1. State Representation

- The state consists of the agent's position (x, y) and collected prizes.

- Implemented using a `State` class for efficiency, with a set tracking collected prizes.

2. Transition Model

- The agent moves in four directions: North, South, East, and West.

- `move_agent(state, action)` updates the position if the move is valid.

- Collects prizes when stepping on them.

3. Goal Test

- `goal_test(state)` checks if all prizes are collected.

4. Future Implementation Plans

Part 2: Depth-First Search (DFS)

- Implement `single_dfs(maze_file)` to explore paths deeply.

- Outputs the solution path, path cost, and nodes expanded.

Part 3: BFS, GBFS, and A*

- Implement `single_bfs()`, `single_gbfs()`, and `single_astar()`.

- BFS guarantees the shortest path, GBFS uses Manhattan distance, and A* combines cost and heuristic.

- Outputs the solution, path cost, and nodes expanded.

Part 4: Multi-Prize A*

- Implement `multi_astar()` to collect multiple prizes efficiently.

- Uses an admissible heuristic to optimize pathfinding.

6. Conclusion

This implementation ensures scalability and efficiency, supporting multiple search algorithms. The structured approach allows easy expansion for complex search problems.