ML ASSIGNMENT 3 REPORT – FINAL

TEAM MEMBERS

1. NEEMA GEORGE (neemageo) - UB PERSON NUMBER : 50545746

2. SRI SAHITH REDDY KUNCHARAM (srisahit) - UB PERSON NUMBER: 50532516

PART 1

1. We have defined RL Environment

Theme : MarathonEnvironment Grid World works with energy drinks and the medal as positive rewards and the stone as negative reward. The runner achieving the medal is the final condition.

State : {S1 = (0,0), S2 = (0,1), S3 = (0,2), S4 = (1,0), S5 = (1,1), S6 = (1,2), S7 = (2,0), S8 = (2,1), S9 = (2,2), S10 = (3,0), S11 = (3,1), S12 = (3,2)}

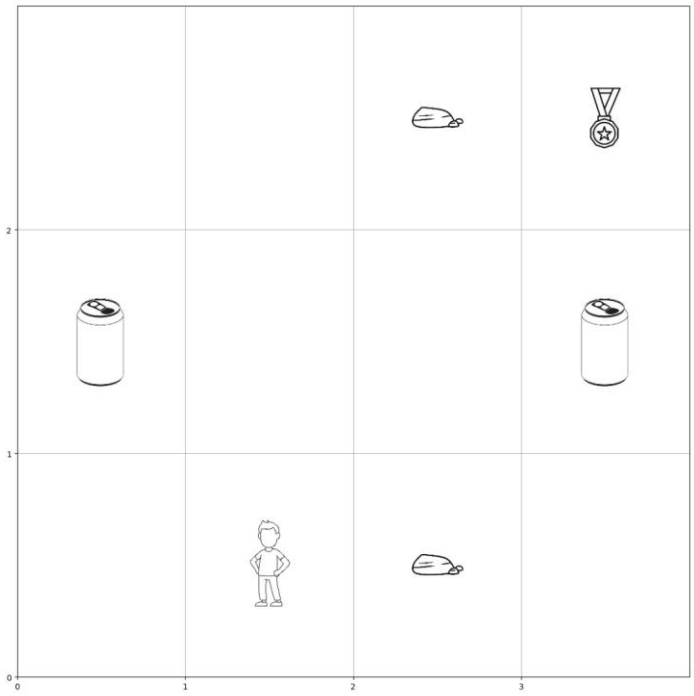
Actions : {Up, Down, Right, Left}

Rewards : {-2, +5, +10}

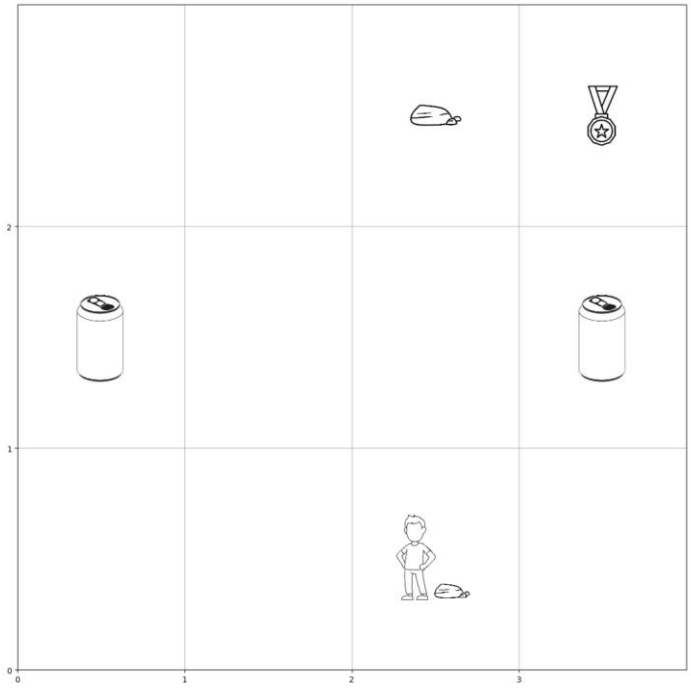
2. We define the environment with the requirements following the Gymnasium structure as mentioned above.

3. We run the agent for 10 timesteps to show the accuracy of the environment logic. Below are the screenshots of the same.

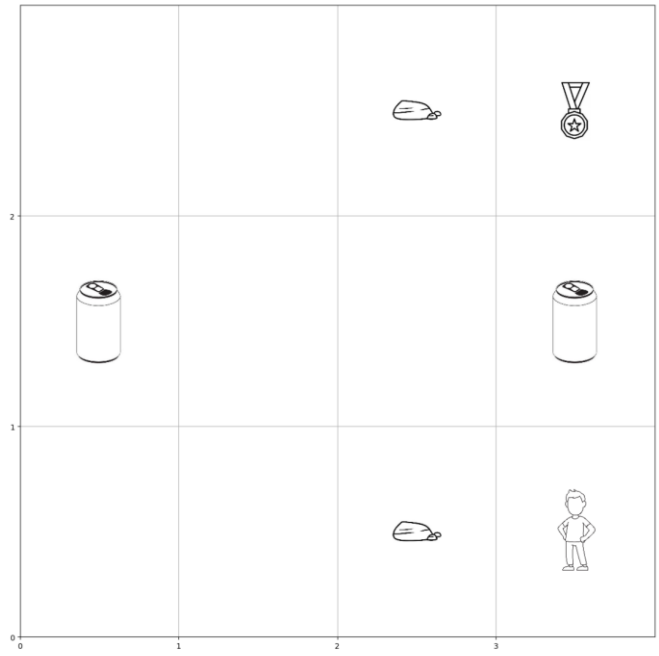
```
Time step 1:
Current State: 0
Chosen Action: 0
Reward: 0.0
```



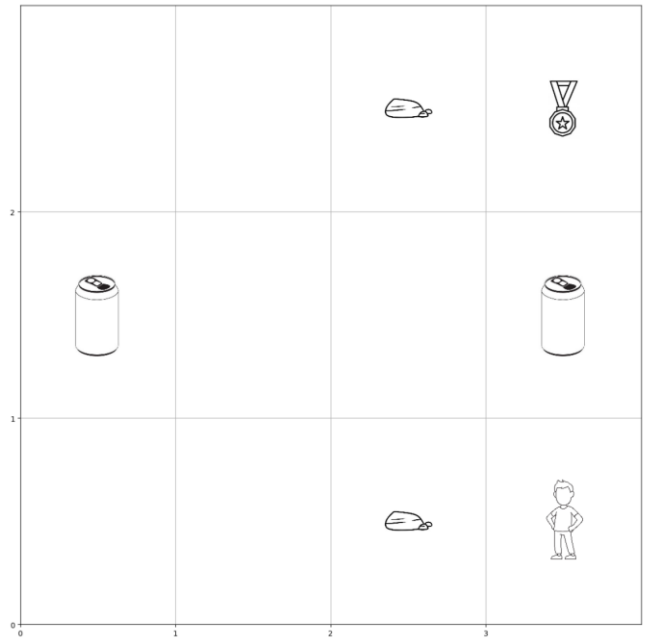
Time step 2:
Current State: 0
Chosen Action: 0
Reward: -2.0



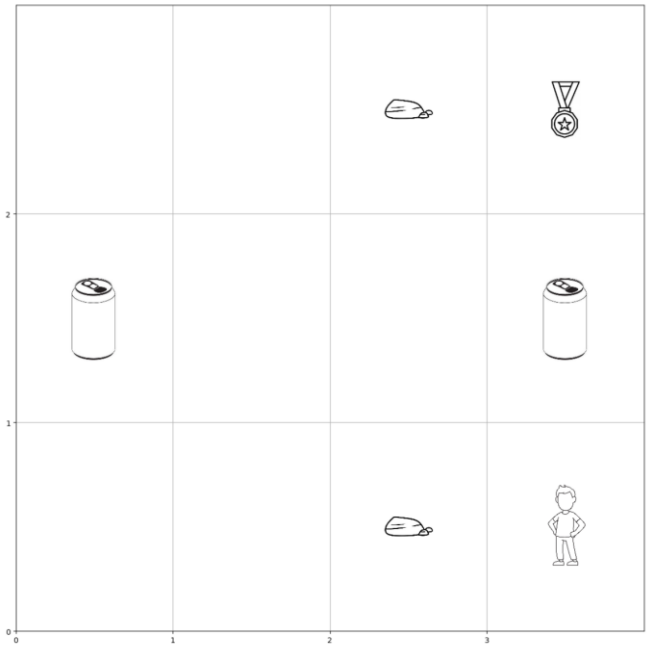
Time step 3:
Current State: 0
Chosen Action: 0
Reward: -2.0



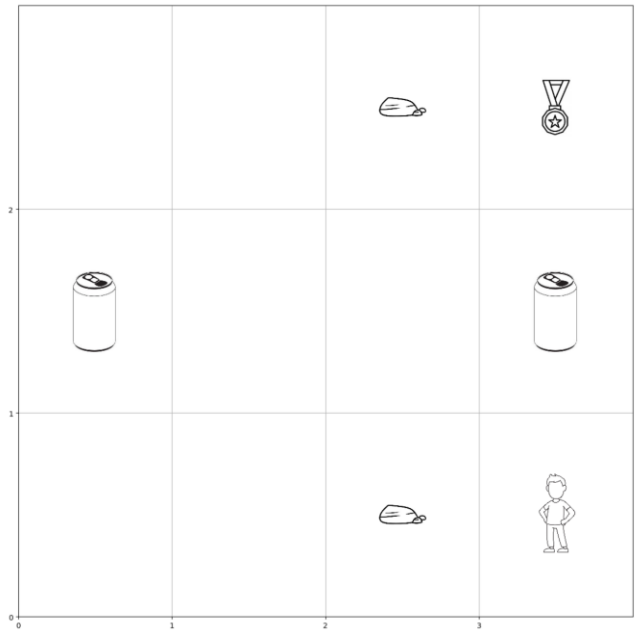
```
Time step 4:
Current State: 0
Chosen Action: 0
Reward: -2.0
```



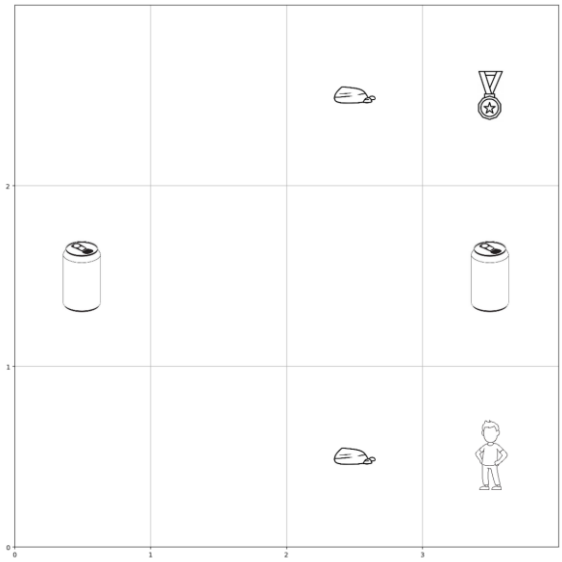
```
Time step 5:
Current State: 0
Chosen Action: 3
Reward: -2.0
```



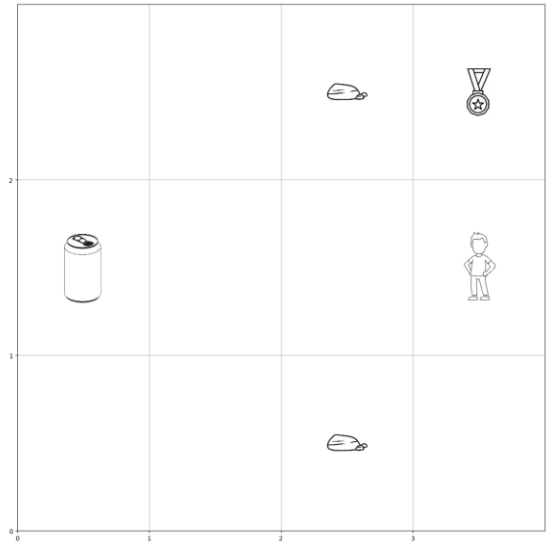
```
Time step 6:
Current State: 0
Chosen Action: 0
Reward: -2.0
```



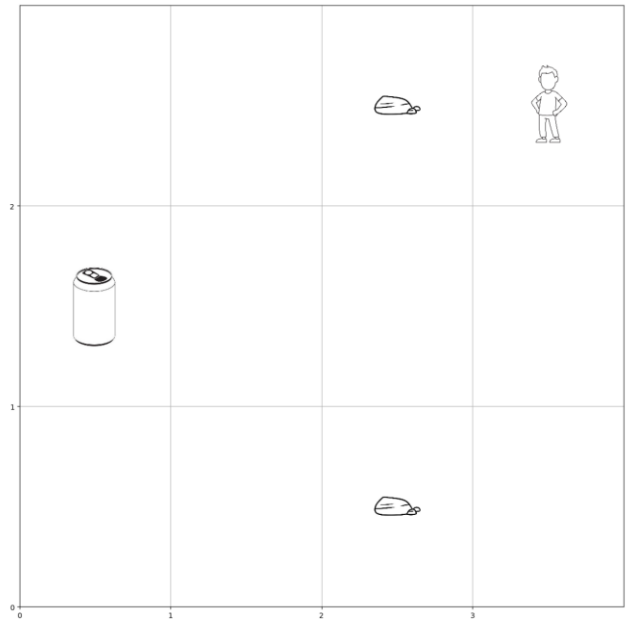
```
Time step 7:
Current State: 0
Chosen Action: 3
Reward: -2.0
```



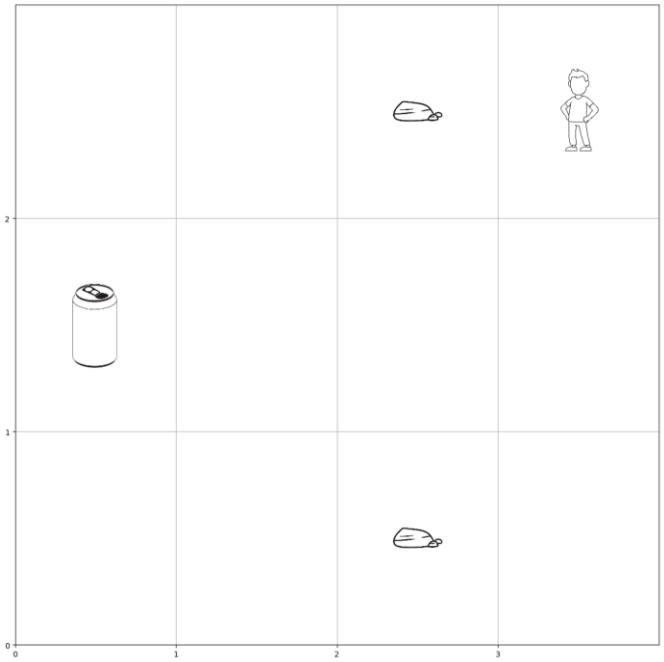
Time step 8:
Current State: 0
Chosen Action: 2
Reward: 3.0



Time step 9:
Current State: 0
Chosen Action: 2
Reward: 13.0



```
Time step 10:
Current State: 0
Chosen Action: 2
Reward: 23.0
```



Safety in AI:

We ensure the safety of the environment by implementing logic to avoid the agent from moving out of the grid at corners and boundaries by the following implementations:

- The grid is defined with a width and height of 4 and 3
- Before updating the state and position, we check if the movement will keep the

agent within the grid boundaries. After the movement, you use the max and min

functions to ensure the agent's position stays within the boundaries:

- self.agent_pos[0] = max(0, min(self.agent_pos[0], max_x))
- self.agent_pos[1] = max(0, min(self.agent_pos[1], max_y))
- This ensures that the agent remains within the confines of the grid, regardless of

the action taken

PART 2

Tabular Methods

SARSA:

SARSA is a tabular Q-learning technique that involves the agent learning a Q-table to approximate the value of taking specific actions in given states.

Choosing Actions: The choose_action function is employed to decide the action based on the current state and Q-table values. It randomly explores actions with epsilon probability and selects the action with the highest Q-value with a probability of (1-epsilon).

Update Rule: The Q-values are updated using the SARSA update rule: Q[state, action] = Q[state, action] + alpha * (target - predict). Here, 'alpha' represents the learning rate, 'target' is the estimated optimal future value, and 'predict' is the predicted value.

Learning Process: The doSARSALearn function implements the learning process across episodes. It initializes the Q-table with zeroes and updates the table as the agent interacts with the environment.

Advantages of SARSA:

SARSA is an on-policy algorithm, enabling learning during exploration.

It's relatively simple and easy to implement.

SARSA tends to be conservative, avoiding high negative rewards near optimal paths during exploration.

It can be combined with neural networks for learning complex policies in high-dimensional environments.

Disadvantages of SARSA:

Sensitivity to hyperparameters like learning rate, alpha, and gamma.

Slower convergence in larger environments toward the optimal policy.

Risk of overfitting if the learning rate is excessively high.

Double Q-Learning:

Double Q-learning is a tabular method where the agent learns from two Q-tables to estimate action values in states.

Action Selection: The choose_action function selects actions based on both Q-table values. It explores randomly with epsilon probability and chooses the action with the highest sum of Q-values with probability (1-epsilon).

Update Method: The update function updates the chosen Q-table with a 0.5 probability using the equation: Q[state, action] = Q[state, action] + alpha * (target - predict). Here, 'alpha' is the learning rate, 'target' represents the estimated optimal future value, and 'predict' is the predicted value.

Learning Process: The doDoubleQLearn function initializes two Q-tables with zeroes and iterates through episodes, updating the selected Q-table based on the agent's interactions**.**

Advantages of Double Q-learning:

Model-free and doesn't require knowledge of the complete environment model.

Faster convergence to an optimal policy compared to SARSA.

Effective in handling large state stationary environments**.**

Disadvantages of Double Q-learning:

Slowness when the number of states is high and rewards are sparse.

Unreliability due to being an off-policy algorithm learning from a different policy than the target policy.

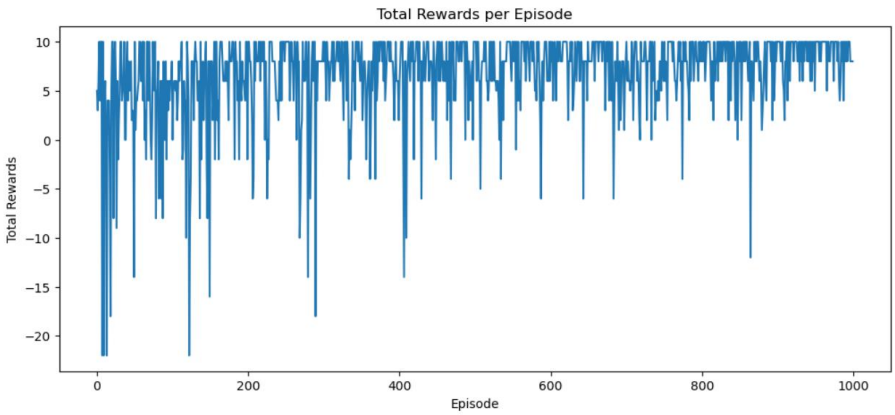
Poor performance in non-stationary environments.

Now, we apply the SARSA algorithm to solve the environment.

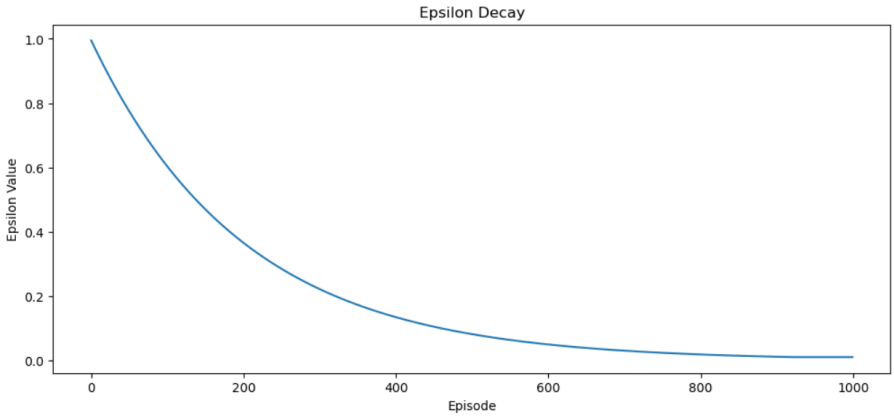
The initial Q table and Q table after training are as below:

```
Initial Q-table:
[[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.]]

Trained Q-table:
[[ 0.26203443  0.50936541  1.42463074  0.25353211]
 [-1.48629758  0.48611695  0.19489249  0.11391455]
 [ 0.37680973  0.37983507  0.55526899 -0.79867746]
 [ 0.1515098  -1.39109567  0.35096999  0.13844038]
 [ 0.33312766  0.38472722  0.21633476  1.54444872]
 [ 0.47840712  0.25376945  0.21146418  0.08244057]
 [ 0.83453845  0.33753502  0.24415371 -1.47708773]
 [ 0.31200109  0.22080552  0.          0.17932695]
 [ 0.25850858  0.11230548  0.14743316  0.13539263]
 [ 0.17827329  0.17833076  0.15529265  0.39428324]
 [ 0.          0.2511177   0.17799181  0.13624194]
 [ 0.          0.          0.          0.        ]]
```

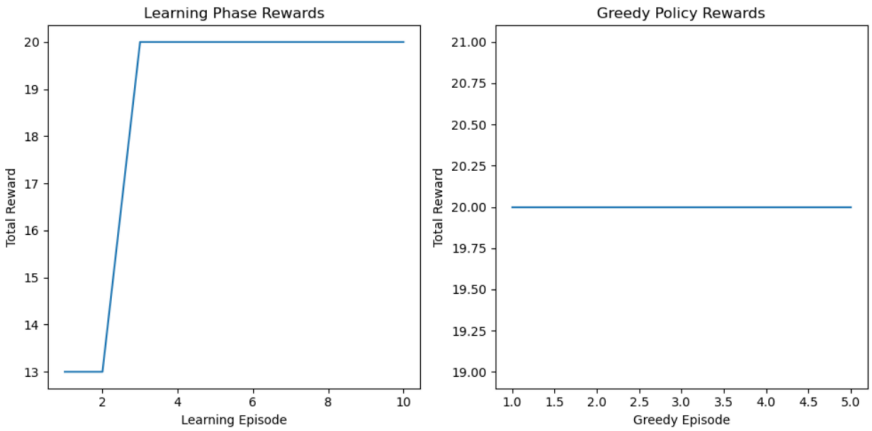
The total reward per episode graph is as below



The epsilon decay graph is as below:



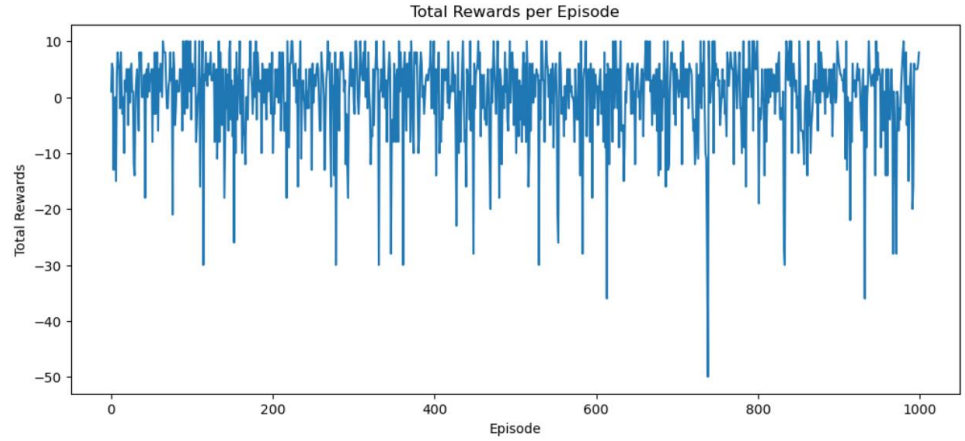
On running the environment for at least 10 episodes, where the agent chooses only greedy actions from the learned policy we obtain the following graphs

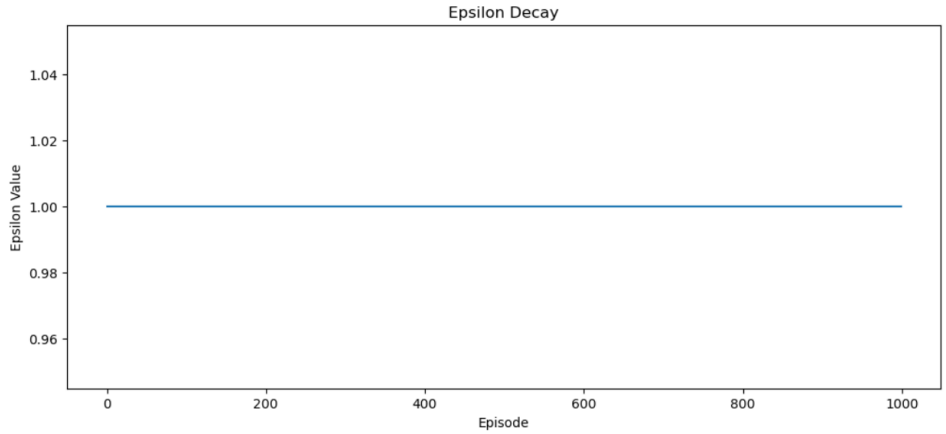


It can be seen from the above graph that the agent only chose the steps that led to the reward of 20. The maximum reward the agent was able to achieve with the current learned policy is 20. For all the episodes, the reward was 20.

Now we will apply the hyperparameter tuning,

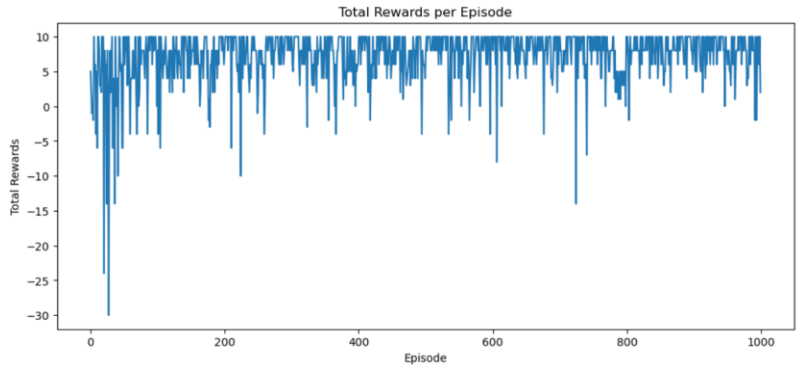
a) Epsilon Decay Rate = 1

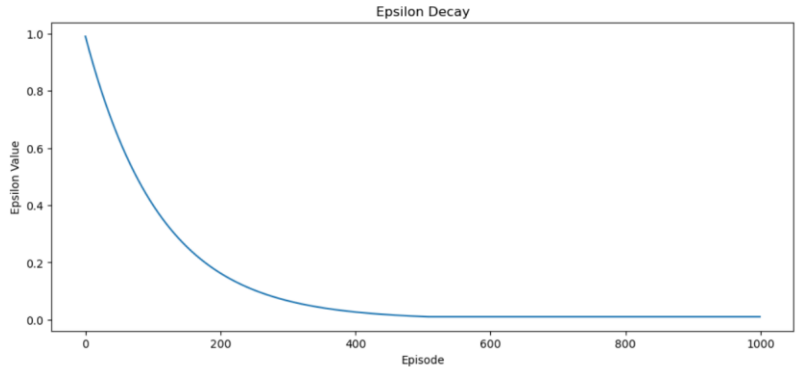





```
Initial Q-table:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

Trained Q-table:
[[-0.77129558  0.6016398   1.98928913  0.49051333]
 [-3.12686365  0.38360646 -0.48860654 -0.99103417]
 [-1.17323789 -0.73758084 -1.20568777 -3.24404759]
 [-0.52464556 -3.20626803  1.0980498  -1.43480443]
 [-0.3588477  -0.03780298 -0.04495862  0.26689592]
 [-0.17489469  0.75332808 -0.19376942 -0.89584295]
 [ 2.39487646 -0.50293314 -0.33694793 -3.33733308]
 [-0.37304025 -0.69549925  0.         -0.93203887]
 [-0.22531132 -0.05361675 -0.01644727  0.03990643]
 [-0.3913305   0.02279999 -0.29775105 -0.51763151]
 [ 0.         -0.25368431 -0.44960295 -0.79513301]
 [ 0.          0.          0.          0.        ]]
```

b) Epsilon Decay Rate = 0.991

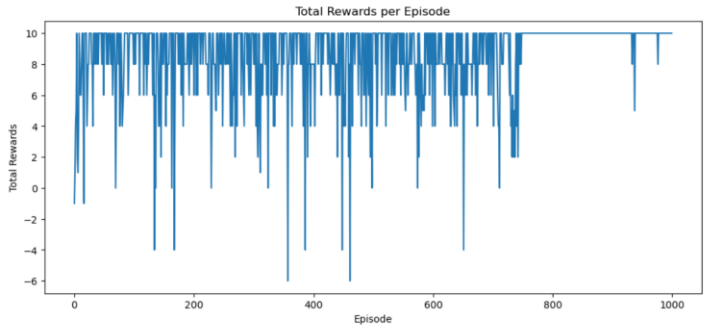


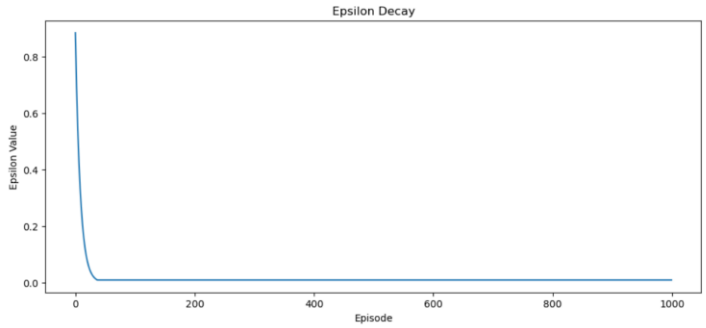


While we consistently achieved around 20 as the maximum reward previously, this recent trial only reached approximately 15. From this outcome, it's apparent that the chosen epsilon decay value didn't lead to optimal results.

```
Initial Q-table:
[[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.]]

Trained Q-table:
[[ 0.04285342  0.15980003  0.17331486  0.17340809]
 [-1.53898146  0.16071741  0.17324219  0.12013129]
 [ 0.26718328  0.26023555  0.43773178 -1.05912219]
 [ 0.14005274 -1.56926887  0.36137762  0.15738706]
 [ 0.17492802  0.17340418  0.17345003  0.17339636]
 [ 0.19586054  0.17351577  0.17364107  0.17365015]
 [ 0.26705068  0.24061006  0.26483041 -1.50687601]
 [ 0.19157458  0.09726104  0.          0.17674026]
 [ 0.17331066  0.17321033  0.17330167  0.1732452 ]
 [ 0.1733201   0.17310851  0.17342986  0.17261863]
 [ 0.          0.17302569  0.17293055  0.15904649]
 [ 0.          0.          0.          0.       ]]
```

c) Epsilon Decay Rate = 0.885





The result obtained was better compared to previous trials. We got a maximum total reward of 25 which is greater than the 20 we were getting in other trials.

```
Initial Q-table:
[[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.]]

Trained Q-table:
[[ 1.61011904e-01  2.43323144e-01  9.54494194e+00  4.16479408e-01]
 [-1.70489827e+00  9.98356466e-02  9.94998293e-01  3.14995653e-02]
 [ 5.61844512e-01  1.03254560e-02  1.24955827e-02 -1.29612206e+00]
 [-4.18348565e-02 -1.76152518e+00  2.11326863e+00  7.44810313e-02]
 [ 4.72402744e+00  7.23212150e-01  1.17566828e-01  2.61974691e+00]
 [ 4.88440290e+00  7.26479919e-02  1.70239004e-01  9.98202118e-02]
 [ 4.99999951e+00  9.33049531e-02  1.02588030e-01 -1.79046491e+00]
 [-1.07191226e-03 -2.65123302e-02  0.00000000e+00 -6.65086268e-02]
 [ 3.37089559e-01  7.25198134e-02  7.25645112e-02  7.25525680e-02]
 [ 6.88583082e-02  7.25618624e-02  7.25438453e-02  1.83690321e+00]
 [ 0.00000000e+00  2.13030469e-01  7.25799643e-02  7.15130273e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```

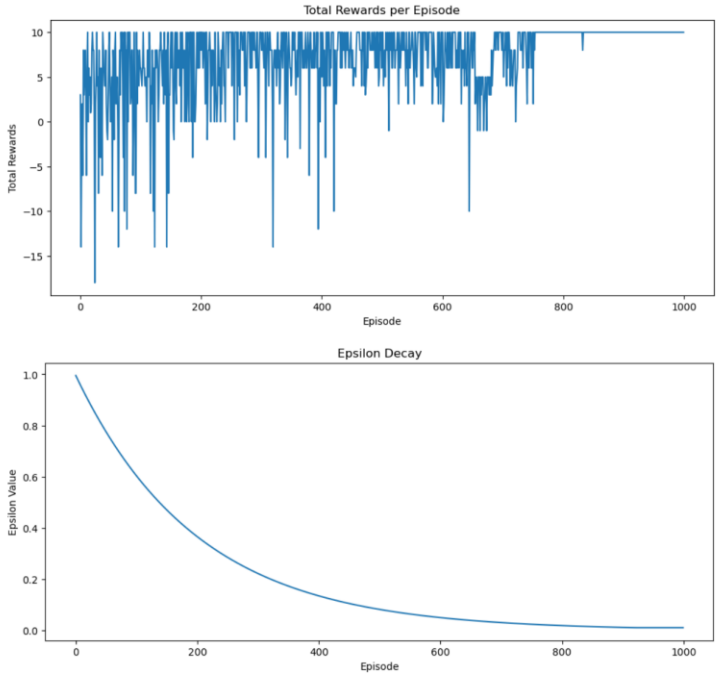
One parameter, gamma, was adjusted from 0.96 to 0.92. This change implies that the agent places greater emphasis on immediate rewards. Consequently, the agent becomes more inclined to explore new states and actions, even if they carry inherent risks.

Comparing the plots between gamma values of 0.96 and 0.92 revealed that the total rewards per episode were higher with a gamma value of 0.92. This shift occurred because the agent, now prioritizing immediate rewards, tended to choose actions that yielded immediate benefits, disregarding their long-term impact on total rewards.

d) Discount Factor = 0.96

With gamma as 0.96, we got the total reward per episode when the agent takes

greedy steps from the learned policy as

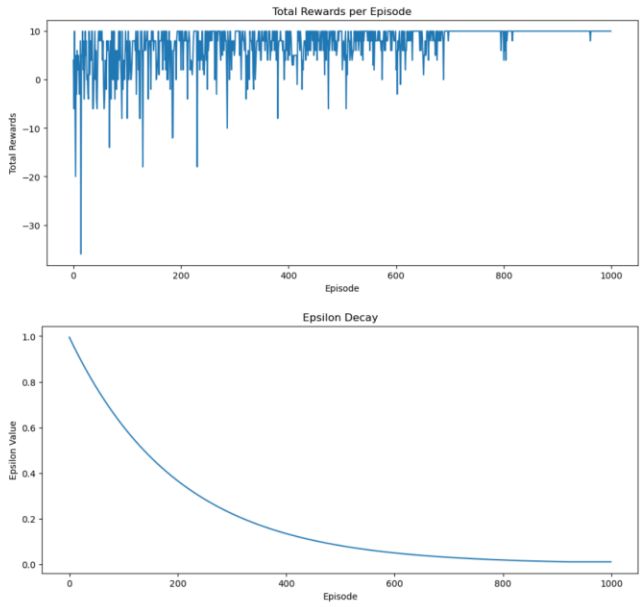


Observation:

The maximum total reward achieved was improved to 25. A significant change from other tries was achieved. The different gamma values we chose are 0.96, 0.94, 0.92. Out of these gammas with 0.96 we got the highest total reward, which is 225. The increased reward is due to the agent trying to explore new states and actions. With the gamma =0.96 the agent values future rewards more heavily. Even though there is risk, the agent tries to explore new states and actions which leads to high rewards in the longer term. So, this makes discovery of new and better policies.

```
Initial Q-table:
[[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.]]

Trained Q-table:
[[ 4.92923485e-02  1.57626137e+00  9.33018825e+00  5.24436504e-02]
 [-1.68685645e+00  1.91988106e+00  3.35734195e-02 -1.11742476e-01]
 [ 1.39171309e-01  1.52966843e-01  1.37908581e-01 -1.44560582e+00]
 [-2.44281194e-02 -1.75496276e+00  4.56176129e-01 -1.02044215e-03]
 [ 4.45167424e+00  5.24494645e-02  5.24016852e-02  5.25393892e-02]
 [ 4.67840668e+00  5.72728328e-02  9.33287586e-02 -1.17186956e-02]
 [ 4.99952902e+00  1.07006124e+00  9.84205988e-02 -1.79747085e+00]
 [-1.15893368e-02 -3.32277965e-02  0.00000000e+00 -1.36387841e-01]
 [ 5.32112750e-02  3.28524758e-02  3.54974828e-02  4.76398391e-01]
 [ 3.28261062e-02  3.28686253e-02  3.87772418e-02  1.81838082e+00]
 [ 0.00000000e+00  4.02986495e-01  3.25375209e-02 -6.65371712e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```

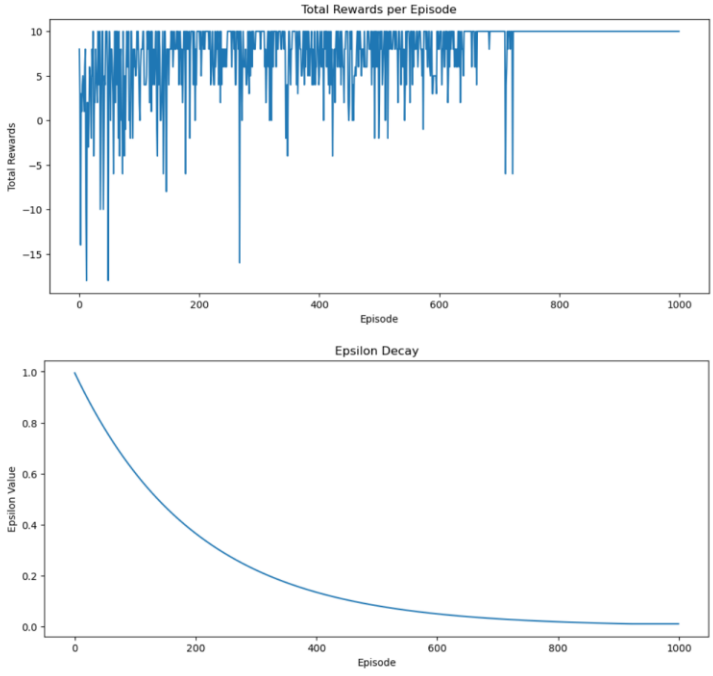
e) Discount Factor = 0.94



With gamma as 0.94, we got the total reward per episode when the agent takes greedy steps from the learned policy as shown above. The maximum total reward achieved was again 20. No change from the try with a gamma value of 0.92

```
Initial Q-table:
[[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.]]

Trained Q-table:
[[ 1.61946497e-01  1.64996362e-01  7.15703753e+00  1.56427265e-01]
 [-1.62462217e+00  1.81291607e-01  8.20621379e-02  8.90060142e-02]
 [ 2.57203050e-01  2.49291146e-01  6.31130328e-01 -1.40424984e+00]
 [-5.82240382e-03 -1.64285889e+00  4.12945814e-01 -6.14736614e-02]
 [ 3.32901564e+00  1.35491978e-01  1.35611393e-01  1.31070987e-01]
 [ 4.26397880e+00  1.40021194e-01  8.80436472e-02  1.82700235e-01]
 [ 4.91110132e+00  1.41508502e-01  6.42994183e-02 -1.56863377e+00]
 [-8.50104402e-03 -2.06439377e-02  0.00000000e+00 -1.72726084e-03]
 [ 8.78336103e-02  8.94070882e-02  6.14334755e-02  1.55344500e-01]
 [ 4.84001793e-02  2.37658280e-01  4.80559995e-02  4.70983326e-02]
 [ 0.00000000e+00  5.09959970e-02  4.82928282e-02 -6.70281690e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```

f) Discount Factor = 0.92



It can be seen that the maximum total reward achieved was 20

```
Initial Q-table:
[[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.]]

Trained Q-table:
[[-6.95372492e-02  8.45557880e-01  8.88151137e+00  8.37970334e-01]
 [-1.62121892e+00  5.16899836e-02  5.23745940e-02  5.35033860e-02]
 [ 1.60298158e-01  1.45182610e-01  3.50334185e-01 -1.61384093e+00]
 [ 1.11038328e-02 -1.78341815e+00  7.58069316e-02  4.25169479e-02]
 [ 4.23073423e+00  4.01432247e-01  9.68095419e-02  7.18891658e-01]
 [ 4.59983246e+00  2.36262155e-02  2.36322329e-02 -9.71218847e-02]
 [ 4.99999994e+00  4.67614574e-01  6.26245011e-02 -1.52832984e+00]
 [-2.17158223e-03 -2.29408118e-02  0.00000000e+00 -1.05362662e-02]
 [ 2.32864833e-02  2.32827389e-02  2.30315905e-02  8.81694582e-01]
 [ 2.32593301e-02  2.31629550e-02  2.32736883e-02  2.31693593e-02]
 [ 0.00000000e+00  2.30584892e-02  2.31949955e-02  1.60096312e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```

Best hyperparameters for Q learning

From all the above experiments on the Double Q-Learning model, the best parameter set we found was that with parameters: gamma=0.96 and epsilon decay=0.885, we got the best result of total reward 25.
Parameters used:
Epsilon = 1
Episode count = 100
Learning rate = 0.1
Discount factor = 0.96
Epsilon decay = 0.885

PART 3

We modify the code from the previous part and implement the double Q learning method.

```python
class DoubleQLearningAgent:
    def __init__(self, state_space, action_space, alpha, gamma, epsilon, epsilon_decay, epsilon_min):
        self.q_table_1 = np.zeros((state_space, action_space))
        self.q_table_2 = np.zeros((state_space, action_space))
        self.alpha = alpha
        self.gamma = gamma
        self.epsilon = epsilon
        self.epsilon_decay = epsilon_decay
        self.epsilon_min = epsilon_min
        self.action_space = action_space

    def select_action(self, state):
        if np.random.rand() < self.epsilon:
            return np.random.randint(self.action_space)
        else:
            return np.argmax(self.q_table_1[state] + self.q_table_2[state])

    def update(self, state, action, reward, next_state, done):
        if np.random.rand() < 0.5:
            best_next_action = np.argmax(self.q_table_1[next_state])
            td_target = reward + self.gamma * self.q_table_2[next_state][best_next_action] * (not done)
            td_error = td_target - sel (variable) gamma: Any on]
            self.q_table_1[state][acti... ... td_error
        else:
            best_next_action = np.argmax(self.q_table_2[next_state])
            td_target = reward + self.gamma * self.q_table_1[next_state][best_next_action] * (not done)
            td_error = td_target - self.q_table_2[state][action]
            self.q_table_2[state][action] += self.alpha * td_error

        if done:
            self.epsilon = max(self.epsilon_min, self.epsilon * self.epsilon_decay)

    def end_episode(self):
        self.epsilon = max(self.epsilon_min, self.epsilon * self.epsilon_decay)
```

```python
env = MarathonEnvironment(environment_type='deterministic')

state_space = env.observation_space.n
action_space = env.action_space.n
alpha = 0.1
gamma = 0.99
epsilon = 1.0
epsilon_decay = 0.995
epsilon_min = 0.01
agent = DoubleQLearningAgent(state_space, action_space, alpha, gamma, epsilon, epsilon_decay, epsilon_min)

episodes = 1000
for episode in range(episodes):
    state = env.reset()
    done = False
    while not done:
        action = agent.select_action(state)
        next_state, reward, done, _ = env.step(action)
        agent.update(state, action, reward, next_state, done)
        state = next_state
```

We now print the initial Q-tables and the trained Q-tables

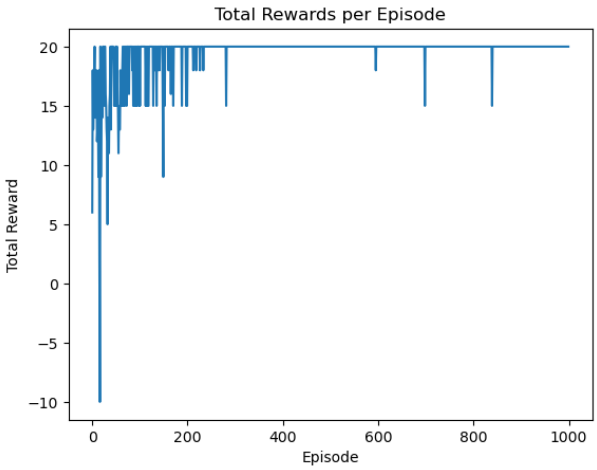
Initial Tables

```
Initial Q-Table 1:
[[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.]]

Initial Q-Table 2:
[[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.]]
```

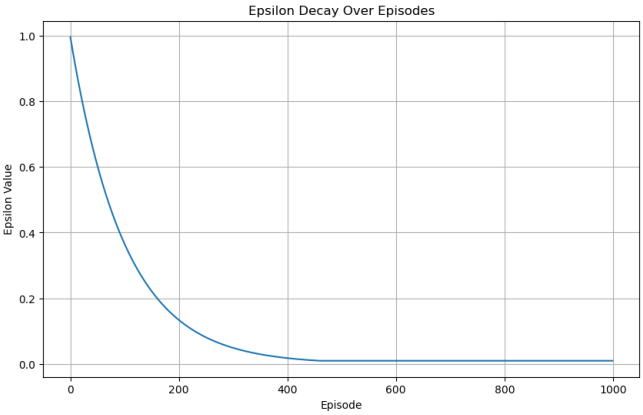
Trained Q-tables

```
Trained Q-Table 1:
[[ 6.23101784 10.63941593 19.45745506 11.98526141]
 [ 0.75785675 11.37122607  1.73870017  1.16312207]
 [ 1.32142862  1.22208466 10.35031297 -0.59616334]
 [ 1.37534015 -0.26745653  7.91221964  1.33913738]
 [14.57970591  6.8731079   3.25890421  9.9298381 ]
 [14.751       5.83933829  3.56528192  3.31111665]
 [14.89910004  9.69345552  5.48696147  2.25541152]
 [ 6.91729294  9.27643222 10.          3.57273084]
 [ 0.59828637  1.12187407  0.61384531  7.38543108]
 [ 6.7378127   0.57858574  0.91606409  2.10510772]
 [ 9.0152291   0.91636379  3.71533958  0.92539441]
 [ 0.          0.          0.          0.        ]]

Trained Q-Table 2:
[[ 5.3207696   9.36206301 19.41525839  6.72002356]
 [ 0.16334902 12.54724519  2.54746037  1.75274683]
 [ 1.13442306  1.35729841  9.36542383 -0.57033199]
 [ 1.96736567  0.61985853  6.74258724  0.82943761]
 [14.60349     7.76127049  3.43849141  9.15869163]
 [14.74308699  7.21051048  2.62578551  3.82350566]
 [14.9         5.9572618   5.28987505  3.16903768]
 [ 7.03150883  8.95888577 10.          4.58074477]
 [ 0.64803781  0.98632686  0.82127312  6.98166185]
 [ 6.20209746  0.79211411  1.39914853  2.71800995]
 [ 9.52898713  1.47235697  2.21312684  0.54086998]
 [ 0.          0.          0.          0.        ]]
```

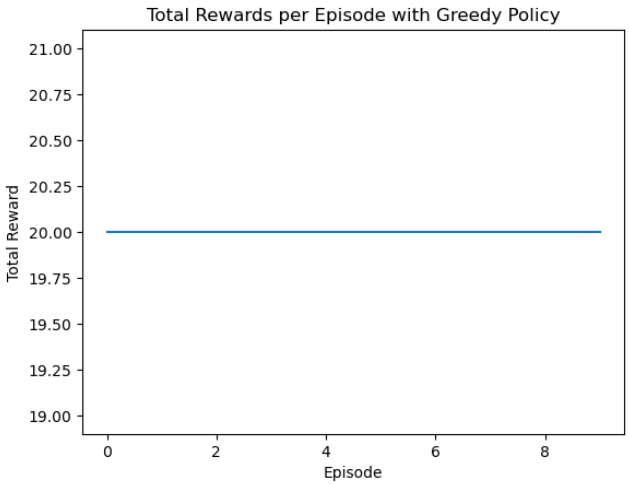
We plot the total reward per episode graph (x-axis: episode, y-axis: total reward per episode).



We plot the epsilon decay graph (x-axis: episode, y-axis: epsilon value)

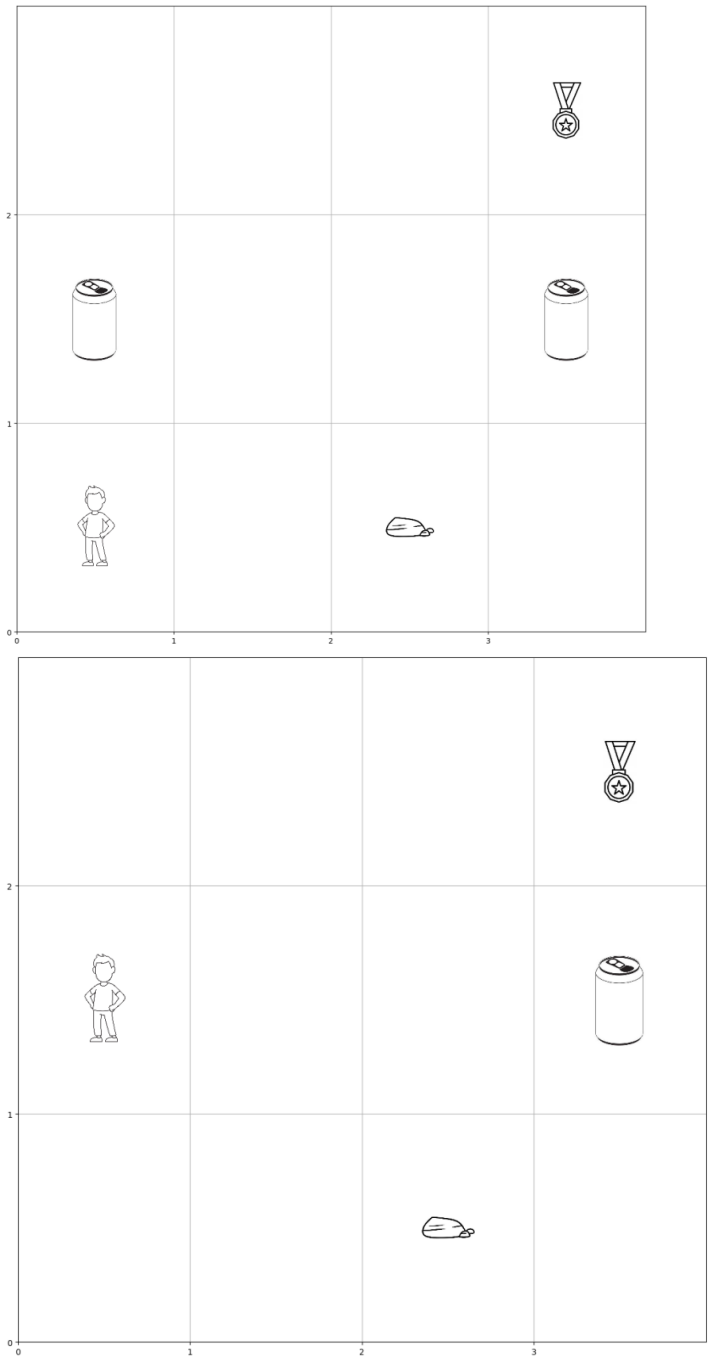


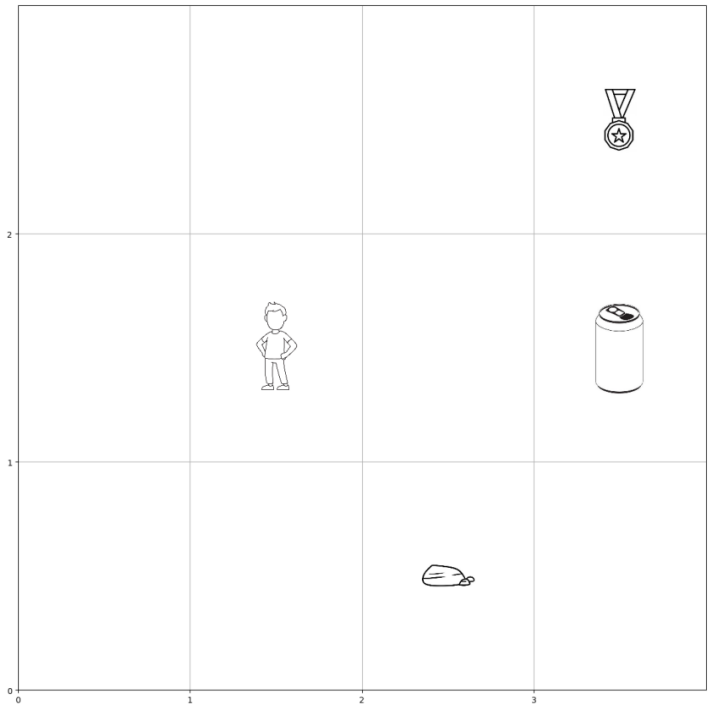
We run the environment for at least 10 episodes, where the agent chooses only greedy actions from the learned policy including a plot of the total reward per episode.

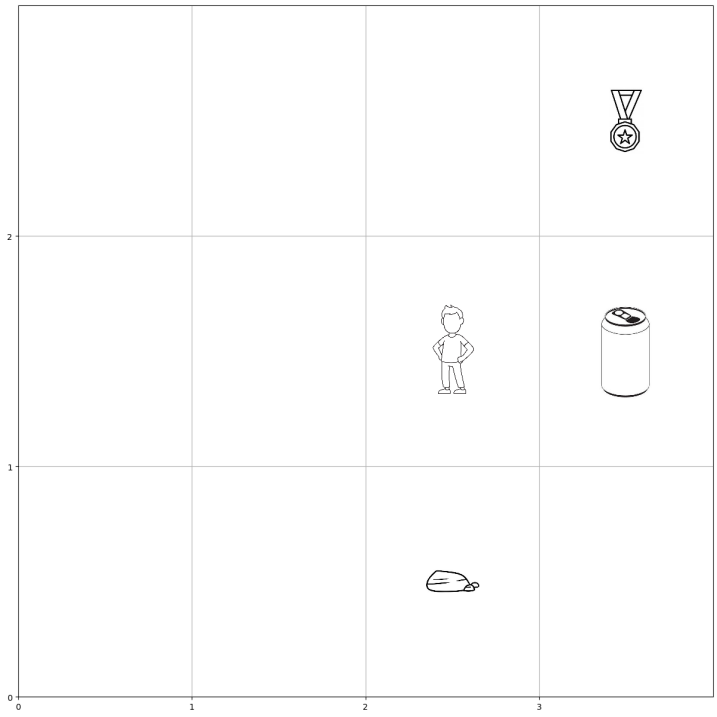


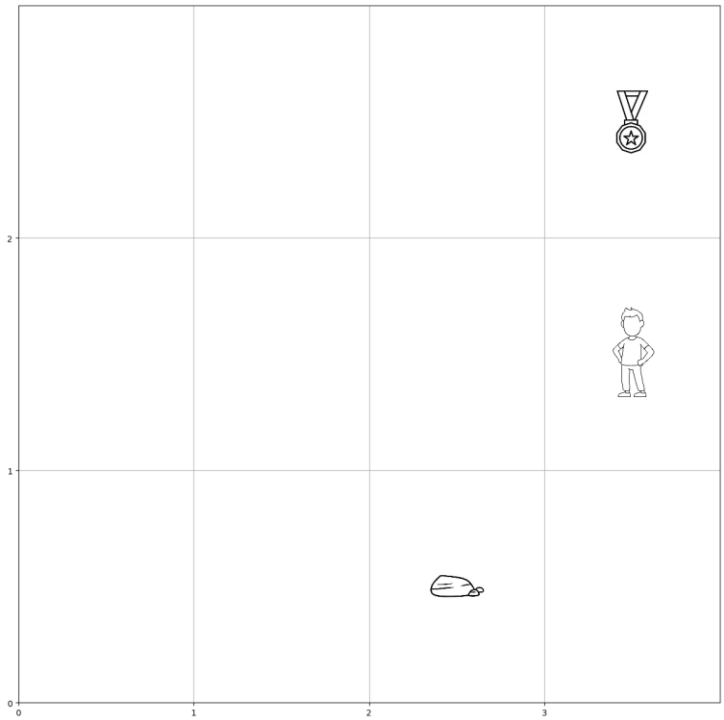
It can be seen from the above graph, that the agent only chose the steps that led to the reward of 20. The maximum reward the agent was able to achieve with the current learned policy is 20. For all 10 episodes, the reward was 20.

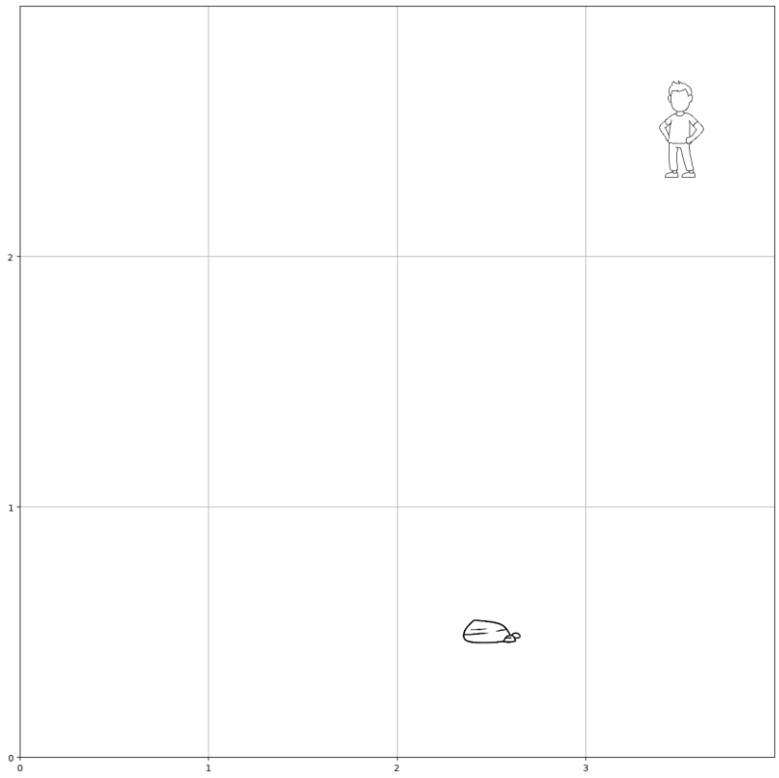
The steps are as shown below :

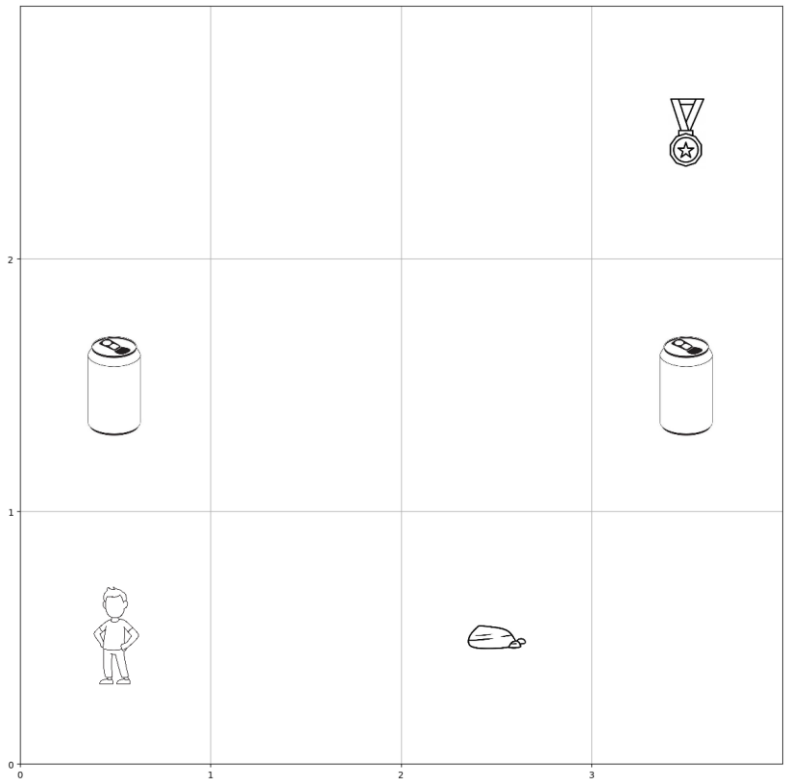


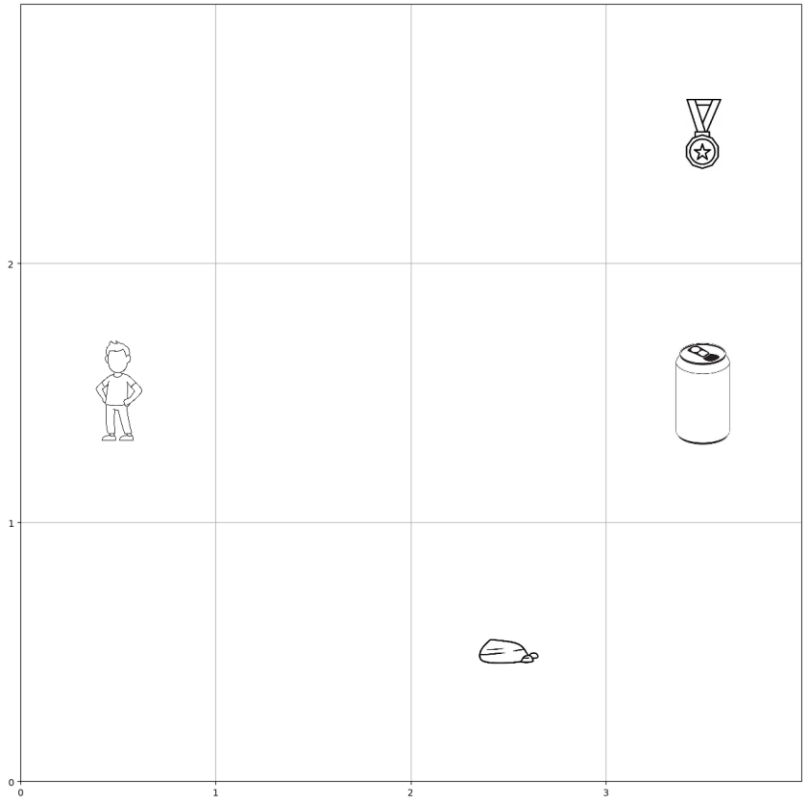

0
1
2
3
0
1
2

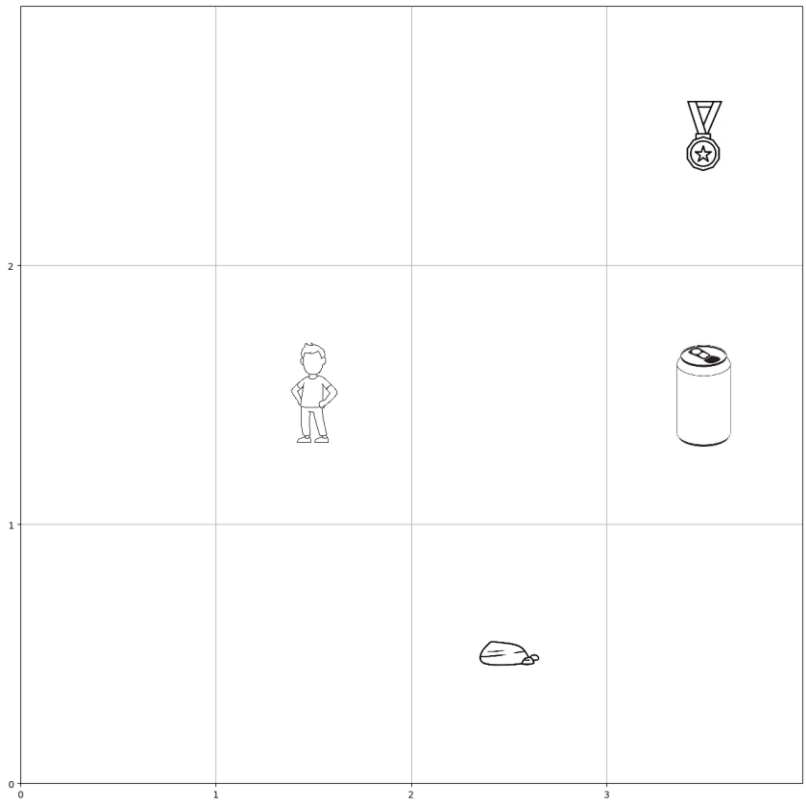

0
1
2
3
0
1
2

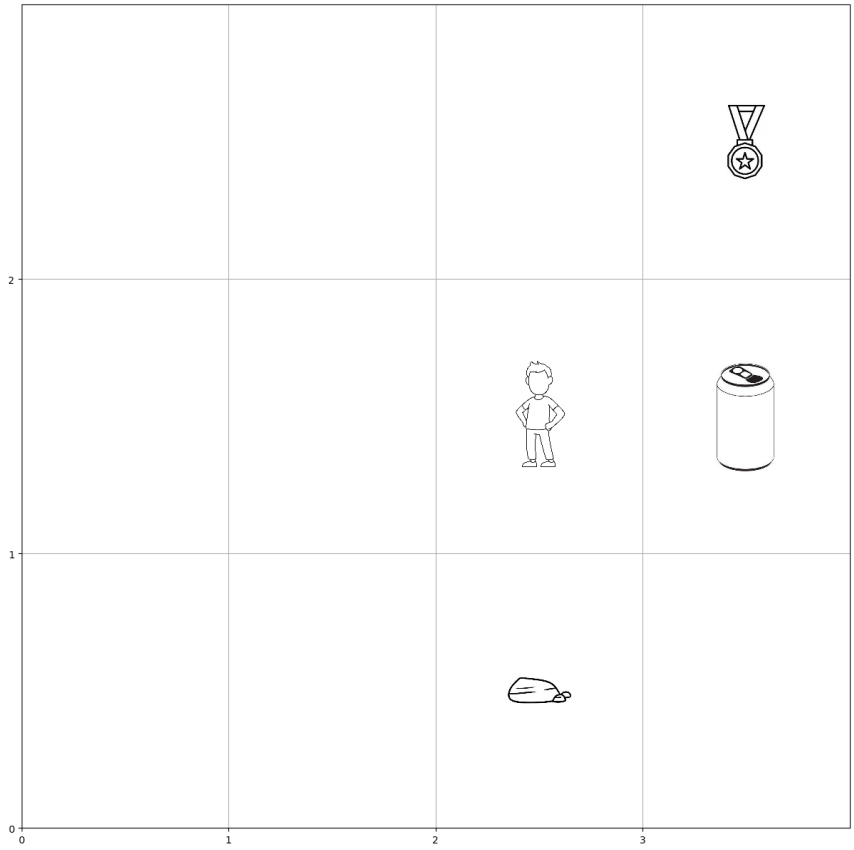

0
1
2
3


0
1
2
3


0
1
2
0
1
2
3


0
1
2
0
1
2
3


0
1
2
3


0
1
2
3

Throughout our experimentation with three distinct epsilon decay values, the lowest being 0.85, the agent achieved a maximum reward of 25 during the learning process. Comparatively, when we tried 0.90 and 0.85, 0.90 yielded better results than 0.85. The slower decay of epsilon at 0.90 allowed for continued exploration, whereas at 0.85, the decay rate inhibited sufficient exploration, impacting the performance.

Upon closer examination of the 0.90 decay case, we noticed that epsilon reached its minimum value prematurely, stalling further exploration and hindering learning. This limitation resulted in a significantly poorer outcome of 15.

Contrarily, with an epsilon decay set at 0.99, we found a favorable balance between epsilon decay and Q-table learning, resulting in the best outcome of 25 rewards compared to other trials.

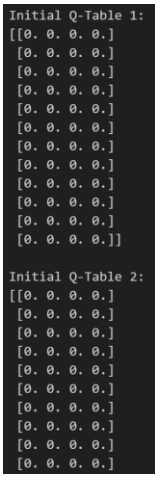
Hyperparameter tuning

One parameter, gamma, was adjusted from 0.95 to 0.85. This change implies that the agent places greater emphasis on immediate rewards. Consequently, the agent becomes more inclined to explore new states and actions, even if they carry inherent risks.

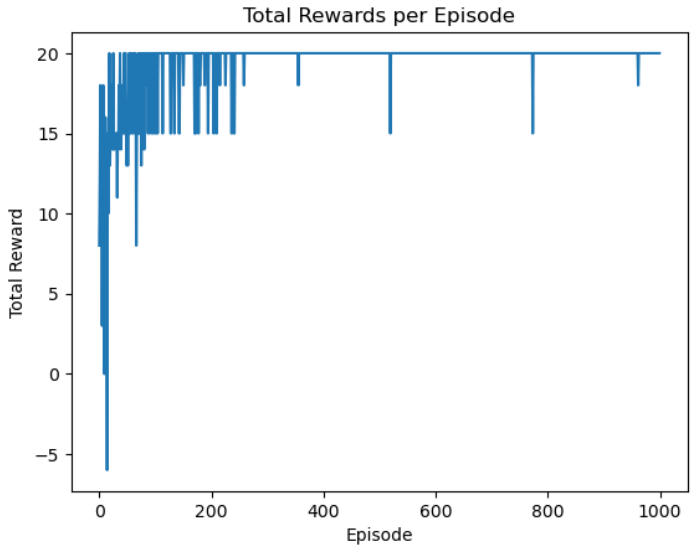
Comparing the plots between gamma values of 0.95 and 0.85 revealed that the total rewards per episode were higher with a gamma value of 0.85. This shift occurred because the agent, now prioritizing immediate rewards, tended to choose actions that yielded immediate benefits, disregarding their long-term impact on total rewards.

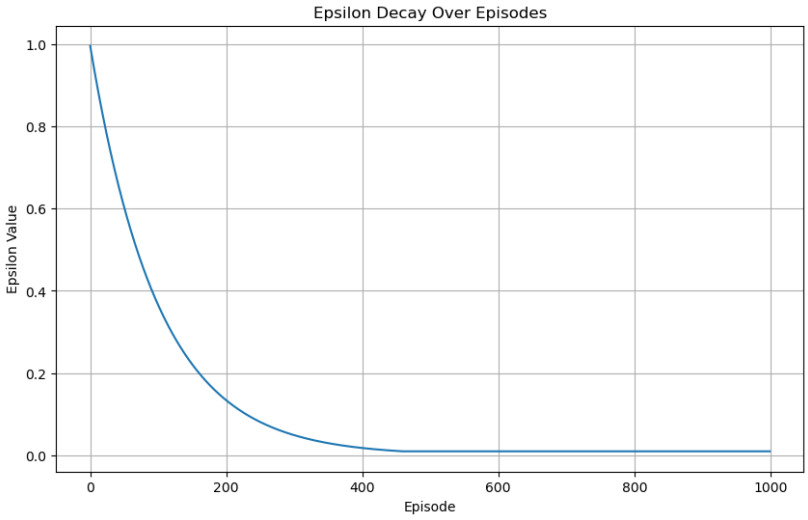
For the tuning, we choose gamma which is the discount factor, and epsilon decay.

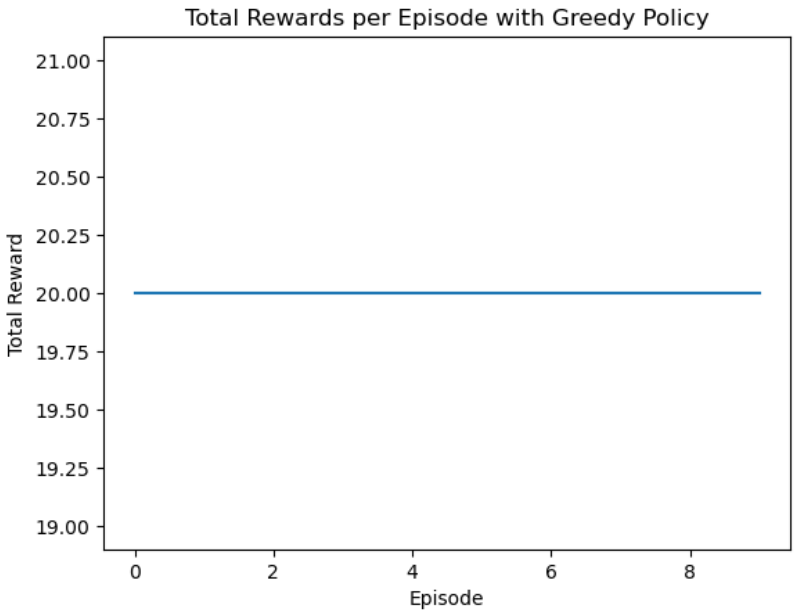
a) Discount Factor gamma = 0.95



```
Trained Q-Table 1:
[[ 4.65549577e+00  9.29735622e+00  1.74319375e+01  9.25754548e+00]
 [ 3.36164882e-03  9.80131604e+00  3.52630076e+00  1.22910903e+00]
 [ 8.50723625e-01  2.38475807e-01  5.83814876e+00 -7.54400177e-01]
 [ 1.36523632e+00 -8.20384958e-01  7.96139485e+00  9.22386202e-01]
 [ 1.30734736e+01  7.02158348e+00  3.31982943e+00  8.63494436e+00]
 [ 1.37750000e+01  6.04461901e+00  3.32070819e+00  1.99613409e+00]
 [ 1.44997178e+01  7.89132584e+00  5.37135013e+00  2.60926886e+00]
 [ 7.22944089e+00  6.47217497e+00  1.00000000e+01  4.24348609e+00]
 [ 1.90757456e+00  1.48152083e+00  1.89798517e+00  7.36968380e+00]
 [ 7.16657443e+00  7.14162195e-01  8.90657379e-01  1.40064865e+00]
 [ 9.61847958e+00  7.14900575e-01  4.60567816e+00  2.60415788e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]

Trained Q-Table 2:
[[ 5.22603423  8.25347401 17.40252915 10.9314384 ]
 [-0.99241239  9.55767342  2.99665291  2.05012911]
 [ 2.12614255  2.21879657  9.28507444 -0.43495533]
 [ 0.86054517 -0.70862317  7.29194414  1.44835989]
 [13.08625     7.5960757   4.1512433   9.27383805]
 [13.77071224  6.65061762  3.53052176  2.5380146 ]
 [14.5         4.04958795  6.46438839  0.87195486]
 [ 6.8434412   7.99290463 10.          4.39861949]
 [ 1.78199151  1.5520387   0.77712521  6.66696207]
 [ 6.00262306  0.42150841  1.46570576  0.5724215 ]
 [ 9.28210201  1.49640521  4.11472701  3.9353601 ]
 [ 0.          0.          0.          0.        ]]
```



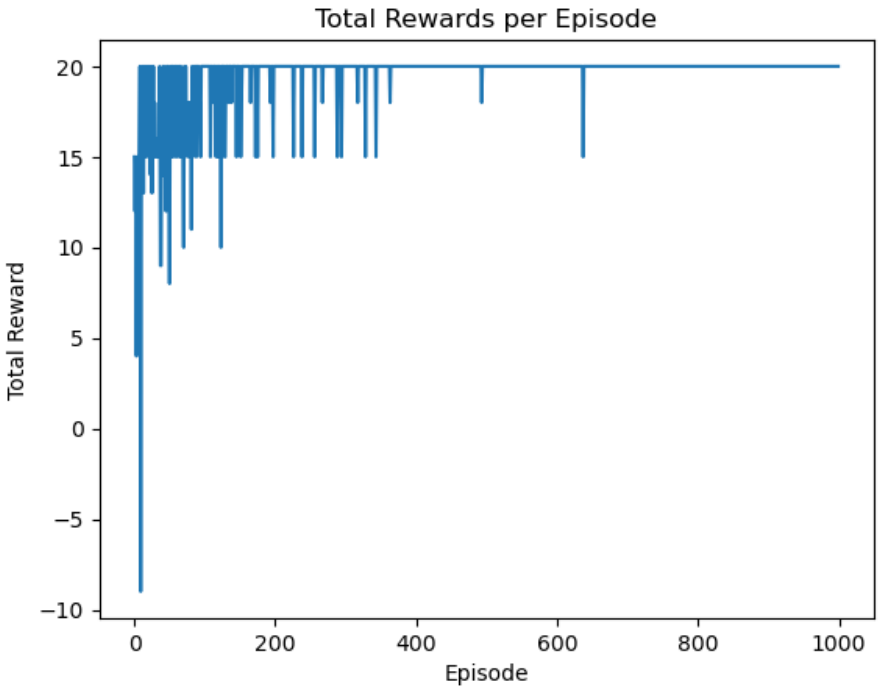


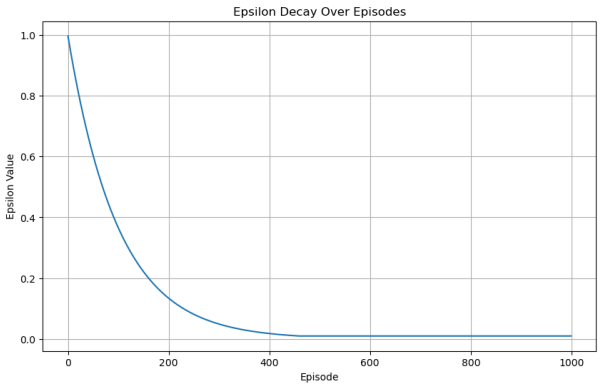


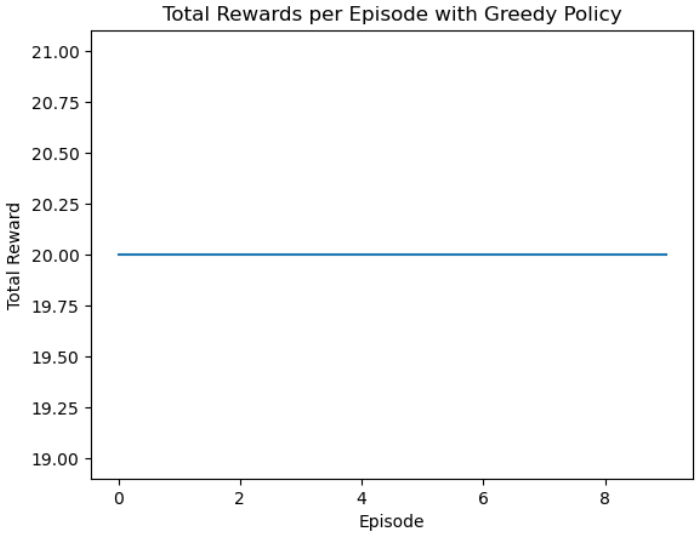
b) Discount Factor gamma = 0.90

```
Initial Q-Table 1:
[[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.]]

Initial Q-Table 2:
[[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.]]
```

```
Trained Q-Table 1:
[[ 3.99576048  7.71823168 15.206       9.21002832]
 [ 0.32192343  9.53909369  2.11570172  1.80744928]
 [ 1.5076884   1.55129539  5.54832538 -0.67967178]
 [ 1.01540562 -0.15655879  5.76196291  1.29907589]
 [11.34        6.10764672  2.78058184  6.38495529]
 [12.6         3.4970123   1.28471536  2.35297095]
 [14.          3.48333841  4.03514633  1.34375194]
 [ 5.44993879  6.87638041 10.          2.61809006]
 [ 1.29286963  0.70694444  0.71691504  5.95017929]
 [ 7.00942139  0.08286036  0.69470475  0.36716028]
 [ 9.41850263  1.05191739  4.08710708  2.24826233]
 [ 0.          0.          0.          0.        ]]

Trained Q-Table 2:
[[ 3.77531422  6.47826084 15.206       7.99296561]
 [-0.61238361  7.11353607  2.73090854  1.55351946]
 [ 1.20516875  0.02008395  9.35221391 -0.76684594]
 [ 0.27187306 -0.41648452  6.82752441  0.19023085]
 [11.34        5.43016541  2.35863421  4.69151004]
 [12.6         4.24750309  4.60377889  2.91317096]
 [14.          3.0735351   5.65771269  0.85267443]
 [ 5.00092328  4.98640114 10.          2.40363482]
 [ 2.73757786  0.36926764  0.33004286  6.3216426 ]
 [ 5.89906585  0.10421266  0.76294654  0.13913321]
 [ 9.52898713  0.75454504  1.62531451  2.15971767]
 [ 0.          0.          0.          0.        ]]
```


Total Rewards per Episode
Total Reward
Episode
20
15
10
5
0
−5
−10
0
200
400
600
800
1000

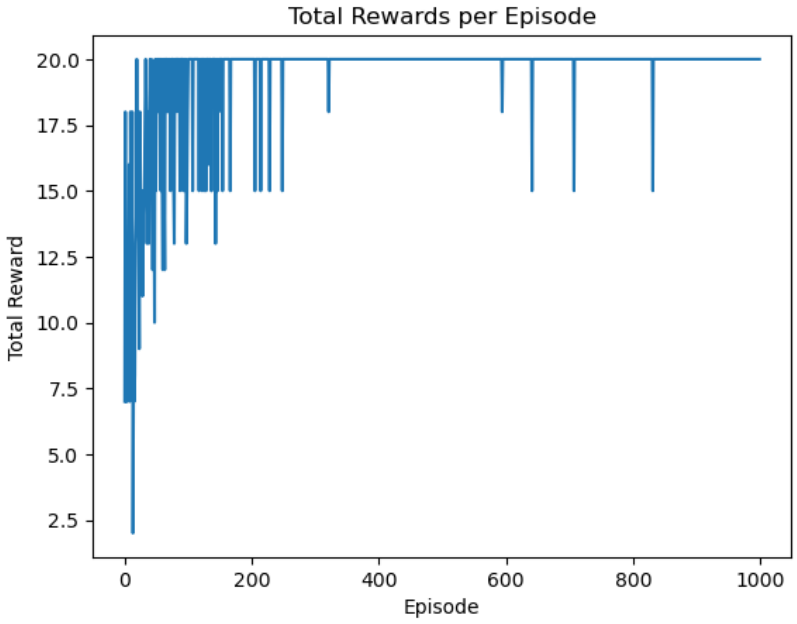

Epsilon Decay Over Episodes
Epsilon Value
Episode
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

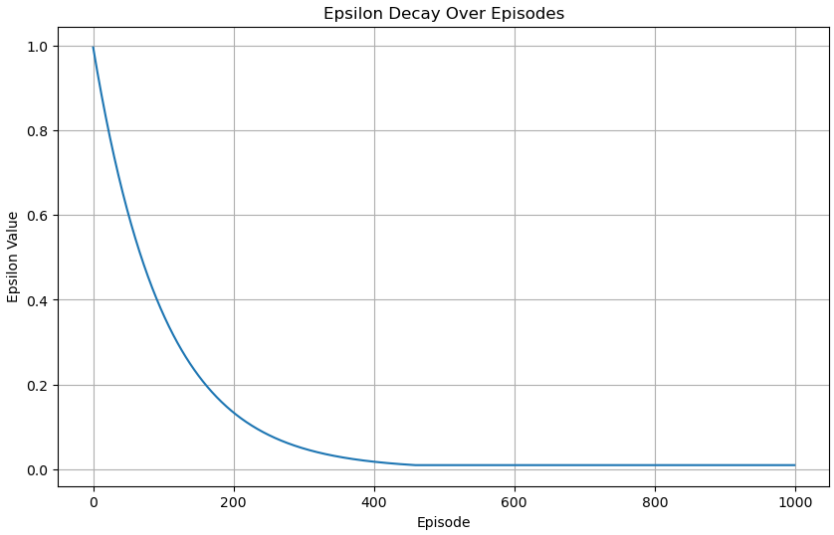

Total Rewards per Episode with Greedy Policy
Total Reward
Episode
21.00
20.75
20.50
20.25
20.00
19.75
19.50
19.25
19.00
0
2
4
6
8

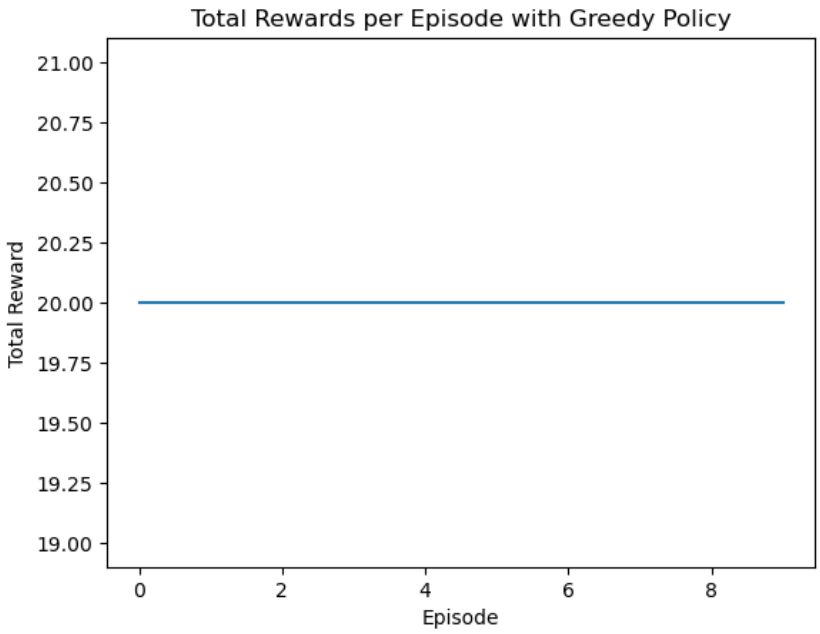
c) Discount Factor gamma 0.85

```
Initial Q-Table 1:
[[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.]]

Initial Q-Table 2:
[[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.]]
```

```
Trained Q-Table 1:
[[ 2.68037735  7.40364847 13.2857584   4.30291211]
 [-0.35881595  6.43373831  1.77276789  0.81067257]
 [ 1.70539693  0.21149277  6.91592152 -0.49568907]
 [ 0.3671541  -0.51967429  7.24412245  0.57672503]
 [ 9.75375     4.90355679  1.99033561  5.71878907]
 [11.47454777  4.21672257  2.26968501  1.97841671]
 [13.5         7.40007895  5.46573609  0.6310514 ]
 [ 6.25348742  6.8602937  10.          3.6577441 ]
 [ 1.10460405  0.91431982  0.33328046  4.7952398 ]
 [ 5.50594606  0.39352259  0.28909948  0.91309601]
 [ 9.35389181  0.96199707  2.52652184  2.38072796]
 [ 0.          0.          0.          0.        ]]

Trained Q-Table 2:
[[ 2.9609961   6.0841242  13.2906875   8.16002751]
 [-0.78873352  8.81666252  1.7391263   0.82200835]
 [ 0.72907211  0.46480428  6.33882777 -0.99137224]
 [ 0.54833891 -0.70711147  7.02968852  0.24724101]
 [ 9.75219249  4.69488199  2.48355954  5.72832649]
 [11.475       4.69892305  2.57485291  2.58045704]
 [13.49992037  5.26683555  4.80837528  1.21350835]
 [ 5.55953398  6.23061291 10.          3.37777532]
 [ 1.23256649  1.17774169  0.40734942  4.92664523]
 [ 5.79400265  0.64219167  0.21001548  0.59632291]
 [ 9.35389181  0.67758442  2.30170264  2.17929047]
 [ 0.          0.          0.          0.        ]]
```


Total Rewards per Episode
Total Reward
Episode
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0
200
400
600
800
1000


Epsilon Decay Over Episodes
Epsilon Value
Episode
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000

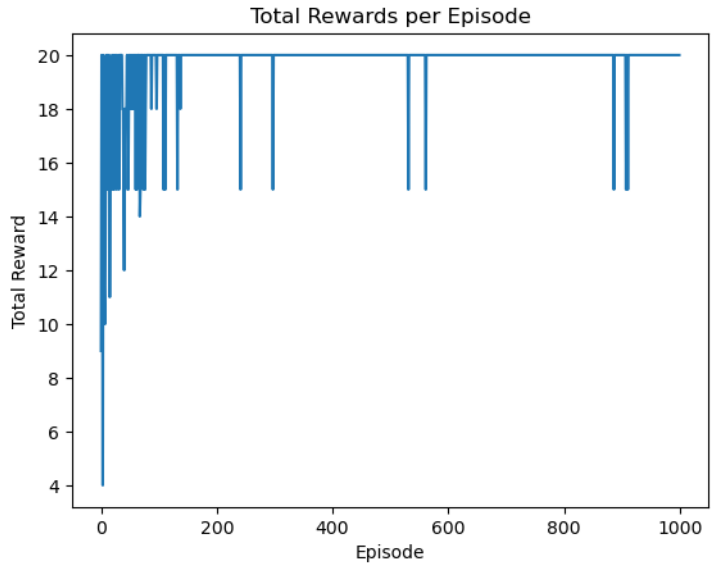


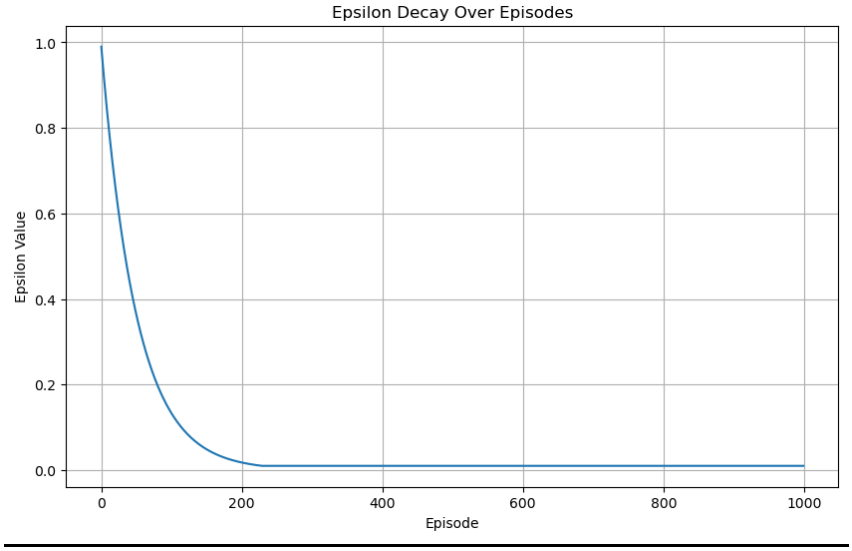
While tuning the discount factor within the range of 0.85 to 0.95, minimal significant changes were observed in the epsilon decay graph. Across all cases, the agent stabilized after approximately 100 episodes. The rewards-per-episode graph illustrates the agent's progressive performance enhancement over time. Specifically, with a discount factor of 0.95, the average episode reward consistently rises from nearly 0 in initial episodes to surpass 100 in about 100 episodes. However, for discount factors of 0.90 and 0.85, similar improvements in reward are visible, albeit accompanied by more pronounced distortions in the graph. Consequently, among the tested values, the discount factor of 0.95 emerges as the most favorable choice.

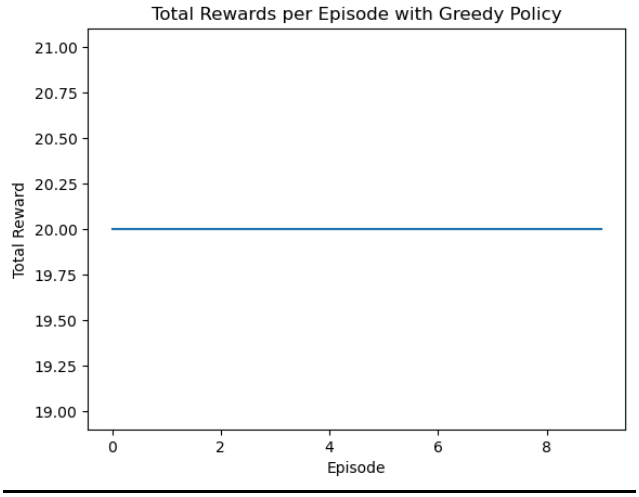
d) Epsilon decay rate 0.99

```
Initial Q-Table 1:
[[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.]]

Initial Q-Table 2:
[[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.]]
```

```
Trained Q-Table 1:
[[ 5.55596860e+00  4.99161246e+00  1.94530094e+01  5.27273976e+00]
 [-1.26174525e+00  8.16466907e+00  1.48156912e+00  7.18934948e-01]
 [ 9.76867198e-01  5.56111505e-01  3.82616848e+00 -4.60191047e-01]
 [ 0.00000000e+00 -7.41314046e-01  5.79632664e+00  3.95452572e-01]
 [ 1.44443088e+01  5.58871715e+00  2.13154115e+00  5.88813769e+00]
 [ 1.47505408e+01  4.69853799e+00  1.28270190e+00  1.47294286e+00]
 [ 1.48073490e+01  8.98560646e-01  3.00353361e+00  1.15542283e+00]
 [ 4.34396979e+00  5.84579998e+00  1.00000000e+01  1.61753198e+00]
 [ 3.02011237e-01  8.88562765e-01  6.87672403e-01  5.53916300e+00]
 [ 3.75392712e+00  5.28303829e-01  2.51565029e-01  2.24823130e-03]
 [ 8.49905365e+00  4.41966835e-01  5.81922000e-01  1.26850398e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]

Trained Q-Table 2:
[[ 3.52265836  6.22868799 19.37363565  5.54305475]
 [-0.75576269 12.70696428  2.36872591  0.41570135]
 [ 0.39558848  0.          7.6524804  -0.59476915]
 [ 0.         -0.53783757  5.11652104  0.21663699]
 [14.60214762  4.38076841  2.60017632  8.13488815]
 [14.56839962  1.16222189  0.95931442  1.52579707]
 [14.89996915  4.94392419  4.3590095  -0.27077581]
 [ 4.13448945  5.45413337 10.          2.08062698]
 [ 0.31636589  0.34167573  0.37589213  3.63712694]
 [ 3.55917448  0.41249614  0.259306    0.5704624 ]
 [ 7.94108868  0.33859456  0.          0.33944099]
 [ 0.          0.          0.          0.        ]]
```



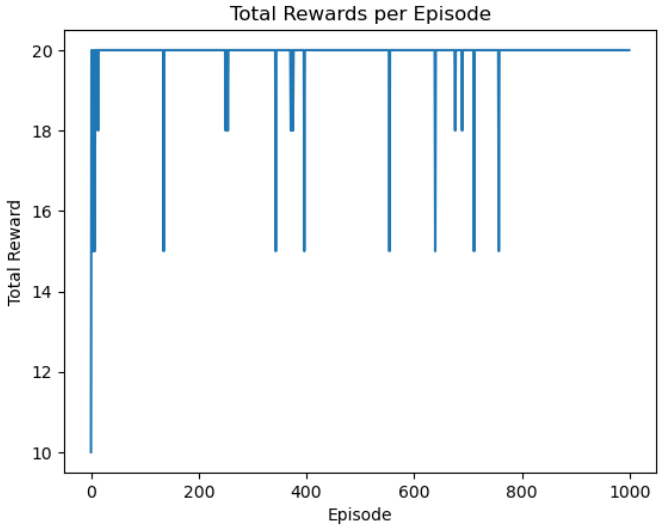


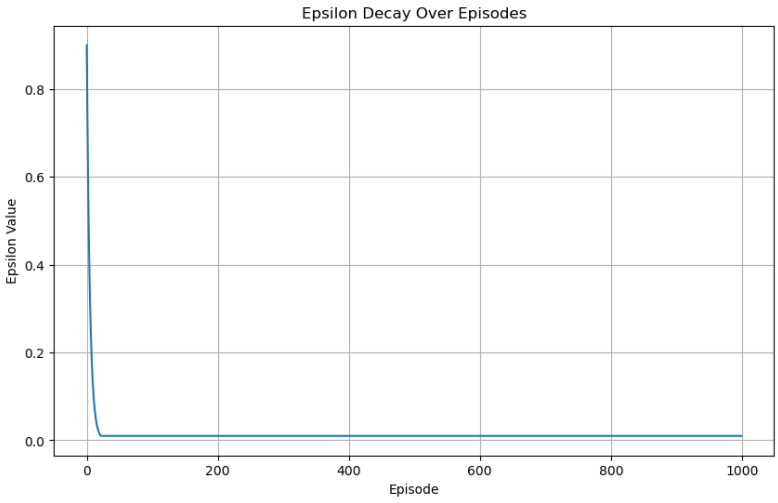


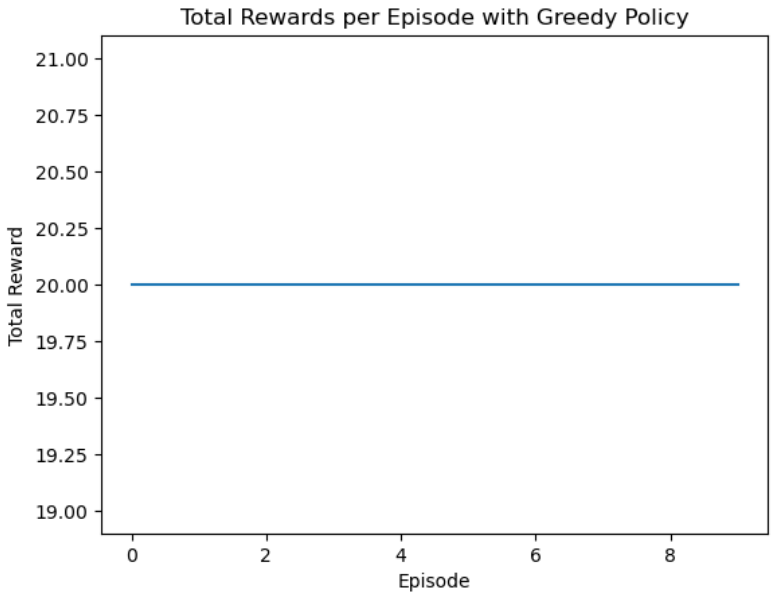
e) Epsilon rate 0.90

```
Initial Q-Table 1:
[[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.]]

Initial Q-Table 2:
[[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.]]
```

```
Trained Q-Table 1:
[[ 0.00000000e+00  2.22099736e+00  1.32906875e+01  2.19387271e+00]
 [-2.00000000e-01  0.00000000e+00  1.49710279e-01  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  3.10943954e+00 -3.80000000e-01]
 [ 0.00000000e+00 -2.00000000e-01  1.61454828e+00  0.00000000e+00]
 [ 9.75374967e+00  3.41635263e+00  9.30055824e-03  1.07172998e-01]
 [ 1.14750000e+01  8.29068709e-01  3.25847500e-02  0.00000000e+00]
 [ 1.35000000e+01  0.00000000e+00  2.92315000e-01  9.59052306e-02]
 [ 8.49999895e-01  1.14741441e+00  1.00000000e+01  0.00000000e+00]
 [ 0.00000000e+00  9.30055824e-03  0.00000000e+00  9.07566918e-01]
 [ 4.99630000e-01  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 4.09510000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]

Trained Q-Table 2:
[[ 0.          0.06163985 13.29068618  1.21138195]
 [-0.2         0.          0.975375    0.        ]
 [ 0.         -0.017       3.37219163  0.        ]
 [ 0.          0.          0.          0.        ]
 [ 9.75375     0.85965834  0.03027005  1.2231972 ]
 [11.47499998  0.82808537  0.         -0.0153    ]
 [13.5         0.          0.56848    -0.55472904]
 [ 0.85        1.181925   10.          0.        ]
 [ 0.          0.          0.          0.9349221 ]
 [ 0.238       0.          0.          0.        ]
 [ 4.0951      0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]]
```



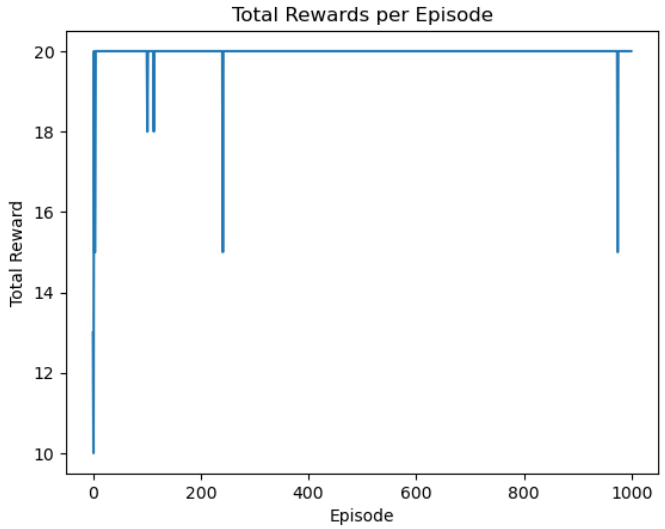


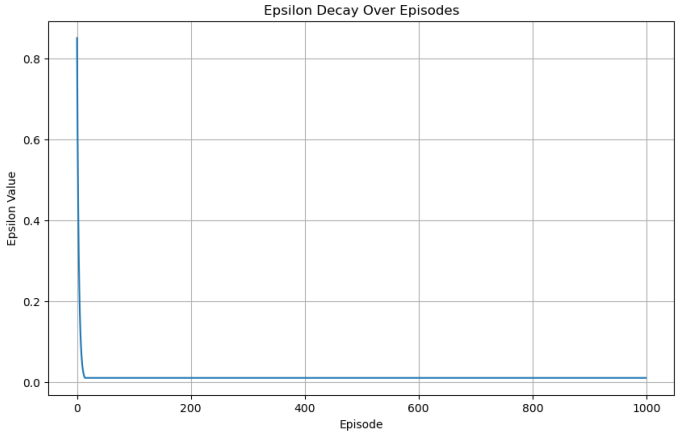


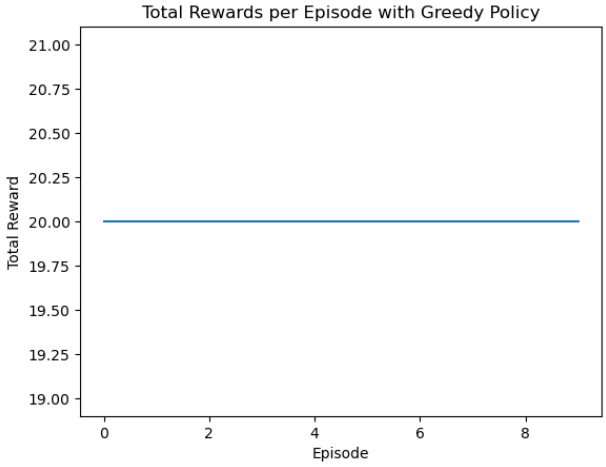
f) Epsilon Decay rate 0.85

```
Initial Q-Table 1:
[[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.]]

Initial Q-Table 2:
[[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.]]
```

```
Trained Q-Table 1:
[[ 0.19571568  2.21710449 13.24144895  1.17690588]
 [ 0.          2.3848277   0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.         -0.38        3.28641276  0.        ]
 [ 9.75374323  0.84071188  0.          2.03623815]
 [11.475       0.91996035  0.          0.09135837]
 [13.4999998   1.85319137  0.1615     -0.2       ]
 [ 0.          1.1475     10.          0.4133842 ]
 [ 0.          0.          0.          0.82906875]
 [ 0.          0.          0.          0.        ]
 [ 2.71        0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]]

Trained Q-Table 2:
[[ 1.29117428e-01  1.16542782e+00  1.32906542e+01  1.19758991e+00]
 [-2.00000000e-01  3.08482005e+00  1.57254643e-02  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00 -2.00000000e-01]
 [ 0.00000000e+00 -2.00000000e-01  1.79459172e+00  0.00000000e+00]
 [ 9.75375000e+00  3.83346940e-03  0.00000000e+00  1.67013455e+00]
 [ 1.14749991e+01  1.21289482e-01  0.00000000e+00  1.18577527e-01]
 [ 1.35000000e+01  0.00000000e+00  0.00000000e+00 -3.80000000e-01]
 [ 0.00000000e+00  1.14749600e+00  1.00000000e+01  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.04611623e-01]
 [ 8.50000000e-02  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 1.00000000e+00  0.00000000e+00  0.00000000e+00  4.25000000e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```







For epsilon decay set to 0.99, the agent demonstrates a balanced approach between exploration and exploitation. The decay graph illustrates that it takes approximately 100 episodes for the agent to stabilize. In contrast, with epsilon decay at 0.95, stabilization requires nearly 200 episodes. However, when epsilon decay is reduced to 0.85, the graph indicates a prolonged exploration phase without reaching stabilization. Over time, the rewards-per-episode graph showcases the agent's performance improvement. Under epsilon decay of 0.99, the average

episode reward steadily climbs from nearly 0 in early episodes to surpass 100 in approximately 100 episodes. In comparison, with epsilon decay at 0.95, achieving a similar milestone takes around 200 episodes. Notably, for epsilon decay set to 0.85, considerable irregularities are observed, signifying ongoing learning. Thus, among the tested values, epsilon decay at 0.99 emerges as the more favorable choice.

Best Hyperparameters for Double Q Learning

Based on our extensive experiments with the Double Q-Learning model, we identified the most optimal parameter set as follows: gamma=0.95 and epsilon decay=0.99, which produced the highest total reward of 25. Our best parameters hence are :
Epsilon=1, Episode count=100, Learning rate=0.1, Discount factor=0.95, and Epsilon decay=0.99.

Final Observations

Upon reviewing the final outcomes of both learning methods, it's evident that Double Q-Learning achieves a higher average reward per episode compared to SARSA.

In Q-Learning, the maximum reward per episode reaches approximately 230, while SARSA achieves around 200. Analysis of the plots indicates that Double Q-Learning enables the agent to learn more rapidly, reaching the maximum reward per episode quicker than SARSA.
Double Q-Learning demonstrates greater stability, devoid of significant spikes in reward per episode, unlike SARSA. SARSA, in contrast, exhibits a slower learning pace compared to Double Q-Learning, taking more time to achieve its maximum reward per episode.
SARSA's stability is lower, marked by more frequent spikes in reward per episode compared to the more consistent performance of Double Q-Learning.

In conclusion, based on the presented plots and observations, Double Q-Learning emerges as the superior algorithm when compared to SARSA.

Team Member	Assignment Part	Contribution Percentage
NEEMA GEORGE	Part 1, Part 2, Part 3, Bonus	50%
SRI SAHITH REDDY KUNCHARAM	Part 1, Part 2, Part 3, Bonus	50%

REFERENCES

- **https://ubuffalo-my.sharepoint.com/personal/avereshc_buffalo_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Favereshc%5Fbuffalo%5Fedu%2FDocuments%2F2023%5FFall%5FML%2F%5Fpublic%2FCourse%20Materials%2FRL%20Environment%20Visualization%20by%20Nitin%20Kulkarni&ga=1**
- **https://optuna.org/**
- **https://blog.floydhub.com/an-introduction-to-q-learning-reinforcement-learning/**