Project Brief

SESSION 4: Complexity Analysis and Testing of the Maze Resolution Algorithm (Bidirectional Search)

HOST : Tingting **SCRIBE :** Daniel **SECRETARY :** Lucas

INTRODUCTION

In this session, we discussed how to effectively apply our bidirectional search approach for maze resolution. We agreed to use the Breadth-First Search (BFS) method, analyzed its computational complexity, and outlined the tests required to ensure robust performance.

COMPLEXITY ANALYSIS OF THE MAZE RESOLUTION ALGORITHM

The algorithm uses a bidirectional BFS from both the start and end points to identify the shortest path.

- Best-Case Scenario: If the start and end points are very close (possibly adjacent) or if there is a short, direct path, BFS will quickly find the solution with minimal exploration.
- Complexity: O(1), as only a few cells need to be visited to find the shortest path.
- General or Worst-Case Scenario: In complex mazes where the start and end points are far apart, BFS may need to explore a significant portion of the maze from both directions until the searches converge.
- With a bidirectional approach, each search only explores approximately half of the total maze area before intersecting. However, in any layout, every cell may be visited, resulting in a Complexity of O(n×m).

TESTING PLAN

To ensure the algorithm is both reliable and efficient, we will conduct the following tests:

Unit Tests(Daniel): These tests will cover basic functionality to confirm that the algorithm performs as expected across typical scenarios, including:

- Simple valid maze
- Multiple paths
- Single path
- No path available
- Complex maze

Limit Case Testing(Anita): Testing for extreme cases will help identify any weaknesses or issues that could arise in unusual or boundary scenarios.

Performance Testing(Karim): We will evaluate the algorithm's performance with increasingly large and complex mazes, using stress tests to assess its efficiency under heavy computational loads.

Randomized Testing(Sara): Generating random mazes and verifying if the algorithm can solve them will help reveal any unexpected bugs or edge cases.