|  |  |
| --- | --- |
| **EXPT NO:4** | **Implementation of Search Algorithms (BFS and DFS)** |
| **DATE: 24.09.2025** |

**PRE-LAB QUESTIONS (PROVIDE BRIEF ANSWERS TO THE FOLLOWING QUESTIONS)**

1. **What is a search algorithm in the context of Artificial Intelligence?**

A search algorithm in AI systematically explores possible states or actions within a problem space to find a solution, aiming to reach the goal state from the initial state.

1. **What is the fundamental difference in the exploration strategy between Breadth-First Search (BFS) and Depth-First Search (DFS)?**

BFS explores level by level, expanding all nodes at the current depth before moving deeper, while DFS explores as far down one path as possible before backtracking to explore alternatives.

1. **Which data structures are typically used to implement BFS and DFS, and why?**

BFS typically uses a queue to maintain nodes in FIFO order, ensuring level-wise exploration. DFS uses a stack (explicit or recursion) to explore deeper nodes before backtracking systematically.

1. **In what scenarios would you prefer to use BFS over DFS?**

BFS is preferred when the goal is likely near the starting state, when the shortest solution path is required, or in small, finite graphs where memory use is manageable.

1. **Define the terms ‘State Space’, ‘Initial State’, and ‘Goal State’.**

State space is the set of all possible configurations; initial state is where the search begins; goal state represents desired target condition where the problem is considered solved.

**IN-LAB EXERCISE**

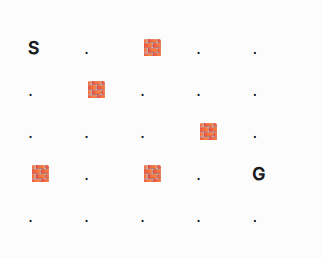
**OBJECTIVE:**

To implement and compare BFS and DFS by programming a robot to navigate a maze, demonstrating how each algorithm finds a path from a start point to an exit.

**PROCEDURE:**

**. Scenario:**

* **Problem Statement:**  
  You are programming a rescue robot trapped in a simple warehouse represented by a grid. The robot must find a path from its starting position to the exit. The robot can only move Up, Down, Left, or Right into empty cells.
* **Warehouse Layout (Maze):**  
  Represent the warehouse as a grid. 'S' is the start, 'G' is the goal, '#' are walls, and '.' are open paths.
* **Task:** Implement both BFS and DFS to find a valid path from 'S' to 'G'.

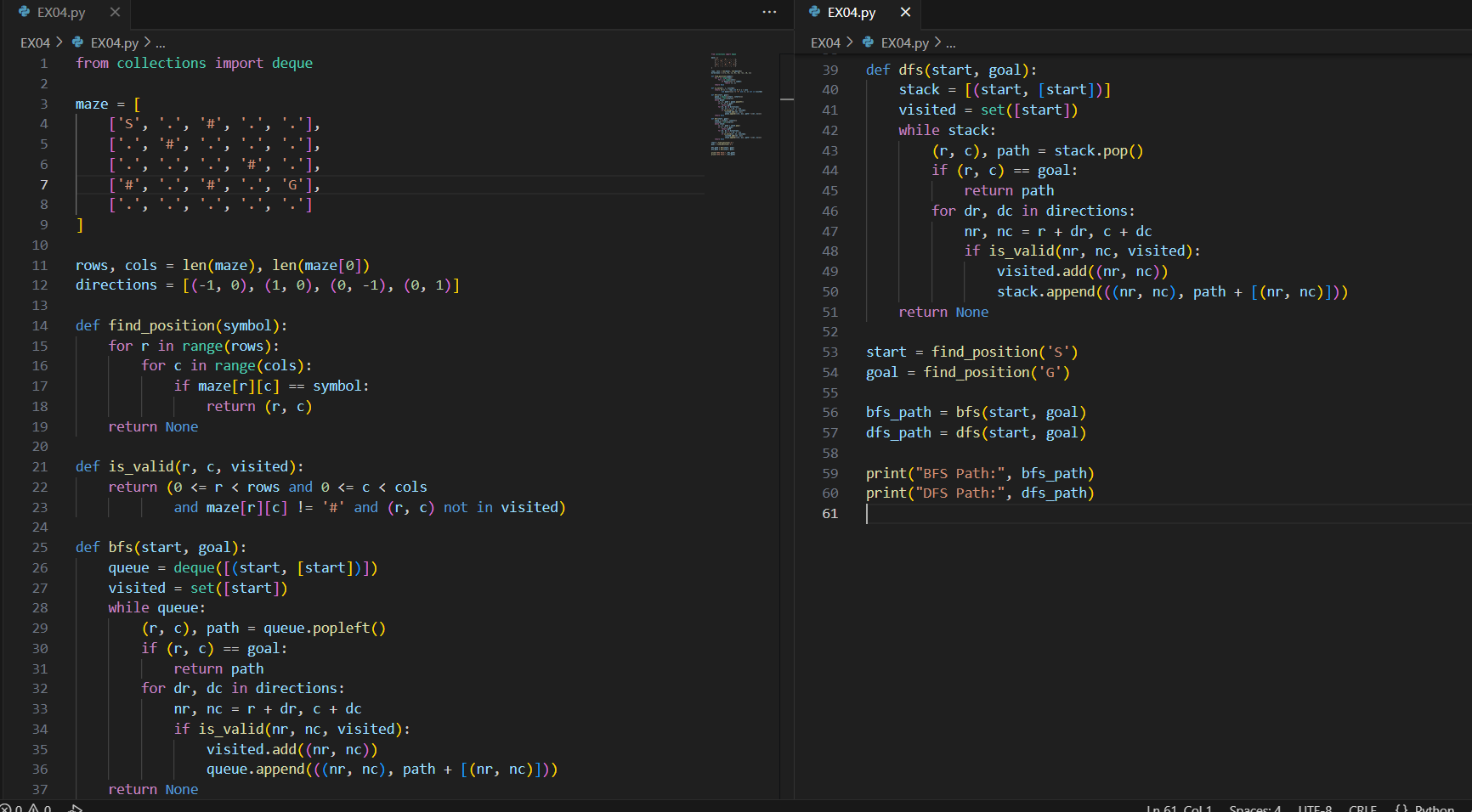


**Legend:**

* **S** / **Start**: The robot's starting position.
* **G** / **Goal**: The warehouse exit.
* **🧱** / **Wall**: An impassable obstacle.
* **.** / **Path**: An open cell where the robot can move.

**2. Implementation Steps:**

* Represent the grid. An adjacency list is a good choice, where each cell (row, col) is a node.
* Implement the BFS algorithm using a Queue to find a path from 'S' to 'G'.
* Implement the DFS algorithm using a Stack or recursion to find a path from 'S' to 'G'.
* For each algorithm, store and print the path found (e.g., a list of coordinates from S to G).

**SCREENSHOT OF CODE**  
****

**SCREENSHOT OF OUTPUT**

A black screen with white text

AI-generated content may be incorrect.

**POST-LAB QUESTIONS (PROVIDE BRIEF ANSWERS TO THE FOLLOWING QUESTIONS)**

1. Looking at the two paths your robot found in the warehouse maze, the BFS path was likely shorter and more direct. The DFS path may have taken a long, winding route before finding the exit. Explain *why* this happens, based on the fundamental difference in how each algorithm explores the open cells of the maze.

BFS explores level by level, so it quickly finds the nearest exit with the shortest route. DFS dives deep into one path, possibly wandering through long dead ends before backtracking.

1. Imagine the warehouse maze was enormous (millions of cells) with very long, dead-end corridors. In terms of computer memory usage (space complexity), which algorithm, BFS or DFS, would likely perform better and why? Explain your choice in the context of how much data each algorithm needs to store at any given time.

DFS usually performs better in memory usage. It only stores nodes along the current path (stack depth). BFS must store every frontier node at each level, consuming huge memory.

1. The "CityNav" GPS company claims its basic feature, which uses BFS, *always* finds the route with the fewest road segments. Is this claim valid? Under what key assumption about the "cost" or "length" of each road segment does this guarantee hold true?

The claim is valid if every road segment has equal cost or length. BFS guarantees the fewest steps only under the assumption of uniform edge cost in the road network.

1. A web crawler (like Google's bot) explores the internet by following links from one page to another. If you designed a crawler using a pure DFS strategy, what major risk would you face that might prevent it from ever finding certain websites, even if they are reachable? Why is a BFS-like strategy generally "safer" for ensuring comprehensive coverage?

A pure DFS crawler risks getting stuck endlessly in deep link chains, missing other reachable sites. BFS-like strategies systematically expand outward, ensuring broader, safer coverage of all accessible websites.

1. The "ConnectLink" social network has two separate communities of users with no connections between them (a disconnected graph). What should your search algorithm report if a user from the first community tries to find a connection path to a user in the second? How does your code know when to stop searching and conclude that "no path exists"?

The algorithm should report "no path exists." It knows to stop when all reachable nodes from the source are explored and the goal is not found, indicating disconnected components.

**RESULT:**

Hence, algorithms for a path-finding robot were successfully implemented using Breadth-First Search (BFS) and Depth-First Search (DFS). The robot successfully navigated a grid-based maze, and the resulting paths demonstrated the core differences between the two search strategies. This experiment highlighted BFS's strength in finding the shortest path in unweighted scenarios and DFS's different, depth-oriented exploration approach.

**ASSESSMENT**

|  |  |  |
| --- | --- | --- |
| **Description** | **Max Marks** | **Marks Awarded** |
| Pre Lab Exercise | **5** |  |
| In Lab Exercise | **10** |  |
| Post Lab Exercise | **5** |  |
| Viva | **10** |  |
| **Total** | **30** |  |
| **Faculty Signature** | |  |