Q-Learning for Path Finding Floor

Team 1: Royer, Ye, Ildar, Tess

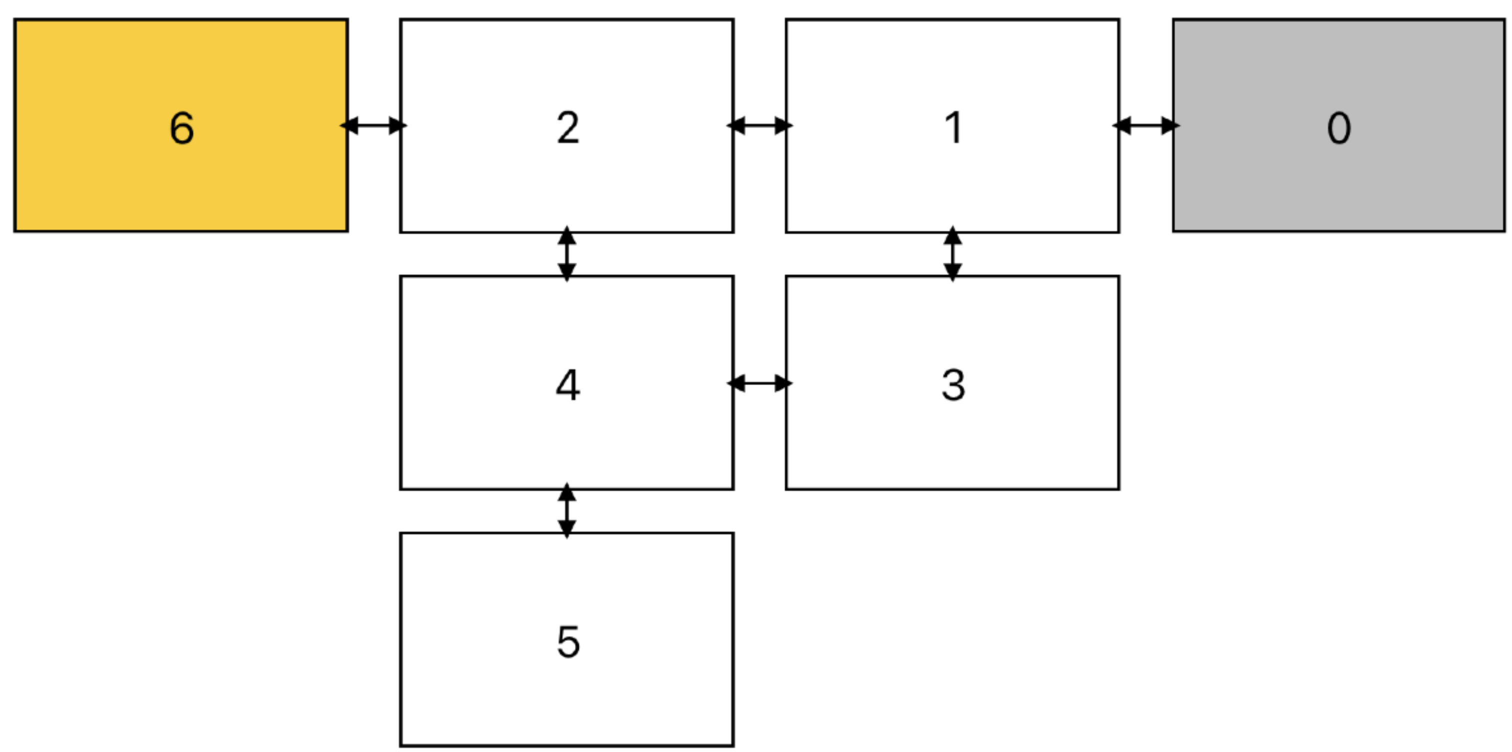
Reinforcement Learning

Path Finding Floor

The objective is to find the shortest route

- The Q-table (or Q-matrix) in this code is a 7x7 matrix, initially filled with zeros. It represents the learned values of state-action pairs, which get updated during the Q-learning process.

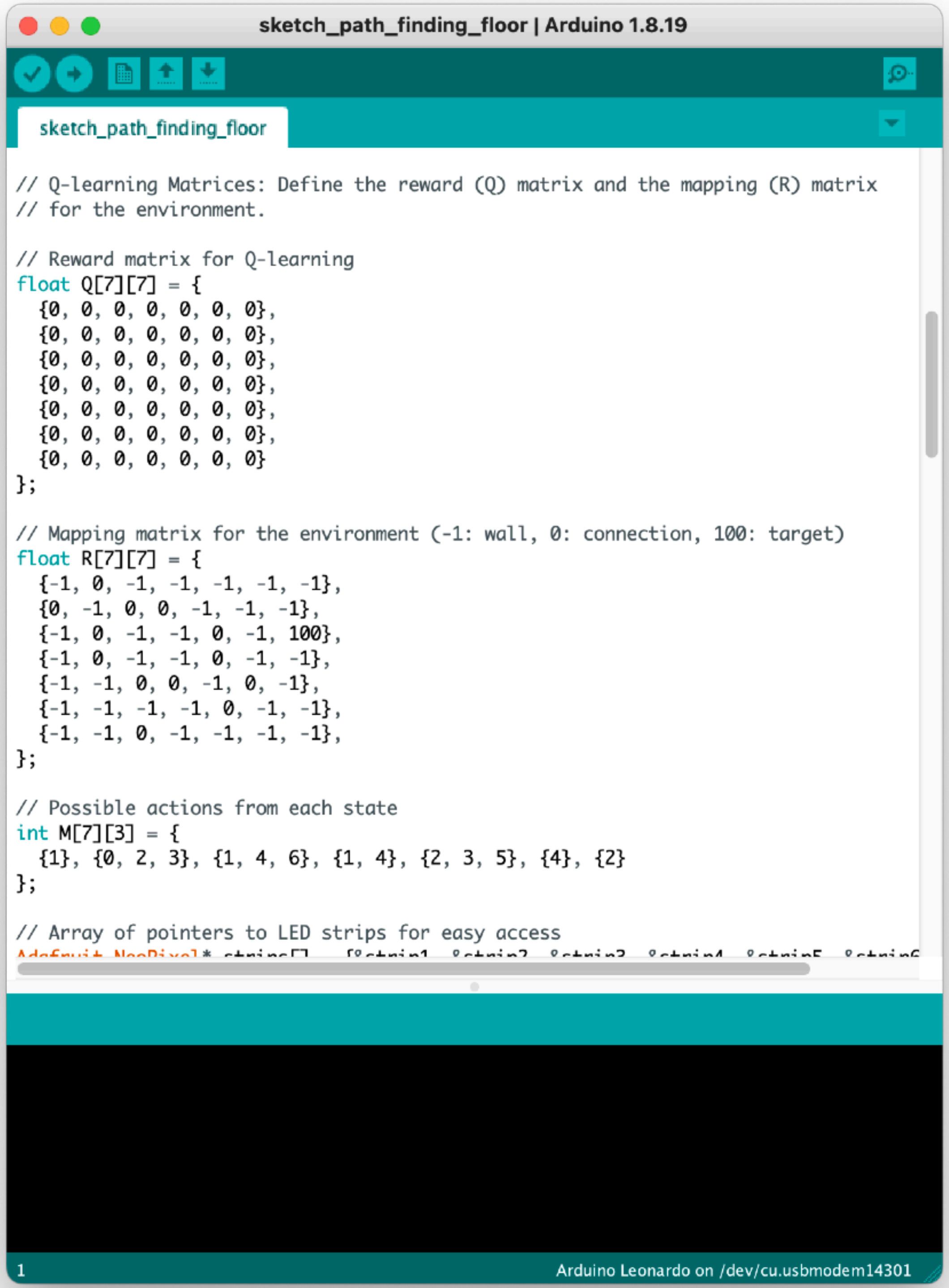
- Each element Q[state][action] in the Q-table indicates the expected utility (or quality) of taking a particular action in a particular state.



Q Table Initialisation

The Q-table is initialised as follows:

```
float Q[7][7] = {
  {0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0}
};
```



Reward and Mapping

- The reward matrix R and the mapping matrix are used to define the environment, including the connections between states, obstacles (walls), and the target.
- **Reward Matrix R:** The reward matrix R specifies the reward values for transitioning from one state to another. It uses -1 for walls, 0 for valid connections, and 100 for the target state.

number of the rooms

reward and possible actions

```
float R[7][7] = {
  {-1, 0, -1, -1, -1, -1, -1},
  {0, -1, 0, 0, -1, -1, -1},
  {-1, 0, -1, -1, 0, -1, 100},
  {-1, 0, -1, -1, 0, -1, -1},
  {-1, -1, 0, 0, -1, 0, -1},
  {-1, -1, -1, -1, 0, -1, -1},
  {-1, -1, 0, -1, -1, -1, -1}
};
```

Reward and Mapping

- **Mapping Matrix M:** The mapping matrix M lists the possible actions (next states) from each state. Each row represents a state, and the columns list the indices of possible next states.

```c
int M[7][3] = {
  {1},          // State 0 can transition to state 1
  {0, 2, 3},    // State 1 can transition to states 0, 2, 3
  {1, 4, 6},    // State 2 can transition to states 1, 4, 6
  {1, 4},       // State 3 can transition to states 1, 4
  {2, 3, 5},    // State 4 can transition to states 2, 3, 5
  {4},          // State 5 can transition to state 4
  {2}           // State 6 can transition to state 2
};
```

Explanation:

- **Reward Matrix (R):** The reward matrix defines the immediate reward for each action from a state. A reward of 100 is given for reaching the target state (state 6 from state 2), while other valid transitions have a reward of 0, and invalid transitions (walls) have a reward of -1.

- **Mapping Matrix (M):** The mapping matrix defines which states can be reached from each current state. This helps in selecting the next action during the Q-learning process.

Q-learning Update

- **Q-learning Update:** During the Q-learning process, the Q-table is updated based on the reward matrix and the possible actions:
  - **Choose an Action:** From the current state, choose an action (next state) based on the possible actions defined in the mapping matrix.
  - **Calculate Q-value:** For the chosen action, update the Q-value using the formula:

$$Q[state][action] = R[state][action] + \gamma \times \max(Q[action][all\_possible\_next\_st$$

where γ is the discount factor.

  - **Transition to Next State:** Move to the next state and repeat the process until the target state is reached.

Training Phase

- The training phase iterates over multiple episodes to update the Q-table and learn the optimal policy:

```
while (k < 10) {
  state = 0; // Start from the initial state
  ...
  while (state != 6) {
    ...
    // Update Q-value
    Q[state][action] = R[state][action] + Qmax;
    state = action; // Move to the next state
    ...
  }
  k++; // Increment episode counter
}
```

Exploitation Phase

- After the training phase, the exploitation phase uses the learned Q-table to follow the optimal policy:

```
while (1) {
  state = 0; // Start from the initial state
  ...
  while (state != 6) {
    ...
    state = l; // Move to the next state based on maximum Q-value
    ...
  }
}
```

In summary

- the Q-table is initially filled with zeros and gets updated during the training phase based on the rewards and possible transitions defined in the reward and mapping matrices.

- The exploitation phase then uses the learned Q-table to find the optimal path to the target state.