Minimizing Moves to Complete a Round: Solution Explanation

In this problem, we need to:

1. Navigate a vehicle around a track in a clockwise direction
2. Start and end at the same position with speed = 0
3. Minimize the number of moves taken
4. Avoid obstacles and stay within lanes
5. Follow specific movement rules:
    - Can change speed by -1, 0, or +1 per move
    - Can move laterally by 1 unit per move
    - Can only change direction when speed is 0 or 1

Here i have treated the problem as a graph search problem where:

- Nodes are states defined by (position, speed, direction, moves, visited_directions)
- Edges are valid moves between states
- We need to find the shortest path from the starting state to a state where we've completed a round

This approach enables us to use A* search algorithm to find the minimum number of moves efficiently. The solution i have used can be divided into following parts:

1. State Representation

Each state includes:

- Position (row, col)
- Current speed
- Current direction (0=right, 1=down, 2=left, 3=up)
- Number of moves taken so far
- Parent state (for path reconstruction)
- Set of visited directions to track clockwise progress around the track

2. Valid Move Generation

For each state, we generate all possible next states by:

- Trying all speed changes (-1, 0, +1) in the current direction
- Checking for direction changes when speed is 0 or 1
- Verifying that paths are clear of obstacles and lane boundaries

3. Path Validation

We check if a path:

- Stays within the grid boundaries
- Avoids obstacles (cells with value 1 or 2)
- Is physically possible (all cells along movement path are clear)

4. Round Completion Check

We verify a round is complete when:

- We return to the starting position
- We have visited all four directions (ensuring clockwise travel)
- Our speed is 0 (as required)

5. Optimization with A* Algorithm

- Use a priority queue to explore states with the lowest estimated total cost
- Track visited states to avoid cycles
- The heuristic function considers:
  - Distance to starting position
  - Current speed (since we need to end with speed 0)
  - Number of directions not yet visited

Algorithmic Approach

1. **Initialization**: Create a starting state at the initial position with speed 0
2. **Search**: Use a priority queue to explore states in order of (moves + heuristic)
3. **Move Generation**: For each state, generate all possible next states

4. **Validation**: Check if each next state is valid (no obstacles, within boundaries)
5. **Completion Check**: Check if we've completed a round
6. **Result**: Return the path with minimum moves