Name	ID	17
Abdelhameid	208385	
Youssef	210780	
Noeman	213124	

Vacuum cleaner

Our agent is a vacuum cleaner which is a goal-based agent. The goal is to clean the entire environment with a strategy. That means the agent will take into consideration the best available actions that improve the progress towards the goal. The agent can have a long sequence of possible actions to achieve the goal. The agent can detect whether there is dirt in the current area or its clean, also the agent can detect if it faces an obstacle. For example, wall or furniture. The vacuum cleaner that works in an environment comprises of two rooms that are connected by an open pathway The room layout is represented as a grid of characters, where 'S' is the start position, 'G' is the goal, '1' denotes obstacles, and '0' represents open space the number of obstacles in the first room is 3 and for the second room is 1. We chose Qlearning algorithm, which is a reinforcement algorithm used to make decisions in an environment. It is particularly well-suited for problems where an agent interacts with an environment, takes actions, receives rewards, and learns to optimize its behavior over time.

The motivation is to develop a vacuum cleaner machine that runs autonomously that can successfully traverse through the given environment and to clean up the dirt on their own.

Scenario:

The vacuum cleaner starts by reading its environment. It then starts to plan its cleaning path, taking into account the location of furniture, walls, and other obstacles. As it moves around the room, it constantly updates its model to reflect any changes in the environment. As the vacuum cleaner moves around the room, it scans for dirt on the floor. When it finds a dirty spot, it then takes action to clean the spot.

- **Initialization (`__init__`):**
  - The **`__init__`** method initializes the Q-learning algorithm with the number of states, number of actions, and hyperparameters (learning rate, discount factor, epsilon).
  - def _init_(self, n_states, n_actions, learning_rate=0.1, discount_factor=0.9, epsilon=0.1)
- **`self`**: The **`self`** parameter refers to the instance of the class. It is a convention in Python to name this parameter **`self`**. It allows you to access the attributes and methods of the class within the class.

- **`n_states`**: The number of states in the environment. In the context of Q-learning, states represent different situations or configurations that the agent can be in.
- **`n_actions`**: The number of possible actions that the agent can take in each state. Actions are the choices the agent can make.
- **`learning_rate`**: A hyperparameter that determines the step size at which the Q-values are updated. It controls how much of the new information is incorporated into the existing Q-values. The default value is set to 0.1.
- **`discount_factor`**: A hyperparameter that represents the importance of future rewards. It discounts the value of future rewards, giving more weight to immediate rewards. The default value is set to 0.9.
- **`epsilon`**: A hyperparameter that controls the exploration-exploitation trade-off. It determines the probability of choosing a random action (exploration) instead of the action with the highest Q-value (exploitation). The default value is set to 0.1.
    -
- **Choosing Action (`choose_action`):**
    - The **`choose_action`** method selects an action for the agent based on an epsilon-greedy strategy.
    - With probability **`epsilon`**, it chooses a random action; otherwise, it exploits the current knowledge by selecting the action with the highest Q-value.
- **Learning (`learn`):**
    - The **`learn`** method updates the Q-values based on the observed state, action, reward, and the next state.
    - It calculates the target Q-value using the Bellman equation and updates the Q-value for the chosen action.
- **Saving and Loading (`save` and `load`):**
    - The **`save`** and **`load`** methods allow for saving and loading the learned Q-table, respectively.

GUI:

This environment consists of two rooms connected by an open pathway. The green block is the starting point of the agent. The blue circle is the vacuum cleaner, and the brown block is the dirt which is the goal. The black blocks are the obstacles