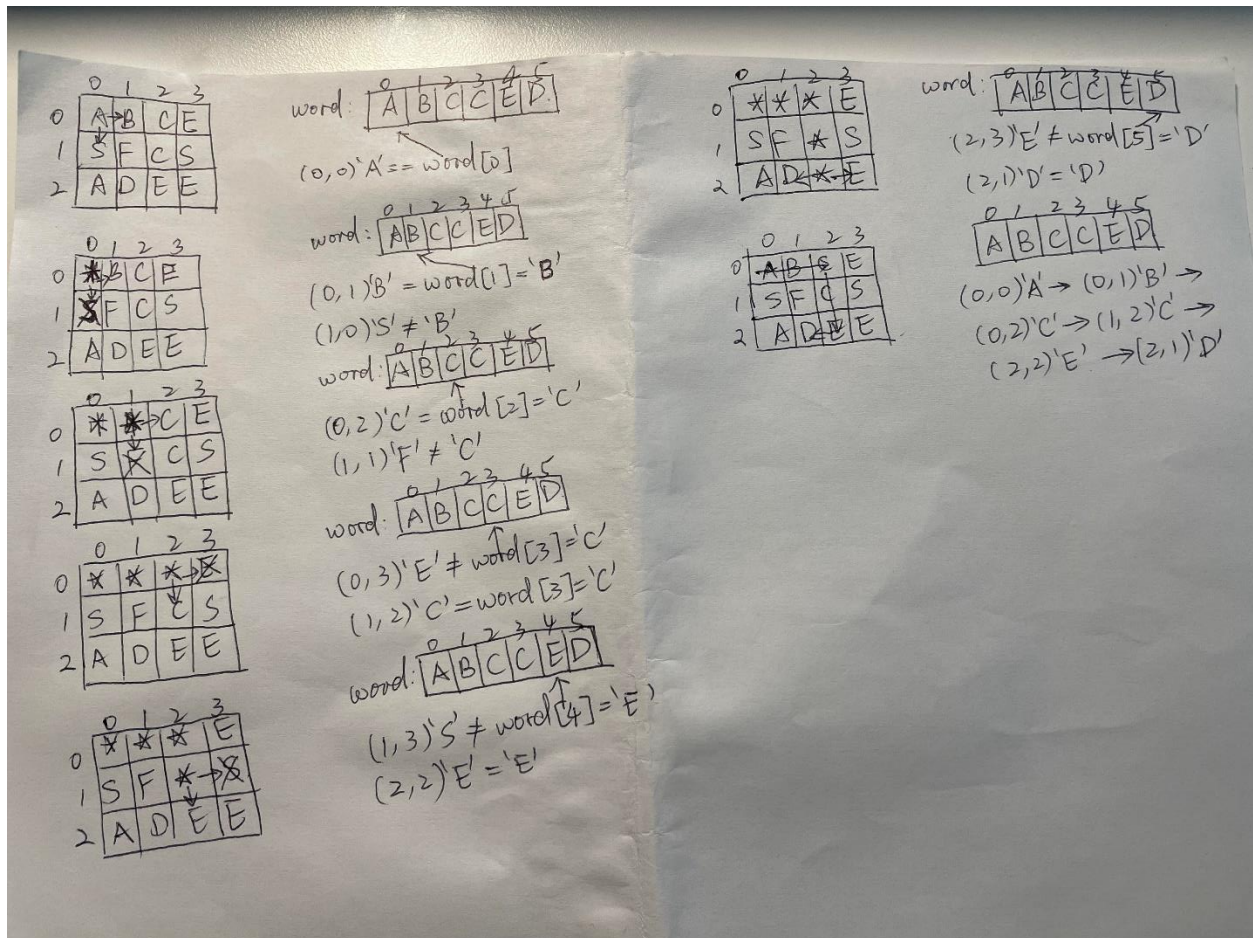



Step 1: Following the "GRID Structure" examples shown on Word Search to manually solve the Example 1 of the problem 79. Word Search (local copy) - 39.8 Medium



Step 2: Ask ChatGPT: "How to use Backtracking approach to implement a Python code to solve the LeetCode question: '79. Word Search', with the test data: 'Input: board =
[["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED", Output: true'"

To solve the LeetCode question '79. Word Search' using a backtracking approach, we can recursively explore the board to find the given word. Here's the Python code that implements the backtracking algorithm for the given test data:

python

 Copy code

```
from typing import List

def exist(board: List[List[str]], word: str) -> bool:
    # Helper function for backtracking
    def backtrack(row: int, col: int, index: int) -> bool:
        # Base case: If all characters in the word are found, return True
        if index == len(word):
            return True

        # Check if the current position is out of bounds or doesn't match the character
        if row < 0 or row >= rows or col < 0 or col >= cols or board[row][col] != word[index]:
            return False

        # Mark the current position as visited
        board[row][col] = "#"

        # Explore the neighboring cells in all four directions
        for dx, dy in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
            if backtrack(row + dx, col + dy, index + 1):
                return True

        # Restore the original character
        board[row][col] = word[index]

    return backtrack(0, 0, 0)
```

```

        # Mark the current position as unvisited (backtrack)
        board[row][col] = word[index]

        return False

    # Get the dimensions of the board
    rows = len(board)
    cols = len(board[0])

    # Iterate over each cell in the board and start backtracking if the char
    for i in range(rows):
        for j in range(cols):
            if backtrack(i, j, 0):
                return True

    return False

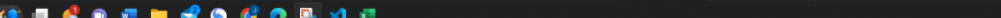
# Test data
board = [
    ["A", "B", "C", "E"],
    ["S", "F", "C", "S"],
    ["A", "D", "E", "E"]
]
word = "ABCCED"

# Call the function and print the result
print(exist(board, word))

```

When you run this code, it will output `True`, which is the expected result for the given test data. The code uses a backtracking approach to explore the board and find the given word. If the word can be formed by adjacent cells (horizontally or vertically) in the board, the code returns `True`; otherwise, it returns `False`.

Step 3: Implement the Python code created by ChatGPT



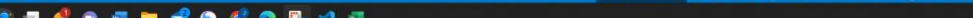
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS D:\HSCS\CS455 Algorithm\w6> & C:\Users\odody\AppData\Local\Programs\Python\Python311\python.exe "d:/HSCS/CS455 Algorithm/w6/test.py"
True
PS D:\HSCS\CS455 Algorithm\w6>
```


Column Selection Ln 126, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live

Search

6:21 PM 7/2/2023



The screenshot shows the terminal output of the command `python.exe "d:\MSCS\CS455 Algorithm\w6\test.py"`. The output is `True`. The terminal window is titled "TERMINAL" and shows the command prompt `PS D:\MSCS\CS455 Algorithm\w6>`. The status bar at the bottom indicates the file is `test.py` in the `d:\MSCS\CS455 Algorithm\w6` directory, using Python 3.11.3 64-bit.



The screenshot shows a Windows terminal window with the following content:

```
PS D:\MSCS\CS455 Algorithm\w6> & C:\Users\odody\AppData\Local\Programs\Python\Python311\python.exe "d:\MSCS\CS455 Algorithm\w6\test.py"
False
PS D:\MSCS\CS455 Algorithm\w6>
```

The terminal window has a title bar with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The status bar at the bottom indicates 'Column Selection', 'Ln 122, Col 13', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', '3.11.3 64-bit', 'Go Live', and a clock showing '8:25 PM 7/2/2023'.