Session 3: Maze Resolution Algorithm(Bidirectional Search)

HOST: DANIEL

SCRIBE: SARA

SECRETARY: ANITA

Introduction:

In this session, we focused on designing an efficient maze resolution algorithm to find the **shortest path** from any given position to the exit. After reviewing and discussing several approaches (DFS, BFS, and Bidirectional Search), the team agreed that **Bidirectional Search** is the best method due to its efficiency in reducing the search space, especially for large mazes.

Step-by-Step Algorithm:

1. **Initialize the Search**:
    - Create two queues:
        - One for the search starting at the **entrance** (q_start).
        - One for the search starting at the **exit** (q_end).
    - Mark the entrance cell as visited by the **start search**.
    - Mark the exit cell as visited by the **end search**.
2. **Define Directions**:
    - Set up directions for moving through the maze: left, right, up, and down.
3. **Run the Search in a Loop**:
    - While both queues are not empty, repeat the following steps:

Process from the Entrance:

- Dequeue the front cell from q_start.
- For each possible direction (left, right, up, down):
    - If the neighbor cell in that direction is within bounds, not blocked by a wall, and not visited:
        - Mark it as visited by the **start search**.
        - Add the neighbor to q_start.
        - Check if this cell has been visited by the **end search**. If yes, the searches have met, and the path is found.

Process from the Exit:

- Dequeue the front cell from q_end.
- For each possible direction (left, right, up, down):
  - If the neighbor cell in that direction is within bounds, not blocked by a wall, and not visited:
    - Mark it as visited by the **end search**.
    - Add the neighbor to q_end.
    - Check if this cell has been visited by the **start search**. If yes, the searches have met, and the path is found.

4. **Combine Paths**:
   - When a cell is found that is visited by both searches, this is the **meeting point**.
   - Combine the path from the entrance to the meeting point with the path from the meeting point to the exit using **parent tracking**.
5. **Return the Result**:
   - If the searches meet, return the combined path as the shortest path from the entrance to the exit.
   - If no meeting point is found and both queues are empty, there is no valid path, so return failure.