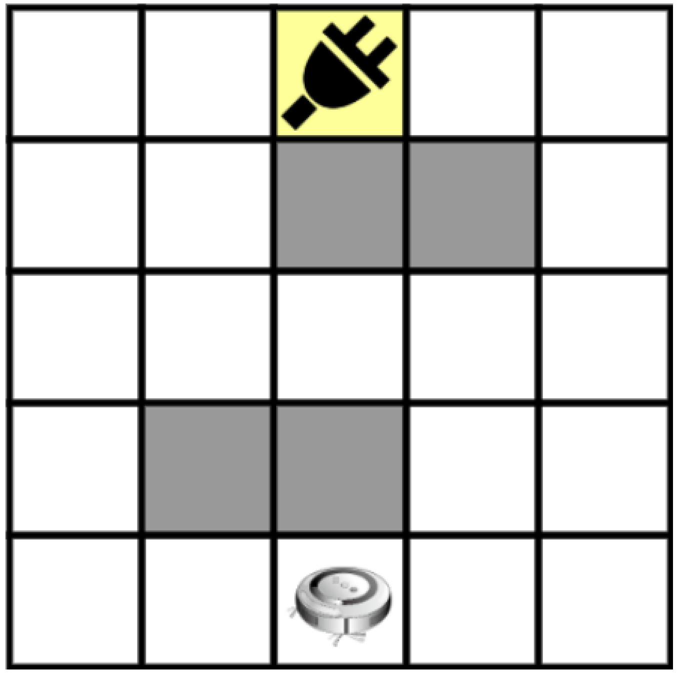
Q-learning Assignment - 1

Shyam Rauniyar

M13944385

Assignment

Homework



Goal:
Dock the robot in 10 steps.

Observations:

You need to define this.

Actions:

Up, Down, Left, Right

Rules (Kinematics only):

Robot stays in the same cell if hit the boundaries or the gray cells.

Reward Function:
You need to define this.

Termination Conditions:

1. Robot reaches charging dock.
2. Episode length greater than 50 steps (you can relax this constraint).

Expected Results:

1. Executable script (Matlab/Python).
2. Total reward growth curve (total reward per episode).
3. Q-table at 0,1,10 and last episode.

Result: 1. Executable Script

- Please Refer to the q-learning_assign_1.py script attached with the submission
- Or refer to the Q_learning_Assign_1.ipynb file attached with the submission which may be uploaded on jupyter or google colab notebook for detailed description of solution

Steps Involved

- Initialize the environment
  - In a 5X5 grid environment, robots search for the path to reach the charging dock area from given position i.e. [4,2] or wherever it is in 10 steps.
  - The states in the environment are all of the possible locations within the grid. Some of these locations are inaccessible (gray squares), while other locations are aisles that the robot can use to travel throughout the grid (white squares). The charging dock area is in yellow cell indicates the charging dock area.

Steps Involved (Contd.)

- Define the Actions
  - The actions that are available to the AI agent are to move the robot in one of four directions:
    - Up
    - Right
    - Down
    - Left
  - Obviously, the AI agent must learn to avoid driving into the inaccessible locations!

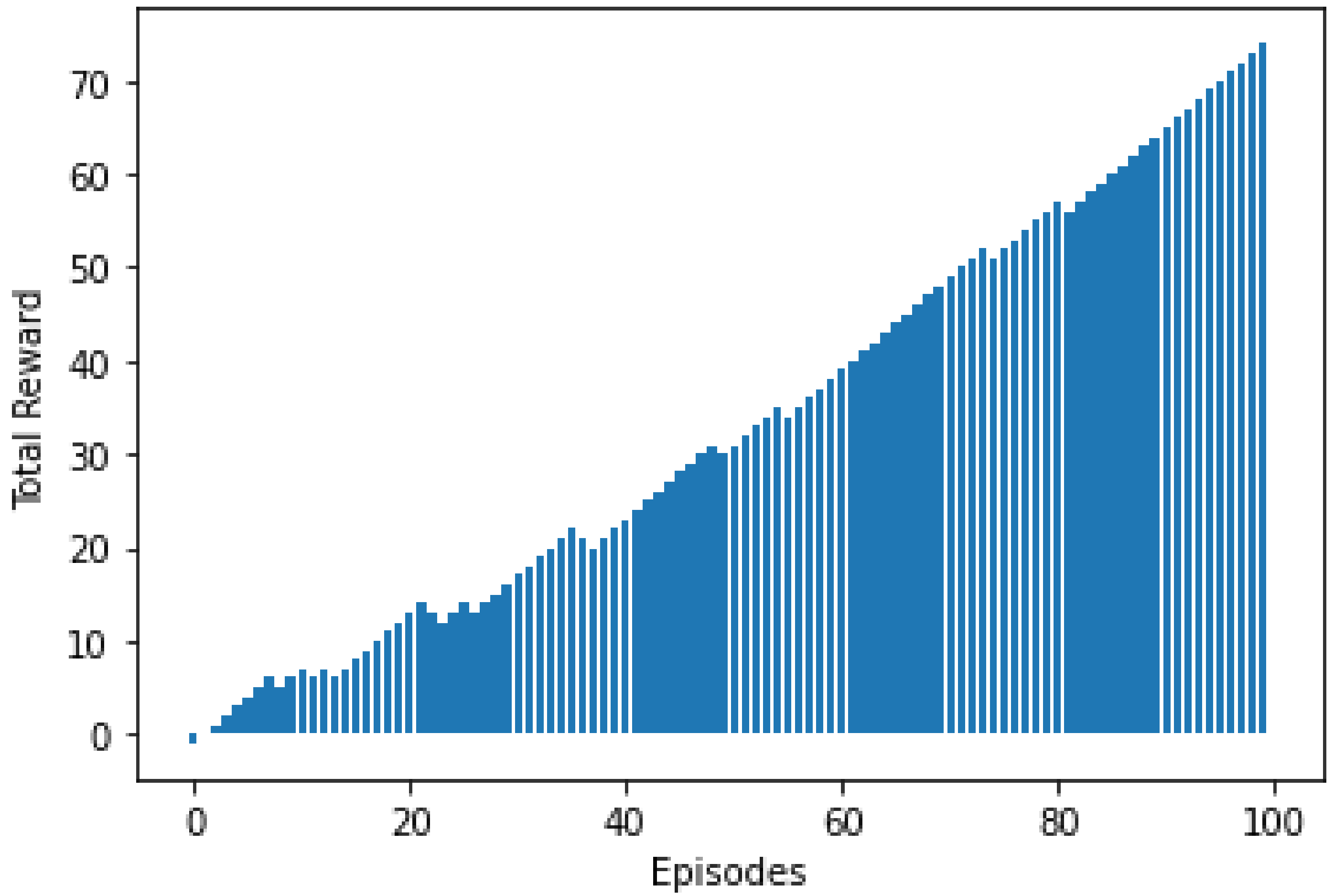
Steps Involved (Contd.)

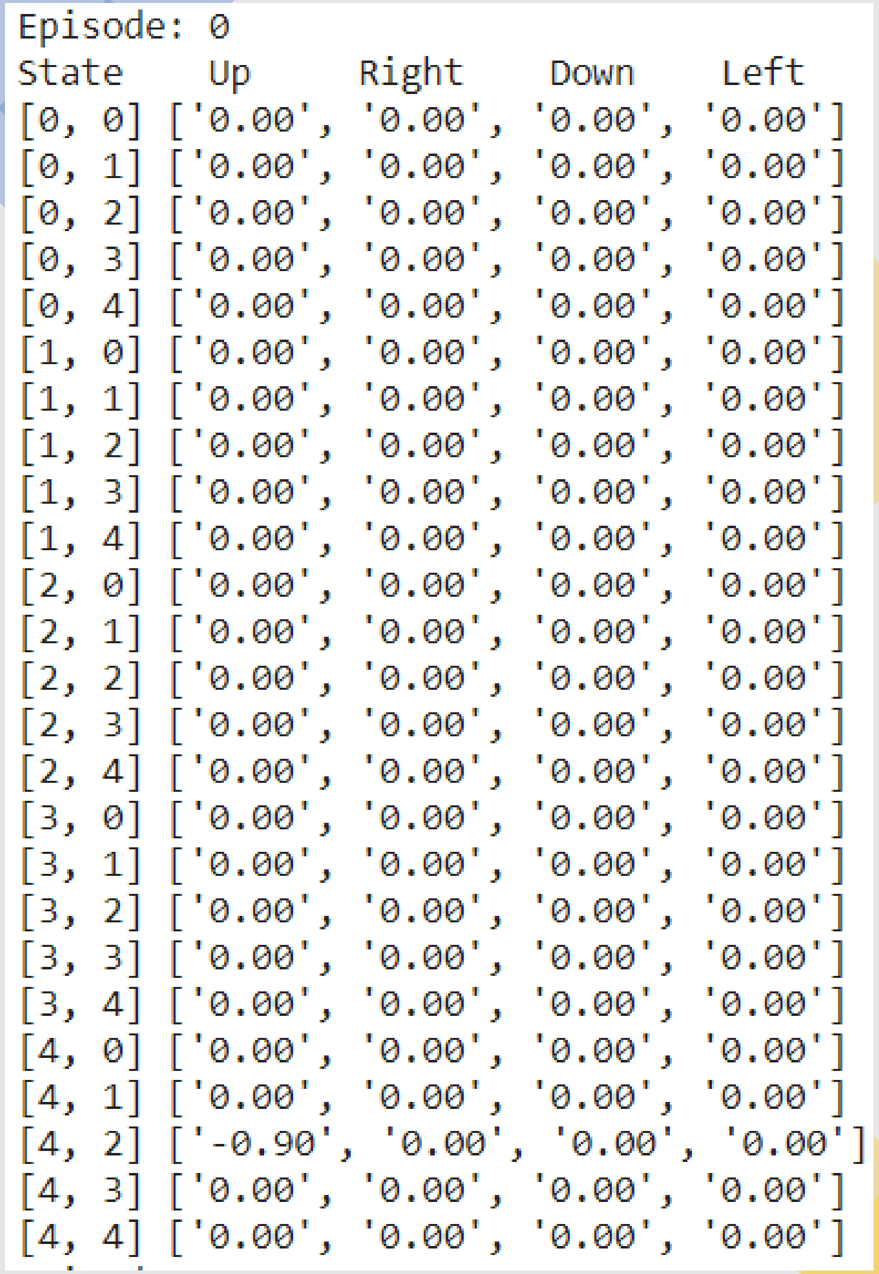
- Define the rewards
  - To help the AI agent learn, each state (location) in the grid is assigned a reward value.
  - The agent may begin at any white square, but its goal is always the same: to maximize its total rewards!
  - The reward for the charging dock area (i.e., the goal) to 1.
  - The reward for all inaccessible locations (i.e., black squares)is set to -1.
  - The reward for all the white squares was set to 0.

Steps Involved (Contd.)

- Training for 100 episodes
  1. Choose a random non-terminal state (white square) or the given position of [4,2], for the agent to begin this new episode.
  2. Choose an action (move up, right, down, or left) for the current state. Actions will be chosen using an epsilon greedy algorithm. This algorithm will usually choose the most promising action for the AI agent, but it will occasionally choose a less promising option in order to encourage the agent to explore the environment with epsilon = 0.8.
  3. Perform the chosen action, and transition to the next state (i.e., move to the next location).
  4. Receive the reward for moving to the new state and calculate the temporal difference with learning rate = 0.9 and discount rate = 0.9.
  5. Update the Q-value for the previous state and action pair.
  6. If the new (current) state is a terminal state, go to #1. Else, go to #2.

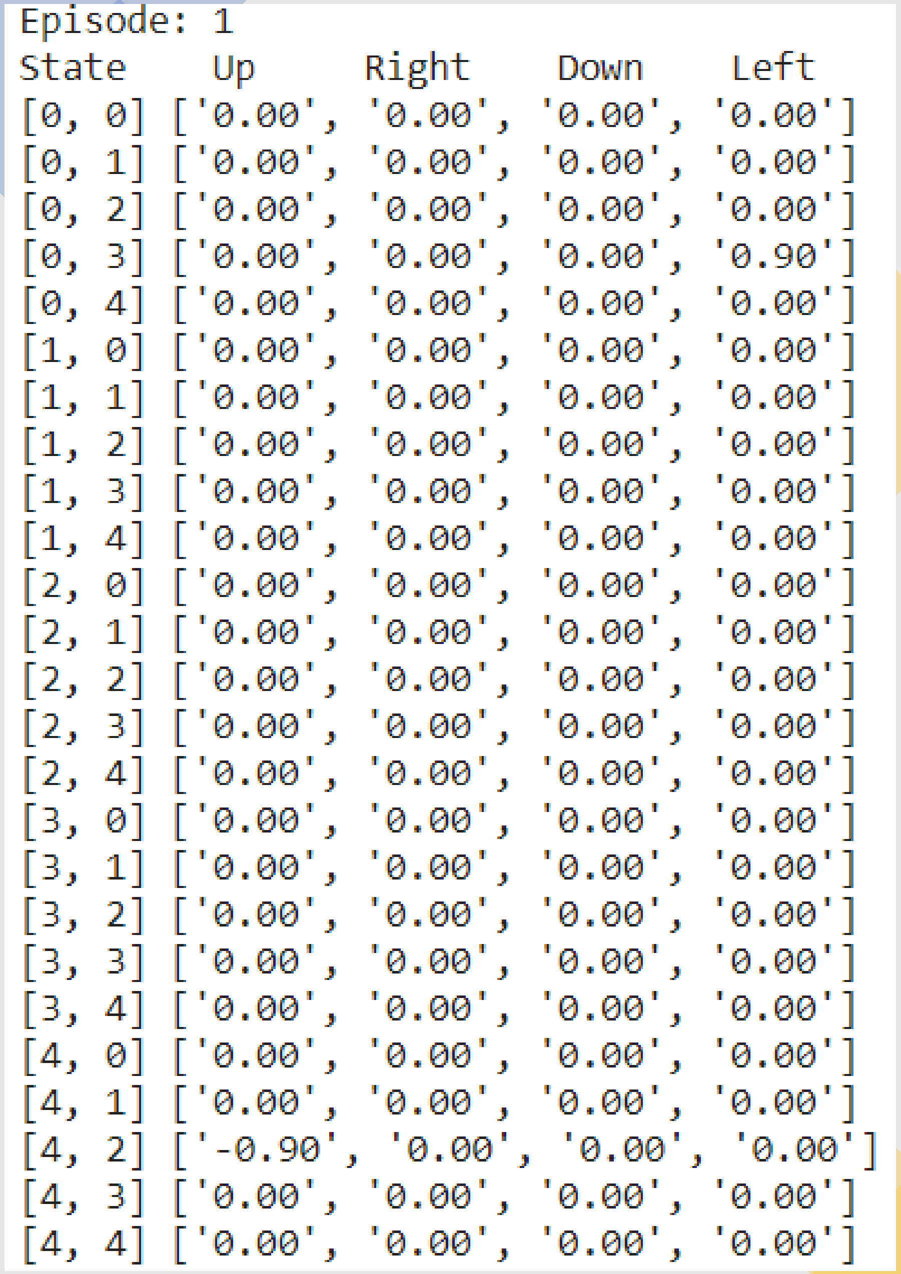
Result: 2. Total Reward Growth Curve (Bar Chart)





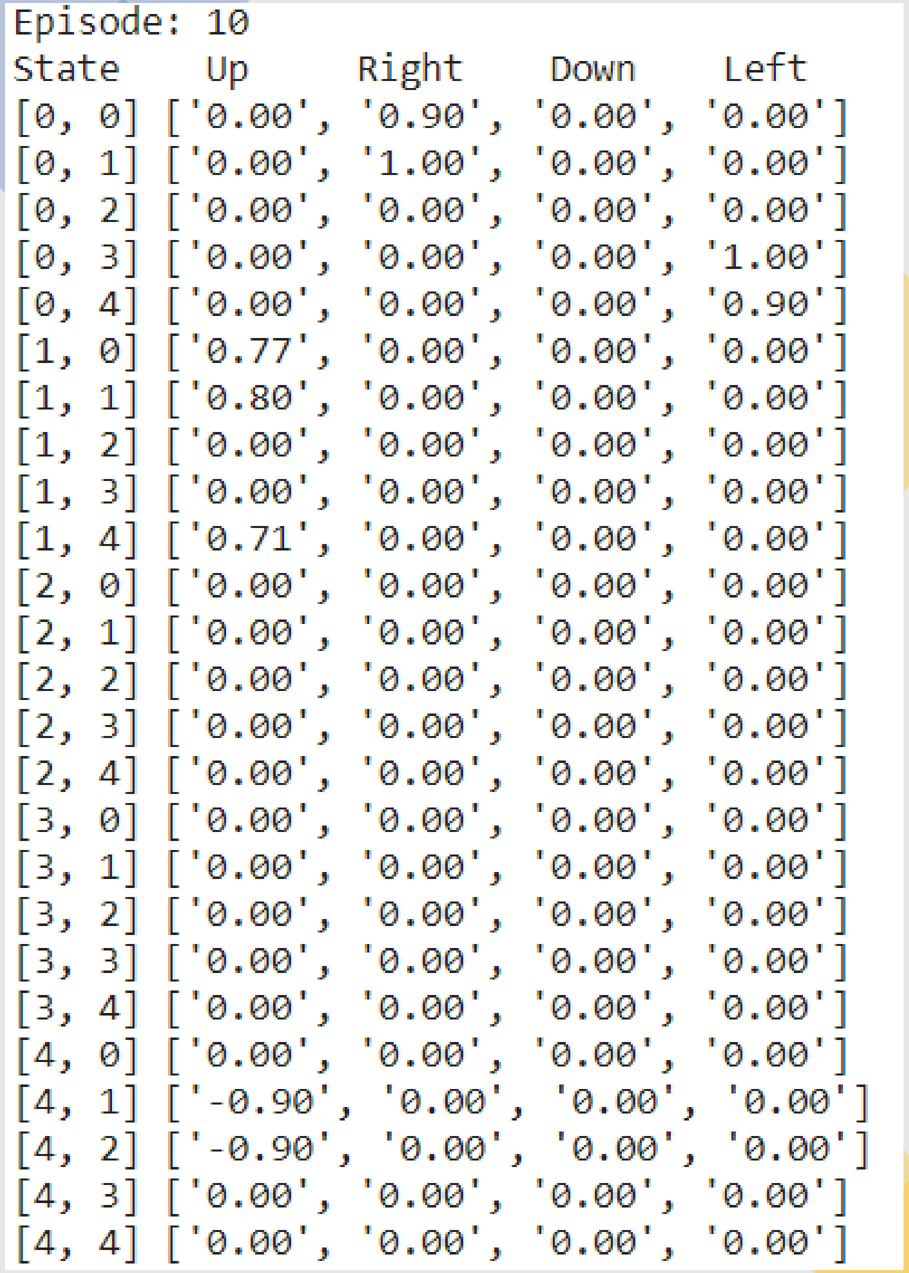
```
Episode: 0
State     Up       Right     Down     Left
[0, 0] ['0.00', '0.00', '0.00', '0.00']
[0, 1] ['0.00', '0.00', '0.00', '0.00']
[0, 2] ['0.00', '0.00', '0.00', '0.00']
[0, 3] ['0.00', '0.00', '0.00', '0.00']
[0, 4] ['0.00', '0.00', '0.00', '0.00']
[1, 0] ['0.00', '0.00', '0.00', '0.00']
[1, 1] ['0.00', '0.00', '0.00', '0.00']
[1, 2] ['0.00', '0.00', '0.00', '0.00']
[1, 3] ['0.00', '0.00', '0.00', '0.00']
[1, 4] ['0.00', '0.00', '0.00', '0.00']
[2, 0] ['0.00', '0.00', '0.00', '0.00']
[2, 1] ['0.00', '0.00', '0.00', '0.00']
[2, 2] ['0.00', '0.00', '0.00', '0.00']
[2, 3] ['0.00', '0.00', '0.00', '0.00']
[2, 4] ['0.00', '0.00', '0.00', '0.00']
[3, 0] ['0.00', '0.00', '0.00', '0.00']
[3, 1] ['0.00', '0.00', '0.00', '0.00']
[3, 2] ['0.00', '0.00', '0.00', '0.00']
[3, 3] ['0.00', '0.00', '0.00', '0.00']
[3, 4] ['0.00', '0.00', '0.00', '0.00']
[4, 0] ['0.00', '0.00', '0.00', '0.00']
[4, 1] ['0.00', '0.00', '0.00', '0.00']
[4, 2] ['-0.90', '0.00', '0.00', '0.00']
[4, 3] ['0.00', '0.00', '0.00', '0.00']
[4, 4] ['0.00', '0.00', '0.00', '0.00']
```

Result: 3. Q-table for Episode = 0



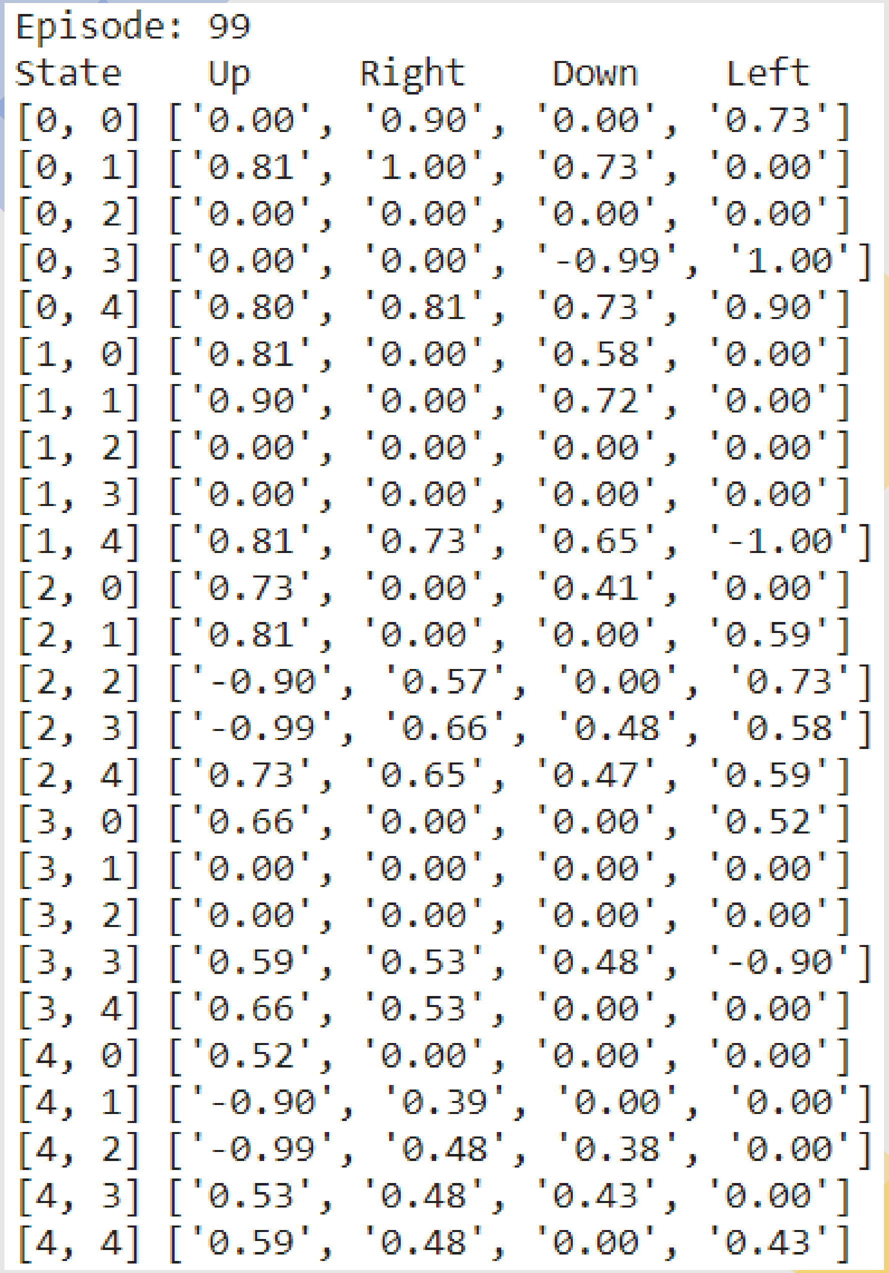
```
Episode: 1
State     Up      Right    Down     Left
[0, 0] ['0.00', '0.00', '0.00', '0.00']
[0, 1] ['0.00', '0.00', '0.00', '0.00']
[0, 2] ['0.00', '0.00', '0.00', '0.00']
[0, 3] ['0.00', '0.00', '0.00', '0.90']
[0, 4] ['0.00', '0.00', '0.00', '0.00']
[1, 0] ['0.00', '0.00', '0.00', '0.00']
[1, 1] ['0.00', '0.00', '0.00', '0.00']
[1, 2] ['0.00', '0.00', '0.00', '0.00']
[1, 3] ['0.00', '0.00', '0.00', '0.00']
[1, 4] ['0.00', '0.00', '0.00', '0.00']
[2, 0] ['0.00', '0.00', '0.00', '0.00']
[2, 1] ['0.00', '0.00', '0.00', '0.00']
[2, 2] ['0.00', '0.00', '0.00', '0.00']
[2, 3] ['0.00', '0.00', '0.00', '0.00']
[2, 4] ['0.00', '0.00', '0.00', '0.00']
[3, 0] ['0.00', '0.00', '0.00', '0.00']
[3, 1] ['0.00', '0.00', '0.00', '0.00']
[3, 2] ['0.00', '0.00', '0.00', '0.00']
[3, 3] ['0.00', '0.00', '0.00', '0.00']
[3, 4] ['0.00', '0.00', '0.00', '0.00']
[4, 0] ['0.00', '0.00', '0.00', '0.00']
[4, 1] ['0.00', '0.00', '0.00', '0.00']
[4, 2] ['-0.90', '0.00', '0.00', '0.00']
[4, 3] ['0.00', '0.00', '0.00', '0.00']
[4, 4] ['0.00', '0.00', '0.00', '0.00']
```

Result: 3. Q-table for Episode = 1



```
Episode: 10
State     Up      Right     Down     Left
[0, 0] ['0.00', '0.90', '0.00', '0.00']
[0, 1] ['0.00', '1.00', '0.00', '0.00']
[0, 2] ['0.00', '0.00', '0.00', '0.00']
[0, 3] ['0.00', '0.00', '0.00', '1.00']
[0, 4] ['0.00', '0.00', '0.00', '0.90']
[1, 0] ['0.77', '0.00', '0.00', '0.00']
[1, 1] ['0.80', '0.00', '0.00', '0.00']
[1, 2] ['0.00', '0.00', '0.00', '0.00']
[1, 3] ['0.00', '0.00', '0.00', '0.00']
[1, 4] ['0.71', '0.00', '0.00', '0.00']
[2, 0] ['0.00', '0.00', '0.00', '0.00']
[2, 1] ['0.00', '0.00', '0.00', '0.00']
[2, 2] ['0.00', '0.00', '0.00', '0.00']
[2, 3] ['0.00', '0.00', '0.00', '0.00']
[2, 4] ['0.00', '0.00', '0.00', '0.00']
[3, 0] ['0.00', '0.00', '0.00', '0.00']
[3, 1] ['0.00', '0.00', '0.00', '0.00']
[3, 2] ['0.00', '0.00', '0.00', '0.00']
[3, 3] ['0.00', '0.00', '0.00', '0.00']
[3, 4] ['0.00', '0.00', '0.00', '0.00']
[4, 0] ['0.00', '0.00', '0.00', '0.00']
[4, 1] ['-0.90', '0.00', '0.00', '0.00']
[4, 2] ['-0.90', '0.00', '0.00', '0.00']
[4, 3] ['0.00', '0.00', '0.00', '0.00']
[4, 4] ['0.00', '0.00', '0.00', '0.00']
```

Result: 3. Q-table for Episode = 10



```
Episode: 99
State     Up      Right     Down     Left
[0, 0] ['0.00', '0.90', '0.00', '0.73']
[0, 1] ['0.81', '1.00', '0.73', '0.00']
[0, 2] ['0.00', '0.00', '0.00', '0.00']
[0, 3] ['0.00', '0.00', '-0.99', '1.00']
[0, 4] ['0.80', '0.81', '0.73', '0.90']
[1, 0] ['0.81', '0.00', '0.58', '0.00']
[1, 1] ['0.90', '0.00', '0.72', '0.00']
[1, 2] ['0.00', '0.00', '0.00', '0.00']
[1, 3] ['0.00', '0.00', '0.00', '0.00']
[1, 4] ['0.81', '0.73', '0.65', '-1.00']
[2, 0] ['0.73', '0.00', '0.41', '0.00']
[2, 1] ['0.81', '0.00', '0.00', '0.59']
[2, 2] ['-0.90', '0.57', '0.00', '0.73']
[2, 3] ['-0.99', '0.66', '0.48', '0.58']
[2, 4] ['0.73', '0.65', '0.47', '0.59']
[3, 0] ['0.66', '0.00', '0.00', '0.52']
[3, 1] ['0.00', '0.00', '0.00', '0.00']
[3, 2] ['0.00', '0.00', '0.00', '0.00']
[3, 3] ['0.59', '0.53', '0.48', '-0.90']
[3, 4] ['0.66', '0.53', '0.00', '0.00']
[4, 0] ['0.52', '0.00', '0.00', '0.00']
[4, 1] ['-0.90', '0.39', '0.00', '0.00']
[4, 2] ['-0.99', '0.48', '0.38', '0.00']
[4, 3] ['0.53', '0.48', '0.43', '0.00']
[4, 4] ['0.59', '0.48', '0.00', '0.43']
```

Result: 3. Q-table for Episode = 99

Conclusion

- As the episodes increase the cumulative rewards fluctuate but have an increasing trend. That is the training model is updating each state to q-values that would optimize the path to reach the goal
- The Q-table shows that in early episodes of training, there is limited to none updates to the q-values but it gradually updates to the values that helps in finding the path for the robot through the most promising states towards the goal.
- The shortest path discovered by this model from the given state of [4,2] of the robot to the charging dock area was
  - [[4, 2], [4, 3], [3, 3], [2, 3], [2, 4], [1, 4], [0, 4], [0, 3], [0, 2]]