**Theoretical Analysis:**

In this section, we'll discuss the high-level concepts and ideas behind the simulation of the vacuum cleaner agent in a grid-based environment. We'll explore the decision-making process of the agent, how it interacts with the environment, and how the performance is measured.

Theoretical Analysis:

The simulation involves an environment with a grid of tiles represented by (x,y) coordinates.

The vacuum cleaner agent operates in this environment and is guided by a simple reflex-based decision-making strategy.

The agent's main objective is to clean dirty tiles and move around the grid to maximize its performance.

At each time step, the agent observes its current location and the dirt status of the tile it is on.

Based on these observations, the agent selects an action, which can be to clean the tile ("Suck") or move in any direction ("Left", "Right", "Up", "Down").

The agent's performance is measured by a performance measure, which is increased when it successfully cleans a dirty tile and decreased with each action it takes.

Data Structure:

Here, we'll describe the data structures used in the simulation to represent the environment, the agent, and the state of the grid.

**Data Structures:**

ReflexVacuumAgent class: Represents the vacuum cleaner agent.

VacuumCleanerEnvironment class: Represents the environment and contains data about the grid tiles, the agent's location, and the performance measure.

2D list (tiles): Represents the grid of tiles in the environment. Each element stores a Boolean value indicating whether the tile is dirty or clean.

Tuple (agent\_location): Represents 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 outlines the core algorithms and functions/methods used in the simulation to enable the agent's decision-making and interaction with the environment.

**Algorithm to Function/Method Representation:**

decide\_action(self, location, is\_dirty): The agent decides on an action based on the current location and tile dirt status. It returns an action ("Suck", "Left", "Right", "Up", or "Down").

Movement methods (move\_left, move\_right, move\_up, move\_down): These methods handle the agent's movement within the grid while respecting boundaries.

suck(self): Cleans the current tile if it's dirty and updates the performance measure.

step(self, action): Receives an action, performs the corresponding action, updates the performance measure, and returns the new performance measure.

Implementation:

This section details the actual implementation of the simulation using Python code. It covers class definitions, methods, and loops.

**Implementation:**

Class ReflexVacuumAgent: Defines the agent's decision-making logic.

Class VacuumCleanerEnvironment: Defines the environment, methods for actions, and methods for interaction with the agent.

run\_simulation(): Sets up the simulation environment, runs simulations, and prints performance results.

Input Test Cases Format:

In this section, we define the format of the input data used to run the simulation, including the dimensions of the grid 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:

Finally, this section explains the format of the output produced by the simulation, including the performance score for each simulation and the overall average score.

**Output Format:**

Performance for each simulation: A list of integers, where each integer represents the performance score of a single simulation run.

Overall Average Score: A single float representing the average performance score across all simulations.

This documentation should provide a comprehensive overview of the simulation, covering theoretical concepts, data structures, algorithms, implementation details, input test case formats, and output formats.