**Theoretical Analysis:**

This section presents a high-level overview of the simulation, focusing on its objectives, agent-environment interaction, and handling of obstacles.

Theoretical Analysis:

* The simulation models the behavior of a vacuum cleaner agent in a grid-based environment.
* The vacuum cleaner agent aims to navigate the grid, clean dirty tiles, and optimize its performance.
* The agent uses a reflex-based decision-making approach to determine its actions based on the current tile's dirt status and its own location.
* The environment includes obstacles randomly placed within the grid, which the agent must navigate around.
* The simulation tracks the agent's performance, considering the cleaning of dirty tiles and the presence of obstacles.

**Data Structure:**

This section describes the key data structures used to represent the agent, environment, grid, obstacles, and their interactions.

Data Structures:

* **ReflexVacuumAgent** class: Represents the vacuum cleaner agent and its decision-making logic.
* **VacuumCleanerEnvironment** class: Represents the environment with the grid, tiles, agent location, and obstacles.
* 2D list (**tiles**): Represents the grid of tiles, each with a Boolean value indicating dirt status.
* 2D list (**obstacles**): Represents the grid of obstacles, each with a Boolean value indicating presence.
* Tuple (**agent\_location**): Stores the current (x,y) coordinates of the agent's location.
* Integer (**performance\_measure**): Keeps track of the agent's performance score.

**Algorithm to Function/Method Representation:**

This section details the algorithms and corresponding functions/methods responsible for the agent's actions, movement, obstacle handling, and simulation execution.

Algorithm to Function/Method Representation:

* **decide\_action(self, location, is\_dirty)**: The agent's decision-making method that selects an action based on dirt status and location.
* Movement methods (**move\_left**, **move\_right**, **move\_up**, **move\_down**): These methods handle agent movements, considering obstacles.
* **suck(self)**: Cleans a dirty tile, increasing the agent's performance score.
* **place\_obstacles(self)**: Places obstacles randomly in the grid, ensuring they are not placed at the agent's location.
* **step(self, action)**: Executes an action, updating the agent's performance and handling obstacles.
* **initialize\_dirt(self, exclude\_tile)**: Initializes dirt on tiles except for the agent's location.

**Implementation:**

This section outlines the overall implementation of the simulation, including class definitions, methods, loops, and the simulation execution.

Implementation:

* **ReflexVacuumAgent** and **VacuumCleanerEnvironment** classes: Define the agent and environment logic.
* **run\_simulation(dimensions, num\_simulations)**: Executes the simulation loop for the specified number of simulations.
* The simulation loop iterates through time steps, initializing dirt and obstacles, deciding actions, and tracking performance.

**Input Test Cases Format:**

This section defines the input format required to run the simulation, specifying the grid dimensions and the number of simulations.

Input Test Cases Format:

* Dimensions of the grid: A tuple **(rows, columns)** representing the size of the grid.
* Number of simulations: An integer indicating how many times the simulation should be run.

**Output Format:**

This section explains the format of the output produced by the simulation, including performance scores and information about obstacle placement.

Output Format:

* Performance for each simulation: A list of integers, where each integer represents the performance score of a single simulation run.
* Obstacles: A 2D list, where each element indicates the presence of an obstacle in the corresponding grid cell.
* Each simulation's performance and obstacle information are printed.

This documentation summarizes the key aspects of the simulation, from theoretical considerations and data structures to algorithms, implementation details, input test case formats, and output formats.