Homework 4: Exploring Maze Solutions Using Depth-First Search

Introduction

In this Home work, you will apply the Depth-First Search (DFS) algorithm to solve a maze. DFS is a fundamental algorithm used for tree and graph traversal and is known for its simplicity and efficiency in certain scenarios. This assignment will simulate an application of DFS in navigating through a complex maze. Your task is to implement the DFS algorithm to find a path from the entrance to the exit in a given maze.

Scenario and Data

You are presented with a grid-based maze where each cell represents a possible path or a wall. The maze has a single entrance and exit point. Your goal is to find a path from the entrance to the exit, navigating through the maze using DFS.

Maze Representation

The maze is represented as a 2D grid. Here is an example:

- 'S' represents the Start (entrance).
- 'E' represents the End (exit).
- '1' represents a wall (not passable).
- '0' represents an open path (passable).

Example Maze (5x5 grid):

S	1	0	1	0
0	1	0	1	0
0	0	0	0	0
1	0	1	1	0
0	0	0	1	E

Task

Implement the DFS algorithm to navigate through the maze from the Start to the End.

Requirements

- Maze Representation: Represent the given maze in your preferred programming language.
- Implement DFS: **FULL** code to implement DFS for maze navigation.
- Path Finding: Apply DFS to find a path from the Start to the End of the maze.
- Documentation: your **FULL** code and the path found, if any.

Submission

Submit your pseudo code along with a description of your implementation, the path found (or not found), and your analysis of DFS in this scenario.

Hint: Pseudo Code function

```
DFS(maze, start, end):
    stack <- a stack data structure initialized with start
    visited <- an empty set
    while stack is not empty:
        current <- pop the top item from stack
        if current is the end:
            return success and the path taken
        if current is not in visited:
            add current to visited
            for each neighbor of current that is passable:
                push neighbor to stack
    return failure (no path found)
```