Step 1: Study 490. The Maze - (local copy)

Step 2: Manual process to demonstrate concepts

Robot   Clear Route (Street, Highway)   Unclear Route (Hotel, Hospital)

Without Wheel

(Legged Robot) Step 1.1: Tree

Following the examples shown on Depth-First Traversal to manually solve the problem

Maze example

With Wheel

(Self-driving Car)

Step	Visited Array	Stack	Print
1	[A]	[B, C]	[A]
2	[A, B]	[D, C]	[A, B]
3	[A, B, D]	[F, C]	[A, B, D]
4	[A, B, D, F]	[E, C]	[A, B, D, F]
5	[A, B, D, F, E]	[C]	[A, B, D, F, E]
6	[A, B, D, F, E, C]	[]	[A, B, D, F, E, C]

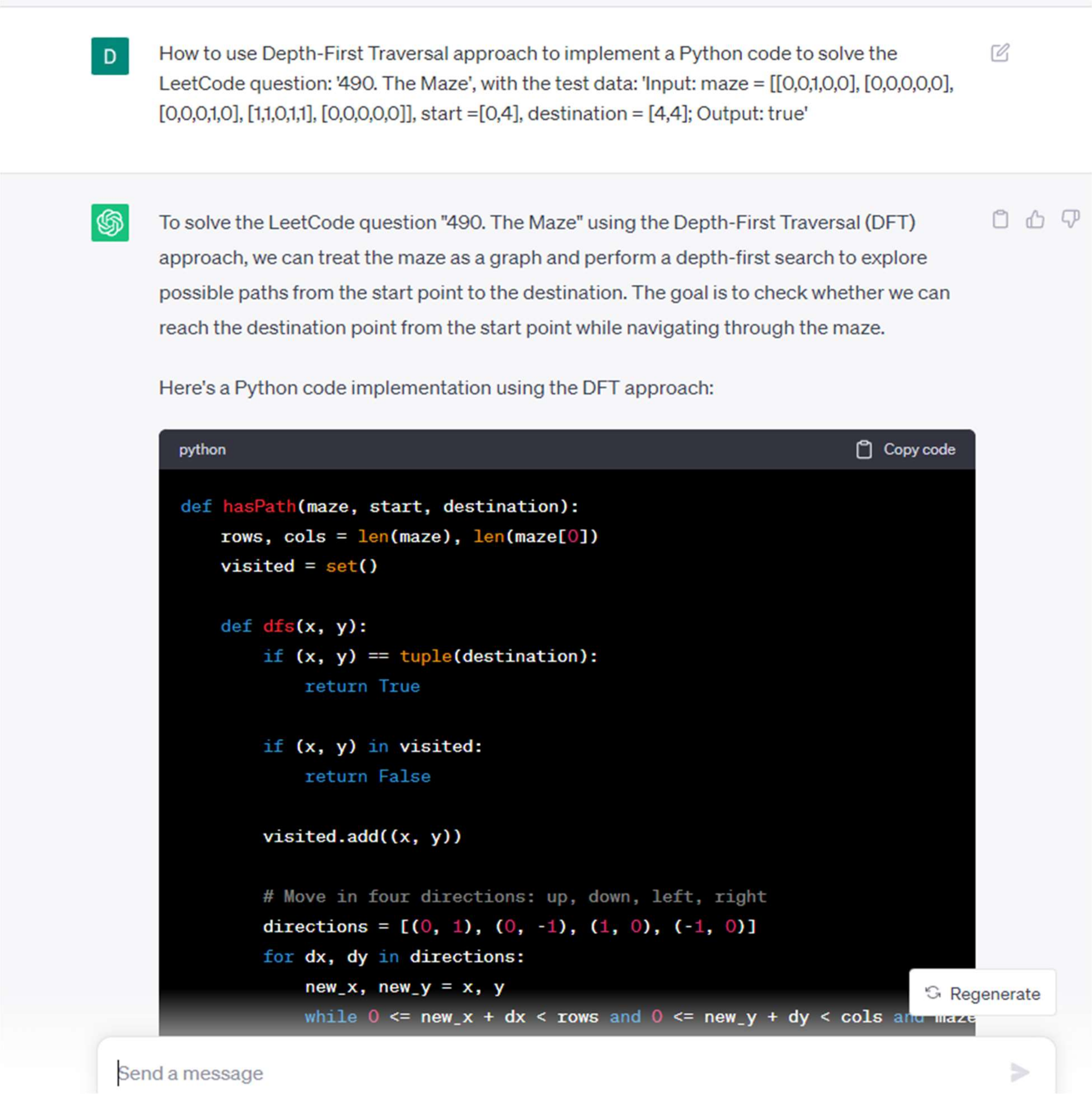
Step 1.2: Matrix

Following the examples shown on Depth-First Traversal to manually solve the problem

Maze example -- assuming the ball can go through the empty spaces by rolling.

Step	Visited Array	Queue	Print
1	[A]	[B, C]	[A]
2	[A, B]	[C, D]	[A, B]
3	[A, B, C]	[D, E]	[A, B, C]
4	[A, B, C, D]	[E, F]	[A, B, C, D]
5	[A, B, C, D, E]	[F]	[A, B, C, D, E]
6	[A, B, C, D, E, F]	[]	[A, B, C, D, E, F]

Step 3: Ask ChatGPT: "How to use Depth-First Traversal approach to implement a Python code to solve the LeetCode question: '490. The Maze', with the test data: 'Input: maze = [[0,0,1,0,0], [0,0,0,0,0], [0,0,0,1,0], [1,1,0,1,1], [0,0,0,0,0]], start =[0,4], destination = [4,4]; Output: true'"



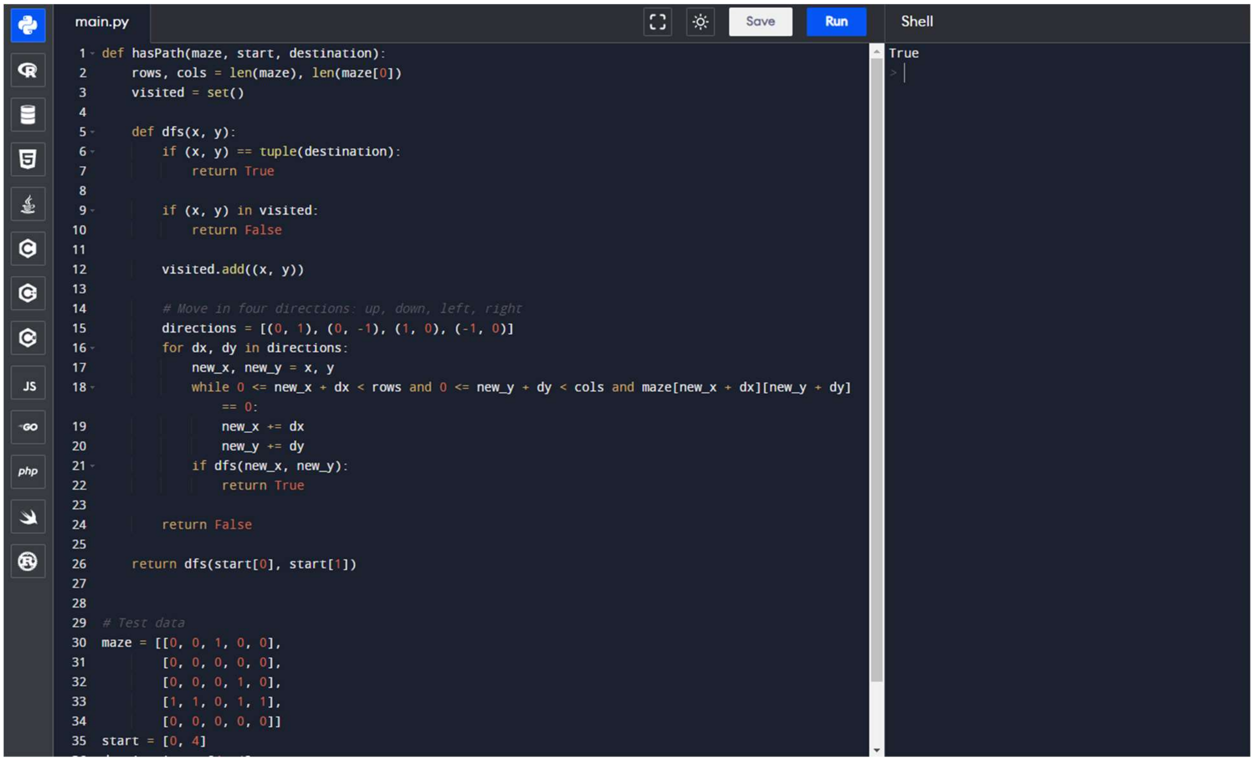
How to use Depth-First Traversal approach to implement a Python code to solve the LeetCode question: '490. The Maze', with the test data: 'Input: maze = [[0,0,1,0,0], [0,0,0,0,0], [0,0,0,1,0], [1,1,0,1,1], [0,0,0,0,0]], start =[0,4], destination = [4,4]; Output: true'

To solve the LeetCode question "490. The Maze" using the Depth-First Traversal (DFT) approach, we can treat the maze as a graph and perform a depth-first search to explore possible paths from the start point to the destination. The goal is to check whether we can reach the destination point from the start point while navigating through the maze.

Here's a Python code implementation using the DFT approach:

```python
def hasPath(maze, start, destination):
    rows, cols = len(maze), len(maze[0])
    visited = set()

    def dfs(x, y):
        if (x, y) == tuple(destination):
            return True

        if (x, y) in visited:
            return False

        visited.add((x, y))

        # Move in four directions: up, down, left, right
        directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
        for dx, dy in directions:
            new_x, new_y = x, y
            while 0 <= new_x + dx < rows and 0 <= new_y + dy < cols and maze
```

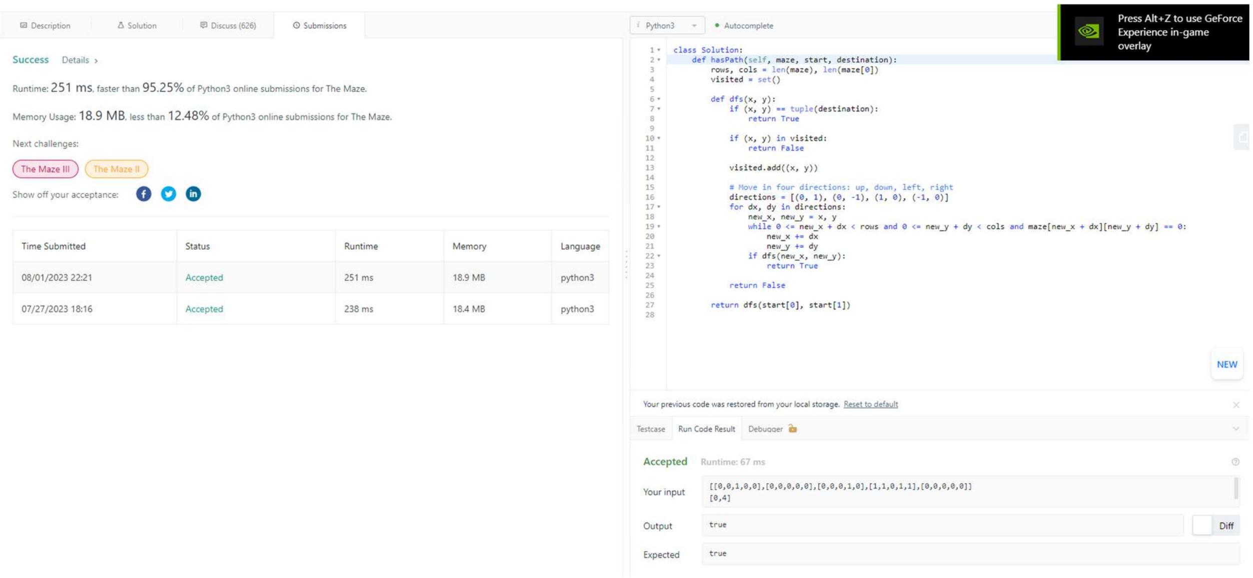
Step 4: Implement the Python code created by ChatGPT



```python
def hasPath(maze, start, destination):
    rows, cols = len(maze), len(maze[0])
    visited = set()

    def dfs(x, y):
        if (x, y) == tuple(destination):
            return True

        if (x, y) in visited:
            return False

        visited.add((x, y))

        # Move in four directions: up, down, left, right
        directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
        for dx, dy in directions:
            new_x, new_y = x, y
            while 0 <= new_x + dx < rows and 0 <= new_y + dy < cols and maze[new_x + dx][new_y + dy]
                == 0:
                new_x += dx
                new_y += dy
            if dfs(new_x, new_y):
                return True

        return False

    return dfs(start[0], start[1])


# Test data
maze = [[0, 0, 1, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 1, 0],
        [1, 1, 0, 1, 1],
        [0, 0, 0, 0, 0]]
start = [0, 4]
```

```
True
>
```

Step 5: Test the Python code with all the test cases provided by 490. The Maze - (local copy)



Success Details ›

Runtime: 251 ms, faster than 95.25% of Python3 online submissions for The Maze.

Memory Usage: 18.9 MB, less than 12.48% of Python3 online submissions for The Maze.

Next challenges:

The Maze III The Maze II

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
08/01/2023 22:21	Accepted	251 ms	18.9 MB	python3
07/27/2023 18:16	Accepted	238 ms	18.4 MB	python3

```python
class Solution:
    def hasPath(self, maze, start, destination):
        rows, cols = len(maze), len(maze[0])
        visited = set()

        def dfs(x, y):
            if (x, y) == tuple(destination):
                return True

            if (x, y) in visited:
                return False

            visited.add((x, y))

            # Move in four directions: up, down, left, right
            directions = [(0, 1), (0, -1), (1, 0), (-1, 0)]
            for dx, dy in directions:
                new_x, new_y = x, y
                while 0 <= new_x + dx < rows and 0 <= new_y + dy < cols and maze[new_x + dx][new_y + dy] == 0:
                    new_x += dx
                    new_y += dy
                if dfs(new_x, new_y):
                    return True

            return False

        return dfs(start[0], start[1])
```

Your previous code was restored from your local storage. Reset to default

Accepted Runtime: 67 ms

Your input

```
[[0,0,1,0,0],[0,0,0,0,0],[0,0,0,1,0],[1,1,0,1,1],[0,0,0,0,0]]
[0,4]
```

Output

```
true
```

Expected

```
true
```

Step 6: Update your portfolio about the Maze project

Please use this structure to describe the project

Algorithm

Depthe First Search

Maze

Step 7: Submit the URL of your GitHub webpage as the homework answer.

https://github.com/wangziming0915/Github-Portfolio/tree/main