Assignment – 3

Contribution Table

Name	Parts	Contribution
Ruthvik Vasantha Kumar	**1 & 2 & 3 & bonus**	**50%**
Pavan Kumar Kulkarni	**1 & 2 & 3 & bonus**	**50%**

Part – 1:

1. Description of the Environment:

States: The environment is a grid world with size 5×5 and the state space consist of all grid positions (i,j), where i,j∈[0,4].

Actions:

- 0: Move up.
- 1: Move right.
- 2: Move down.
- 3: Move left.

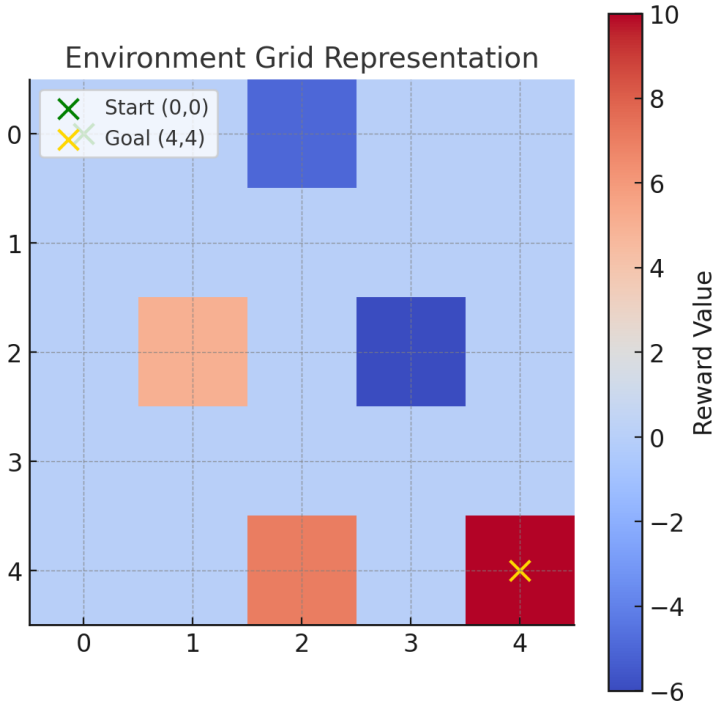
Rewards: Specific rewards are associated with certain grid cells:

- Negative rewards (e.g., −5 at (0,2) and −6 at (2,3)).
- Positive rewards (e.g., +5 at (2,1), +7 at (4,2), and +10+10+10 at the goal state (4,4)).
- Default reward: 0 for other cells.

Objective: The agent starts at the top-left corner (0,0) and aims to reach the goal state (4,4) while maximizing cumulative rewards.

Termination: The episode ends when the agent reaches the goal state.

2. Visualization of the Environment:



The visualization shows the 5×5 grid environment:

- The green marker represents the starting position (0,0).
- The gold marker represents the goal position (4,4).
- The grid cells are color-coded to represent reward values:
    - Positive rewards are shown in warm colors (e.g., red, orange).
    - Negative rewards are in cool colors (e.g., blue).

3. Safety in AI:

- Allowed Actions: The agent's actions are restricted to valid grid moves making sure that no transitions outside the defined grid space take place.
- State Validation: Before updating the state, checks ensure that actions do not move the agent beyond the grid boundaries.
- Reward and Termination Checks: The system monitors and applies rewards based only on predefined states and the episode terminates safely upon reaching the goal.
- Visualization and Debugging: The render() function provides a visual inspection to ensure the agent's trajectory and the environment's integrity.
- Random Sampling Safeguards: Even with random actions, constraints on state transitions prevent invalid updates, maintaining stability.

Part - 2

Explain SARSA method used to solve the problem, including the update function and key features. Discuss its advantages and disadvantages.

SARSA (State-Action-Reward-State-Action) is an on-policy reinforcement learning algorithm used for solving Markov Decision Processes (MDPs). It is appropriate for safe, policy-driven learning since it learns the action-value function Q(s,a) under the same policy that is used for action selection.

Key Concepts

On-Policy Learning

SARSA is an on-policy reinforcement learning algorithm, it updates

the action-value function Q(s, a) based on the action a′ chosen by the current

policy (e.g., ϵ-greedy). This makes it suitable for learning under the same policy

used for acting.

Update Rule

The SARSA update rule is:

$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$

where:

• α: Learning rate, controls the step size for updates.

• γ: Discount factor, determines the weight of future rewards compared to

immediate rewards.Steps

1. Initialize Q(s,a).
2. Choose a in s using the policy.
3. Take a, observe r and s′.
4. Select a′ in s′ using the same policy.
5. Update Q(s,a) using the update rule.
6. Repeat until termination.

Advantages

- Safe updates as it learns under the policy which is executed.
- Handling the stochastic environments effectively.
- Converges to the best course of action after investigation.

Disadvantages

- Slower convergence compared to off-policy methods like Q-learning.
- Dependent on the quality of the exploration strategy.
- May favor suboptimal actions under poor exploration settings.

Comparison with Q-Learning

Feature	SARSA	Q-Learning
Nature	On-policy	Off-policy
Update Rule	Uses Q(s′,a′)	Uses max Q(s′,a)
Safety	Safer for risky environments	Riskier in stochastic settings
Convergence	Slower	Faster

SARSA is suitable for safe learning, such as in robotics, while Q-learning is better for faster learning when safety is of less importance.

Provide a short description on how hyperparameters influence performance. Suggest the most efficient hyperparameter values for your problem setup

Impact of Hyperparameters on Performance

Hyperparameters such as Gamma and Epsilon Decay play an important role in the performance of reinforcement learning algorithms:

- Gamma determines the weight given to future rewards. Higher values give priority to long-term rewards and lower values prioritize immediate ones.
- Epsilon Decay controls the transition of model from exploration to exploitation. Slower decay encourages in depth exploration and faster decay might lead to inadequate exploration.

The plot shows how performance, measured by average reward, changes with different combinations of hyperparameters. Finding the right balance is important for achieving both stability and efficiency in learning

Recommended Hyperparameter Values

Based on the results:

- The optimal hyperparameter values for this setup are:
    - Gamma = 0.9
    - Epsilon Decay = 0.98

These values acheived the highest average reward of 584.82, indicating the best performance.

Provide the description, plots and evaluation results for the setup the returns the best results (Step 4). Provide your detailed analysis.

The setup demonstrates the implementation of **Q-learning**, showing the agent's learning process from an initial state to an optimal policy.

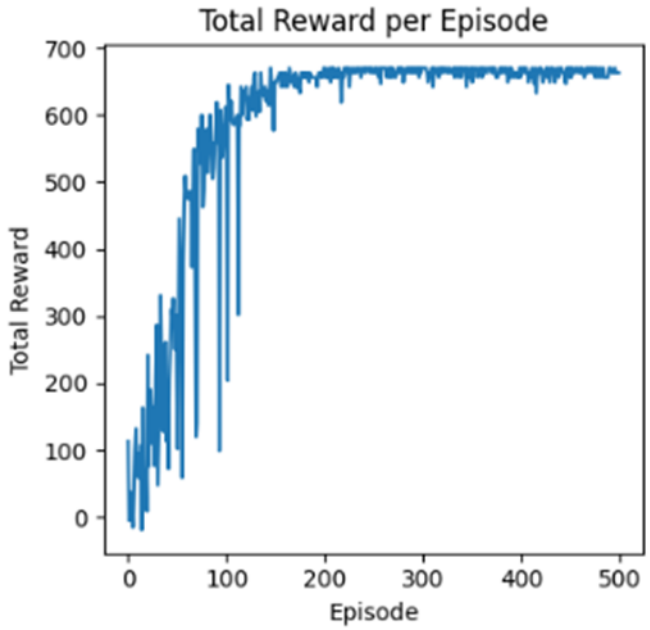
Key Aspects:

1. **Initial Q-Table:** The Q-table starts with all values set to zero, indicating that the agent has no prior knowledge of the environment defined. This table is updated iteratively as the agent learns from interactions and rewards through the process.
2. **Trained Q-Table:** After training, the Q-table shows the optimal state-action values, showing the agent's learned policy to maximize cumulative rewards.
3. **Epsilon Decay:** The decay plot shows how gradually reducing the epsilon value will allow the agent to explore more at the beginning and focus on making better decisions as training goes on.
4. **Performance Metrics:**
    a. **Total Reward per Episode:** Rewards steadily increase over time and saturate around 700, showing that the agent has settled into an optimal strategy.
    b. **Greedy Policy Evaluation:** Testing with a greedy approach (no exploration) resulted in consistently high rewards, proving the effectiveness of the policy.

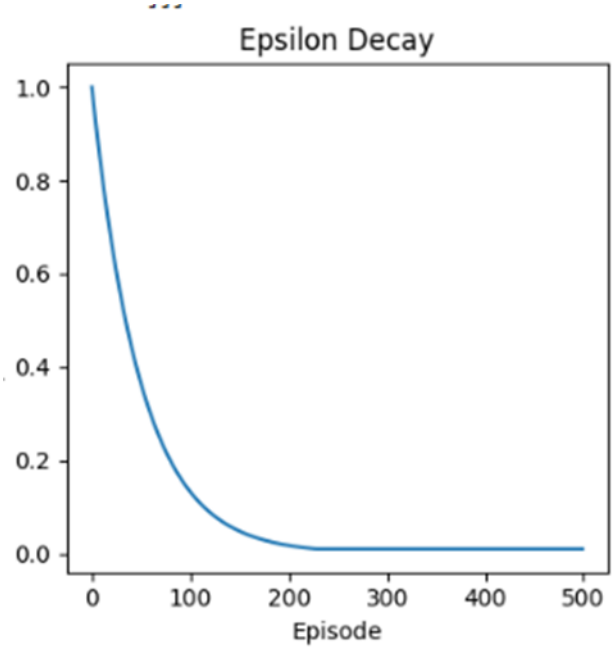
Evaluation Results:

```
Trained Q-table:
[[[ 1.82245181e+01  2.70779536e+00  4.43758941e+01  3.34658211e+00]
  [ 2.09289801e+00 -4.64219196e+00  1.04329581e+01  2.76444805e-01]
  [-1.74604143e+00 -3.34532834e-01 -2.40804436e-01 -3.78792652e-01]
  [-5.41239389e-01 -5.47161600e-02 -3.05012213e-01 -3.58051711e+00]
  [-6.59260225e-02 -3.42125690e-02 -1.88350191e-02 -5.07267196e-01]]

 [[ 1.14457373e+01  1.40848253e+01  4.95392091e+01  4.21500431e+00]
  [ 4.28379782e-01  4.03851691e+00  3.20811299e+01  3.24250389e+00]
  [-3.99411502e+00 -2.23633618e-01  2.06989580e+00  1.15338270e+01]
  [-1.74660584e-01 -2.11764563e-02 -3.87021001e+00  1.37530530e+00]
  [-4.92153212e-02 -3.18880584e-02  9.50829632e-02 -3.99583047e-01]]

 [[ 1.19285619e+01  5.48741735e+01  8.74479471e+00  1.52017446e+01]
  [ 9.92635539e+00  1.25570005e+01  5.49467325e+01  2.06670169e+01]
  [ 3.99779921e+00 -4.95954692e+00  1.08761863e+01  4.03926831e+01]
  [-2.12147699e-01 -3.31485531e-01  3.05836547e+00  3.06802220e+00]
  [-2.00556098e-02  4.58453454e-02  1.09697965e+00 -2.75691743e+00]]

 [[ 2.79001138e+01  5.10849285e+00  4.07042739e+00  2.63958967e+00]
  [ 2.34258764e+01  6.12996406e+01  3.27450662e+01  1.00104979e+01]
  [ 2.13585873e+01  3.06896377e+01  6.93533214e+01  2.62038293e+01]
  [-2.68800287e+00  1.02598836e+00  5.08332567e+00  4.58065634e+01]
  [ 1.13779495e-01  1.00076902e+00  6.86189404e+00  1.69363986e+00]]

 [[ 2.67826926e+00  4.25073064e+01  5.79545435e+00  2.40956856e+00]
  [ 2.39138003e+01  6.88534455e+01  3.26651655e+01  2.30971095e+01]
  [ 6.24437942e+01  6.19402806e+01  6.92996759e+01  6.13958267e+01]
  [ 1.95605075e+01  9.11370619e+00  3.31460257e+01  6.88095071e+01]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]
```

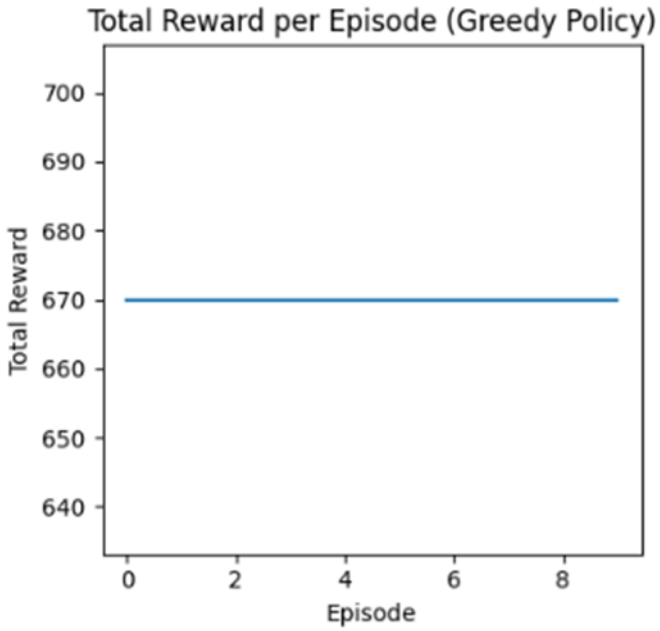
The final Q-values indicated the agent's understanding of the environment, with higher values representing actions that acheive better cumulative rewards.



The Total Reward per Episode plot shows steady improvement, with rewards leveling off around 700 after about 250 episodes, indicating that the policy has settled.



The epsilon value gradually drops from 1 to almost 0 running over 500 episodes, allowing for exploration early on and focused decision-making later for stable learning.



During the **Greedy Policy Evaluation**, the agent consistently achieves rewards close to 700, validating the stability and optimality of the learned policy.

Why Is It a Straight Line?

- **Consistent Policy:** After training, the agent's Q-table has settled, with state-action values showing the best decisions. Because the policy would always pick the action with the highest Q-value, the agent's behavior is consistent and predictable.
- **Optimal Rewards:** In this environment, greedy policy will earn the same high reward every episode. With no randomness or exploration, the total reward stays consistent across episodes.

Key Insights from the Graph:

1. **Policy Stability:** The straight line indicates that the policy learnt by the agent is stable. It consistently achieves the same reward under test, showing that the agent has successfully converged to an optimal solution.
2. **No Exploration:** During evaluation, with epsilon set to 0, there's no randomness, so the rewards are the same in each episode
3. **Maximum Achievable Reward:** The flat line around 680–700 rewards shows that the agent's policy is nearly optimal for this setup.

Part –3

1. **Explain Double Q-learning method used to solve the problem, including the update function and key features. Discuss its advantages and disadvantages.**

Double Q-Learning Method

Double Q-learning is an improvement over standard Q-learning, designed to address the issue of overestimating Q-values. In regular Q-learning, when updating the Q-values, the algorithm relies on the maximum estimated value of the next state. This can lead to overly optimistic predictions, especially in environments with noisy or unpredictable rewards.

Core Idea:

To reduce this overestimation bias, Double Q-learning keeps two separate Q-value tables, Q1 and Q2. These tables are updated independently, with one table being used to select an action and the other to evaluate it. This separation helps make the value estimates more accurate.

Update Function

In Double Q-learning, one of the Q-tables is randomly selected for updating at each step. Here's how it works:

1. **Action Selection:**
    a. Use one table (e.g., Q1) to choose the best action a*:
    $a^* = \text{argmax}_a Q1(s', a)$
    b. Here, s' is the next state after taking action a in state s.
2. **Action Evaluation:**
    a. Evaluate the value of this action $a*$ using the other table (e.g., Q2):
    $Q2(s', a^*)$
    This represents the value of action $a*$ in state s'.
3. **Update Rule:**
    a. If *Q1* is selected for updating, the update rule is:
    $Q1(s,a) \leftarrow Q1(s,a) + \alpha[r + \gamma Q2(s', a^*) - Q1(s,a)]$
    b. If *Q2* is selected for updating, the update rule is:
    $Q2(s,a) \leftarrow Q2(s,a) + \alpha[r + \gamma Q1(s', a^*) - Q2(s,a)]$
    c. Here:
        i. α is the learning rate.
        ii. γ is the discount factor.
        iii. r is the reward for taking action a in state s.

By alternating updates between Q1 and Q2, the algorithm reduces the risk of overestimating action values.

Key Features

1. **Two Q-Tables**:
    a. Maintains two independent Q-value tables (Q1 and Q2).
2. **Separate Roles**:
    a. One table is used to select the action, and the other is used to estimate the value of that action.
3. **Reduces Overestimation**:
    a. The separation of action selection and evaluation leads to more realistic value estimates.

Advantages

1. **Reduces Bias**:
    a. Helps to counteract the tendency of Q-learning to overestimate values, resulting in more accurate predictions.
2. **Improved Learning Stability**:
    a. By decoupling action selection from evaluation, the algorithm produces smoother and more reliable learning.

3. **Better Performance**:
    a. Particularly effective in environments with noisy or unpredictable rewards.
4. **Compatibility**:
    a. Can be extended to work with deep learning methods, as seen in Double DQN (Deep Q-Networks).

Disadvantages

1. **Extra Computational Cost**:
    a. Maintaining and updating two Q-tables increases memory usage and computational load compared to standard Q-learning.
2. **Slower Updates**:
    a. Only one table is updated per step, so learning can take longer.
3. **Complexity**:
    a. The alternating update process adds implementation complexity.

2. Provide your derivation for the n-step Double Q-Learning update rule (Step 1).

Steps Breakdown:

Theoretical Task: Derive the n-step Double Q-Learning

Update Rule: Double Q-learning maintains two Q-tables (Q1 and Q2) and randomly selects one for action evaluation and the other for value estimation. For the n-step return, we consider the cumulative discounted rewards over n steps.

1 Update Rule

For a state-action pair (st, at), let:

R(n)

t = rt+1 + γrt+2 + · · · + γn−1rt+n (n − step cumulative reward).

G(n)

t = R(n)

t + γn max

a Q2(st+n, a),

if using Q1 for the update (and vice versa if using Q2).

The update for Double Q-learning becomes:

Q1(st, at) ← Q1(st, at) + α

G(n)

t − Q1(st, at)

Randomly switch roles between Q1 and Q2 at each update

3. **Provide a short description on how hyperparameters influence performance. Suggest the most efficient hyperparameter values for your problem setup (Step 3)**

How Hyperparameters Affect Performance

Hyperparameters play a crucial role in determining how effectively a reinforcement learning agent learns and performs. Here's how the key hyperparameters in the provided code influence the agent:

1. **n (Number of Steps in Return Calculation)**:
    a. This controls how far into the future the agent considers rewards when updating its Q-values. Smaller values of n (e.g., n=1) focus on short-term rewards, making the learning process faster but potentially overlooking long-term strategies. Larger values (e.g., n=4 or 5) account for long-term rewards but may require more time to converge and can lead to instability if the environment is noisy.
2. **gamma (Discount Factor)**:
    a. The discount factor determines how much weight the agent gives to future rewards. A higher gamma value (close to 1) prioritizes long-term rewards, which is useful for tasks requiring long-term planning. Lower values (closer to 0) favor immediate rewards, which may result in shortsighted policies.
3. **epsilon_decay and epsilon_min (Exploration vs. Exploitation):**
    a. These parameters guide the balance between exploring new actions and exploiting known ones. A slower epsilon_decay (e.g., 0.995) allows the agent to explore longer, which is important in complex environments. Setting epsilon_min to a small value like 0.01 ensures that the agent continues exploring at least minimally, even in later stages, helping avoid local optima.
4. **num_episodes and max_timesteps**:
    a. The total number of episodes (num_episodes) and steps per episode (max_timesteps) influence how much time the agent spends training. More episodes help the agent refine its policy, while sufficient timesteps ensure meaningful interaction with the environment during each episode.

Recommended Hyperparameter Values (Step 3)

Based on the code and problem setup, the following values are likely to be effective:

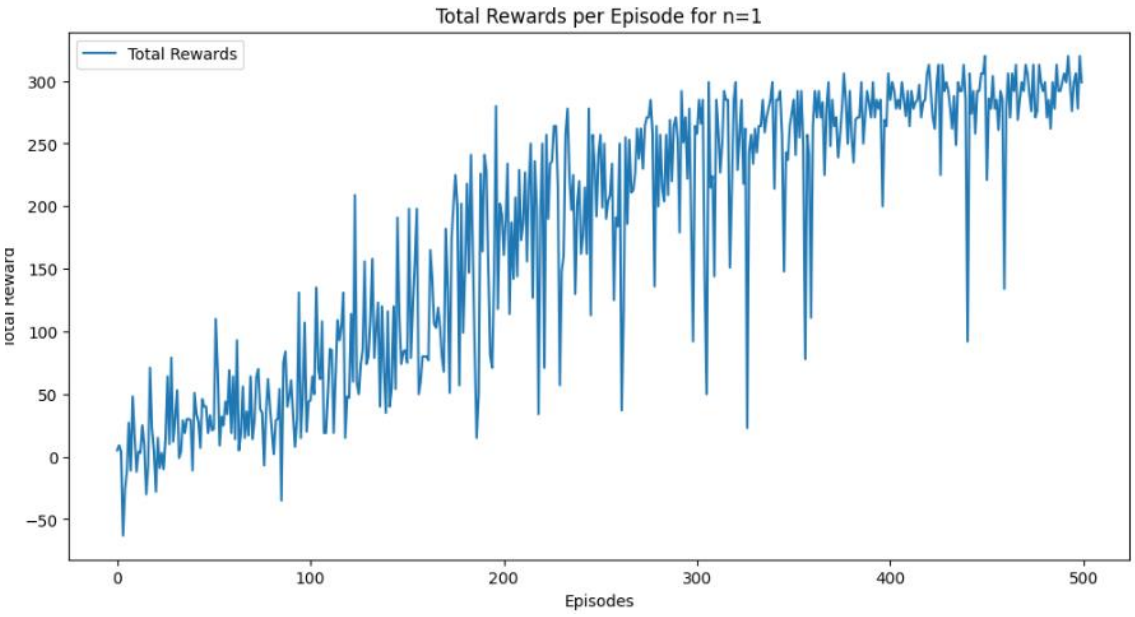
- **n**: A moderate value like n=3 strikes a balance between immediate and long-term rewards, offering stable and efficient learning.
- **gamma**: A value of 0.9 works well, as it considers future rewards without completely neglecting immediate ones.
- **epsilon_decay**: A decay rate of 0.995 ensures a gradual shift from exploration to exploitation, which is helpful in learning complex environments.
- **epsilon_min**: A value of 0.01 allows for occasional exploration, preventing the agent from getting stuck in suboptimal policies.
- **num_episodes**: Setting this to 500 provides sufficient training for the agent to learn a robust policy.
- **max_timesteps**: A value of 50 ensures meaningful interaction per episode without unnecessarily prolonging training.

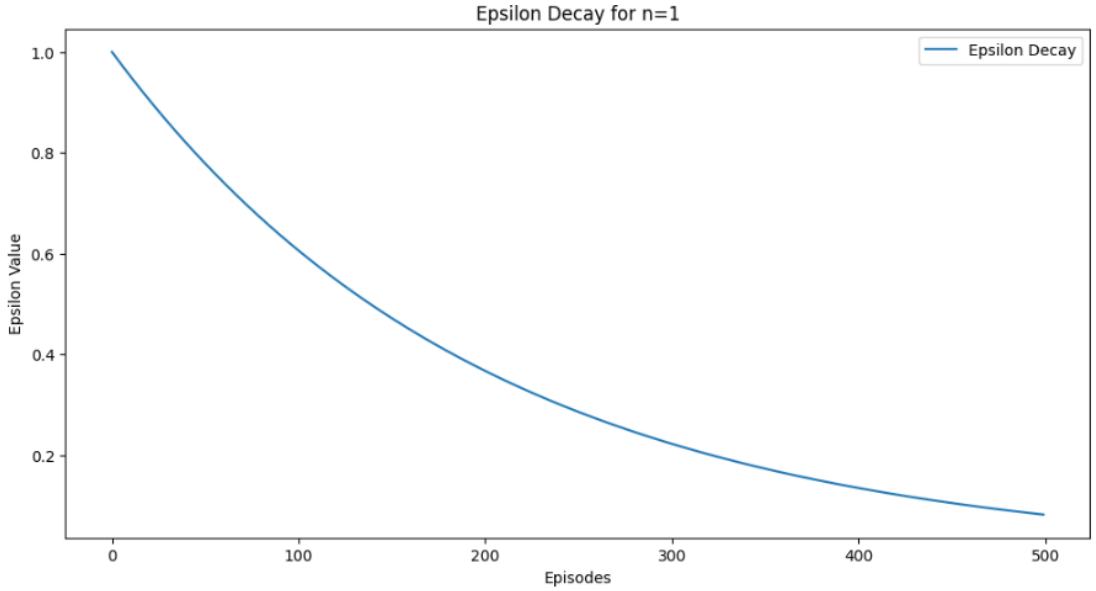
These values provide a good balance between efficient learning and reliable performance, tailored to environments of moderate complexity.

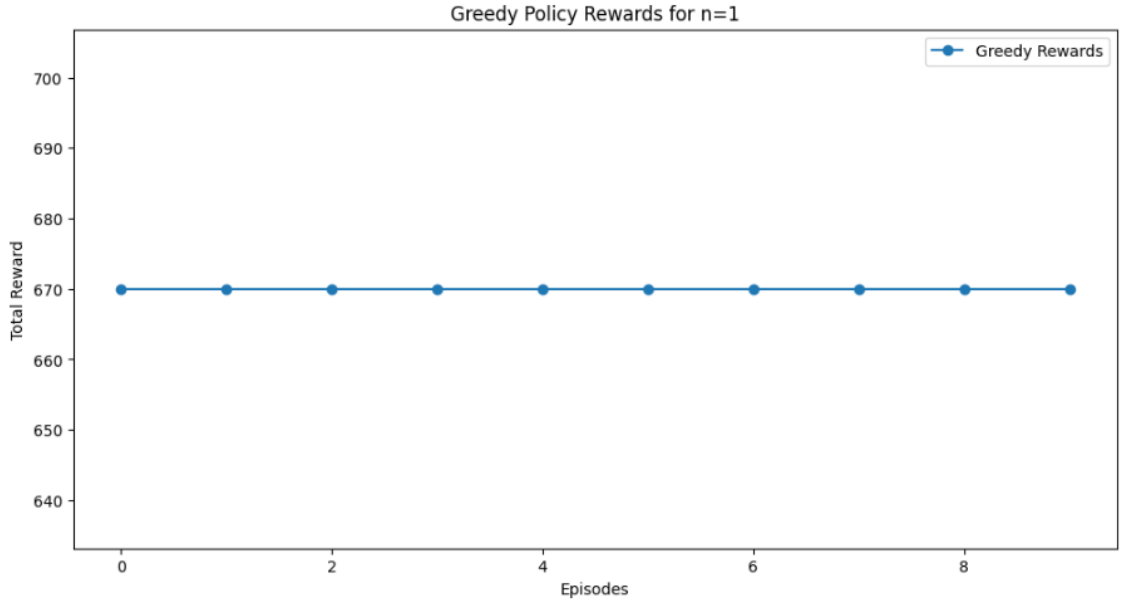
4. Provide the results with different values of n (1, 2, 3, 4, 5). Determine the optimal n-step return for your environment (Step 4).

Results:

For n=1:








```
n=1, Average Greedy Reward: 670.0
```

n=1, Average Greedy Reward: 670.0
Initial Q-table for n=1 (Q1):
[[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]

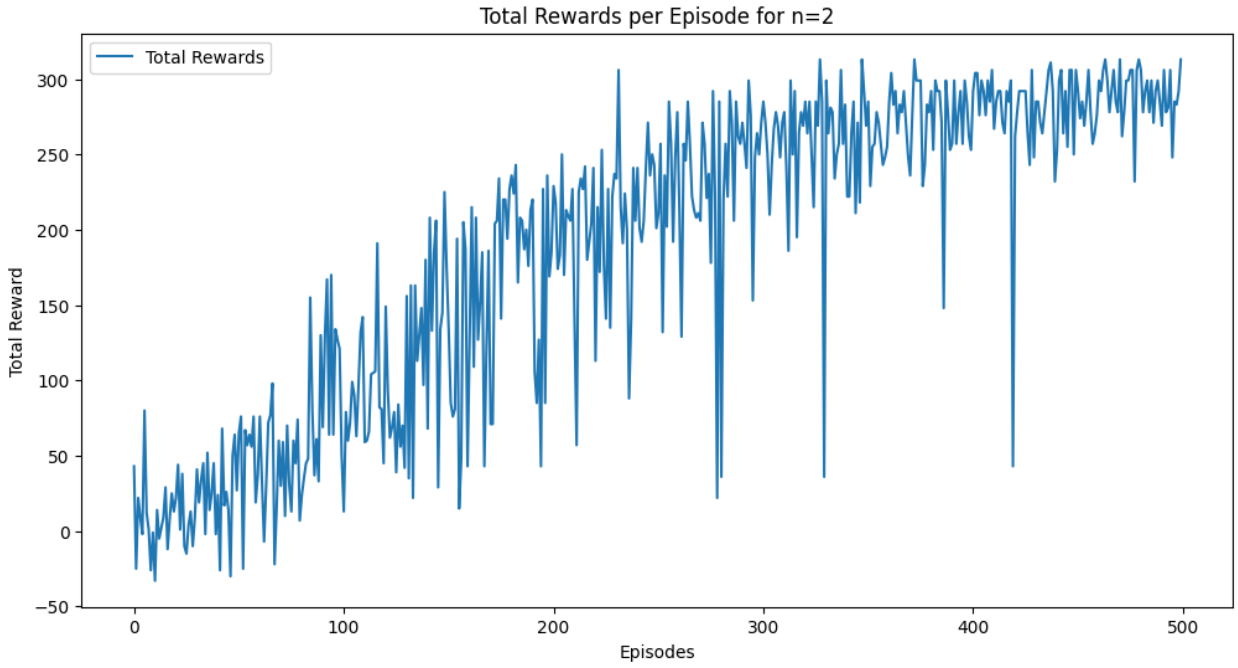
[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]

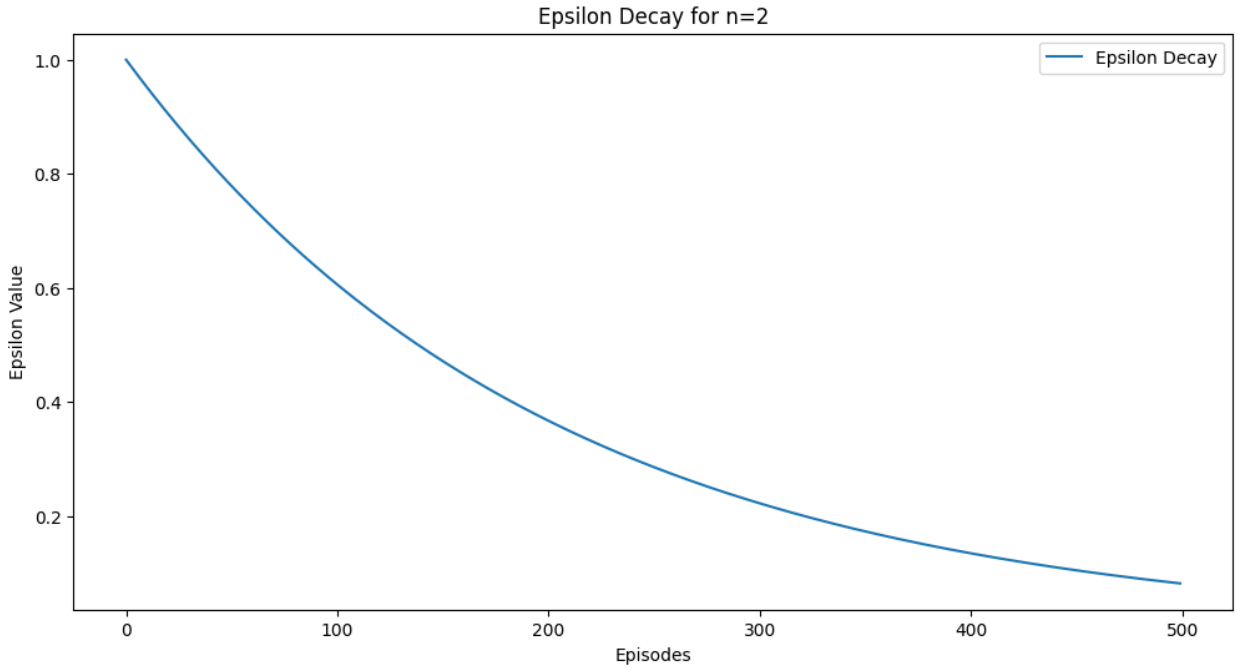
```
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [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 for n=1 (Q2):
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
```

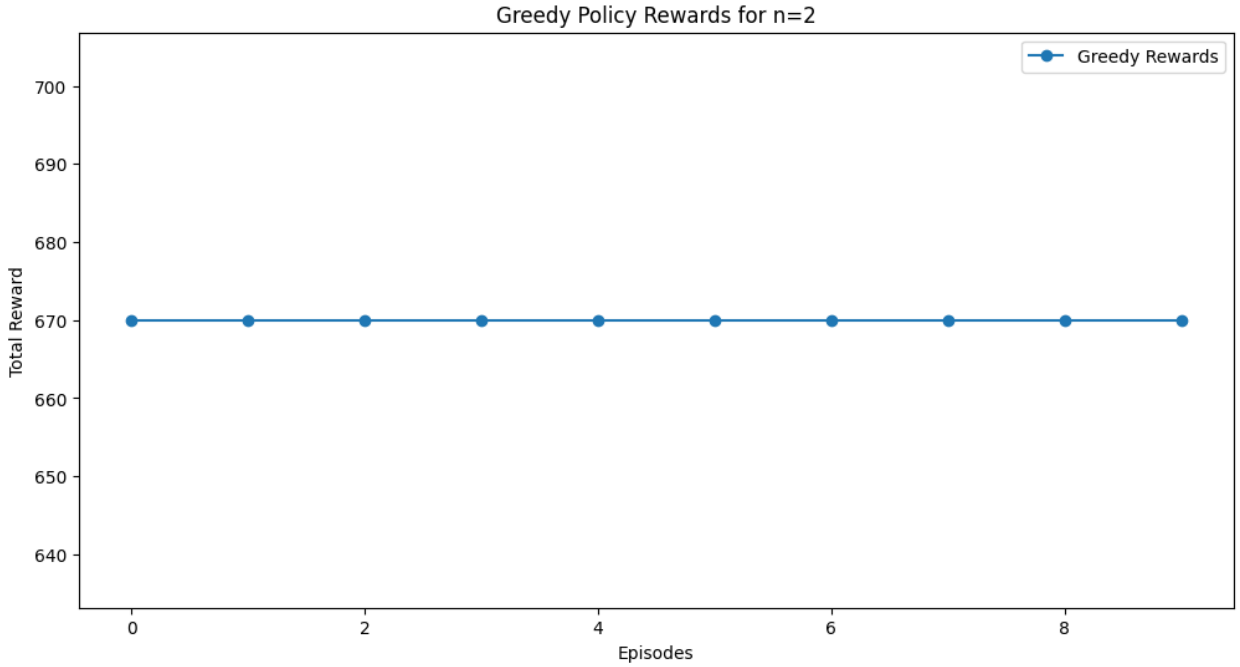
```
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]]
Trained Q-table for n=1 (Q1):
[[[ 2.86942796e+01  2.01672589e+01  4.53827102e+01  1.97427464e+01]
  [ 1.61133870e+01  1.92240501e+00  4.15040648e+01  9.85564935e+00]
  [-2.68506196e+00  6.52562043e-01  6.46422579e+00  1.60639643e+01]
  [ 4.13845977e-02  4.31252463e-02  2.35685501e+00 -1.65160066e+00]
  [ 3.24803078e-03  8.11670664e-04  6.37816364e-02  4.64420474e-01]]

 [[ 2.63736201e+01  5.04268187e+01  3.15232201e+01  3.13350961e+01]
  [ 2.70988106e+01  2.98691246e+01  5.60299971e+01  3.60337677e+01]
  [ 1.04307582e+00  6.24235157e+00  2.07914199e+01  4.71748815e+01]
  [ 7.00128759e-01  4.22804551e-01 -2.45939171e+00  1.43239303e+01]
  [ 4.55708050e-03  1.25115066e-01  1.21707708e+00  5.77084600e-03]]

 [[ 1.07178530e+01  5.03360920e+01  6.69493211e+00  1.70446050e+01]
  [ 4.55494694e+01  5.66999997e+01  3.71056591e+01  2.97134261e+01]
  [ 2.98533545e+01  2.61697252e+01  6.30000000e+01  4.01730127e+01]
  [ 2.52352239e+00  8.87479342e-01  7.66978260e+00  4.57631507e+01]
  [ 1.99506212e-01  1.34204338e+00  4.94611047e+00 -1.55121282e+00]]

 [[ 9.92308650e+00  2.17956575e+01  5.96330261e+00  2.96221331e+00]
  [ 5.54167269e+01  1.92528093e+01  3.29853091e+01  1.05765924e+01]
  [ 5.34137728e+01  5.06705371e+01  7.00000000e+01  4.13850789e+01]
  [ 5.22272413e+00  3.96253846e+00  3.12766824e+01  6.11233015e+01]
  [ 8.69349653e-01  1.23770897e+00  8.64914828e+00  3.60586585e-01]]

 [[ 4.85642831e+00  5.33262698e+01  1.37374245e+00  5.57710217e+00]
  [ 4.03590246e+01  7.00000000e+01  5.52088533e+01  4.07207394e+01]
  [ 6.30000000e+01  6.29999999e+01  7.00000000e+01  6.29999999e+01]
  [ 3.76356713e+01  9.77471600e+00  5.05044239e+01  7.00000000e+01]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]
Trained Q-table for n=1 (Q2):
[[[ 2.49392522e+01  2.59348703e+01  4.53836051e+01  2.36946184e+01]
  [ 1.20270139e+01  2.84968474e+00  4.56758890e+01  1.72443348e+01]
  [ 3.74029110e-02  2.06640867e-01  3.35468147e+00  1.75281100e+01]
  [ 3.09324612e-01  1.69946651e-02  3.54091244e+00 -2.82988860e+00]
  [ 1.15280270e-03  1.19929742e-03  1.97559000e-03  1.32451475e-02]]

 [[ 2.35118804e+01  5.04268560e+01  2.78662905e+01  3.43249259e+01]
  [ 2.68662066e+01  2.55108213e+01  5.60299684e+01  3.31559610e+01]
  [ 3.39067320e+00  7.04999575e+00  1.93060123e+01  4.25930219e+01]
  [ 3.08981317e-01  2.99212953e-01  2.98805086e-03  1.28698858e+01]
  [ 3.12710030e-02  1.76791271e-02  1.81978355e+00  7.15162088e-01]]
```

```
[[ 1.60110113e+01  5.15097168e+01  7.87590804e+00  1.20293811e+01]
 [ 3.41575074e+01  5.66999997e+01  3.55828334e+01  3.22544994e+01]
 [ 2.06267555e+01  1.32755473e+01  6.30000000e+01  4.31797233e+01]
 [ 4.34254877e+00  1.58668698e+00  6.47549134e+00  4.94251693e+01]
 [ 1.40596225e-01  2.37852310e-01  5.04901107e+00  5.30267408e-01]]

[[ 8.38021523e+00  2.41512962e+01  1.74860027e+00  4.04632546e+00]
 [ 5.54374737e+01  3.21193734e+01  3.83068937e+01  1.03731633e+01]
 [ 5.28213717e+01  4.74562872e+01  7.00000000e+01  4.52062707e+01]
 [ 1.04769718e+01  3.24877503e+00  1.43964553e+01  6.24876879e+01]
 [ 1.54472052e+00  1.74433668e+00  8.49905365e+00  6.63877585e-01]]

[[ 5.09199124e+00  5.69845034e+01  3.36485403e+00  1.30373548e+01]
 [ 3.24669663e+01  7.00000000e+01  5.79530499e+01  3.16468564e+01]
 [ 6.30000000e+01  6.29999999e+01  7.00000000e+01  6.30000000e+01]
 [ 4.62457154e+01  9.47665237e+00  4.89494982e+01  7.00000000e+01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]
```

For n =2







n=2, Average Greedy Reward: 670.0
Initial Q-table for n=2 (Q1):
[[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]

[[0. 0. 0. 0.]

```
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [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 for n=2 (Q2):
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [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 for n=2 (Q1):
[[[ 2.37555208e+01  2.23749375e+01  4.16763412e+01  2.61373119e+01]
  [ 7.78011509e+00 -7.63506997e-01  2.85170024e+01  3.97576998e+00]
  [-2.98898070e+00 -8.93607234e-01  1.45597332e+01  4.10459853e+00]
  [ 2.94649760e-02  3.31609256e-02  1.30737849e-01 -2.40515708e+00]
  [ 7.77248906e-02  9.88205305e-02  1.99799508e-01 -4.52270793e-01]]

 [[ 1.99690000e+01  3.06907649e+01  4.71351077e+01  2.63444277e+01]
  [ 1.49822482e+01  1.91577164e+01  5.01663155e+01  1.85712174e+01]
  [-1.07262708e+00  9.99550953e-01  6.43524585e+00  1.88852691e+01]
  [-2.69904625e-01  1.50014136e-01 -8.88179831e-01  4.24400867e+00]
  [ 1.19050642e-01  3.44231746e-01  6.49291096e-01  1.39211929e-01]]

 [[ 2.57798450e+01  5.26404854e+01  2.90163836e+01  3.39824238e+01]
  [ 3.70547820e+01  2.84423606e+01  5.26482135e+01  3.97768154e+01]
  [ 1.25857336e+01  2.69666015e+00  2.97049888e+01  4.73067627e+01]
  [ 9.20923479e-01 -2.28444445e-01  3.45711215e+00  1.76176374e+01]
  [ 9.83610916e-02  2.39486326e-02  4.11223948e+00 -3.46179737e+00]]

 [[ 3.86967014e+01  2.91386891e+01  2.05195188e+01  1.39137352e+01]
  [ 4.39931582e+01  4.32184431e+01  5.87629277e+01  2.83111416e+01]
  [ 3.58669150e+01  4.33414014e+01  6.75275080e+01  4.95738483e+01]
  [ 5.89581689e+00  8.10233942e+00  2.03213613e+01  5.14045349e+01]
  [ 7.12263402e-01  2.73913402e+00  0.00000000e+00  8.47334894e+00]]

 [[ 1.60534734e+01  5.45604101e+01  1.58733153e+01  1.43550425e+01]
  [ 4.92568413e+01  6.76591154e+01  5.46672998e+01  4.89138061e+01]
  [ 5.59374810e+01  5.89754963e+01  6.77889493e+01  6.05313567e+01]
  [ 3.45747210e+01  0.00000000e+00  4.85556776e+01  6.70815200e+01]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]
Trained Q-table for n=2 (Q2):
[[[ 1.85637389e+01  1.85163622e+01  4.17541752e+01  2.39897156e+01]
  [ 1.08270303e+01 -5.87892535e-01  3.06826835e+01  5.88247287e+00]
  [-2.63966111e+00 -1.29556022e+00  8.17171924e+00  2.30231316e+00]
  [-1.32644124e-01  1.14556875e-01 -3.79403047e-01 -2.86167029e+00]
  [ 6.96144468e-02  7.14194563e-02  2.17895619e-01 -1.92428229e-01]]

 [[ 1.99975503e+01  3.16067830e+01  4.62990162e+01  2.85194522e+01]
  [ 1.14663230e+01  7.61319492e+00  4.85392011e+01  1.76912523e+01]
  [-3.87607202e-01  9.77358336e-01  5.32369958e+00  3.28296499e+01]
```

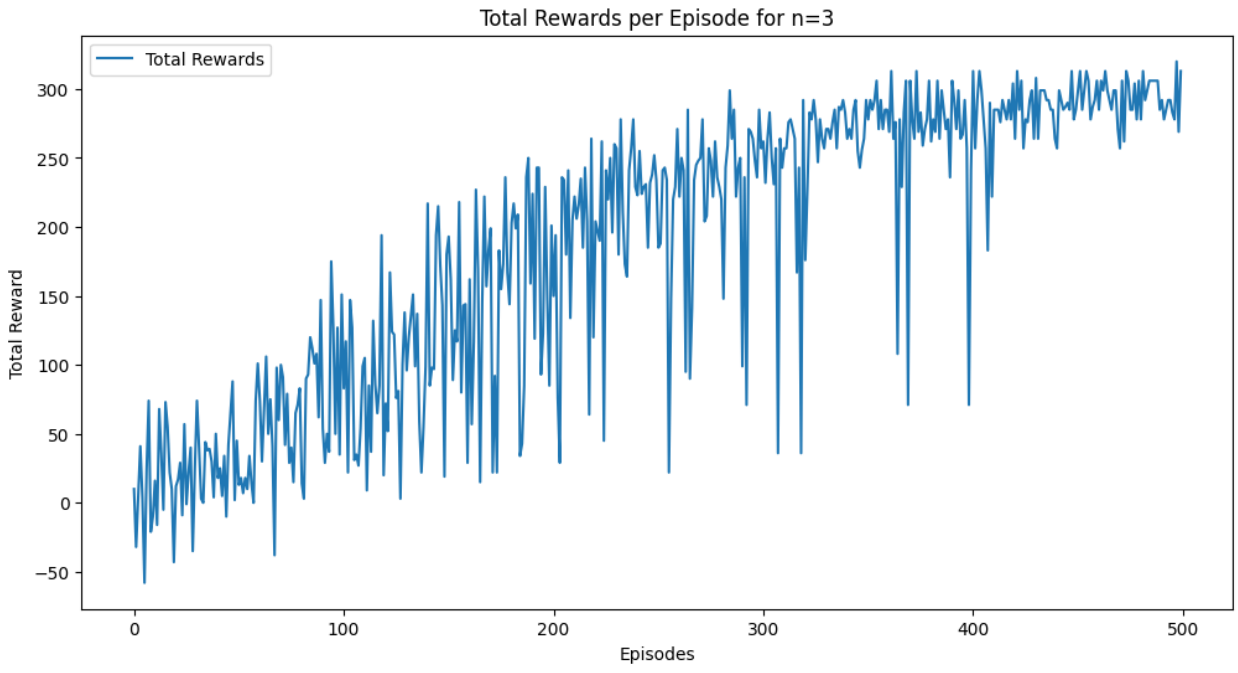
[-3.37793835e-01 2.46221431e-01 -8.55798327e-01 3.34226064e+00]
 [7.74035367e-02 3.62602443e-01 -4.83081948e-01 -1.08272192e+00]]

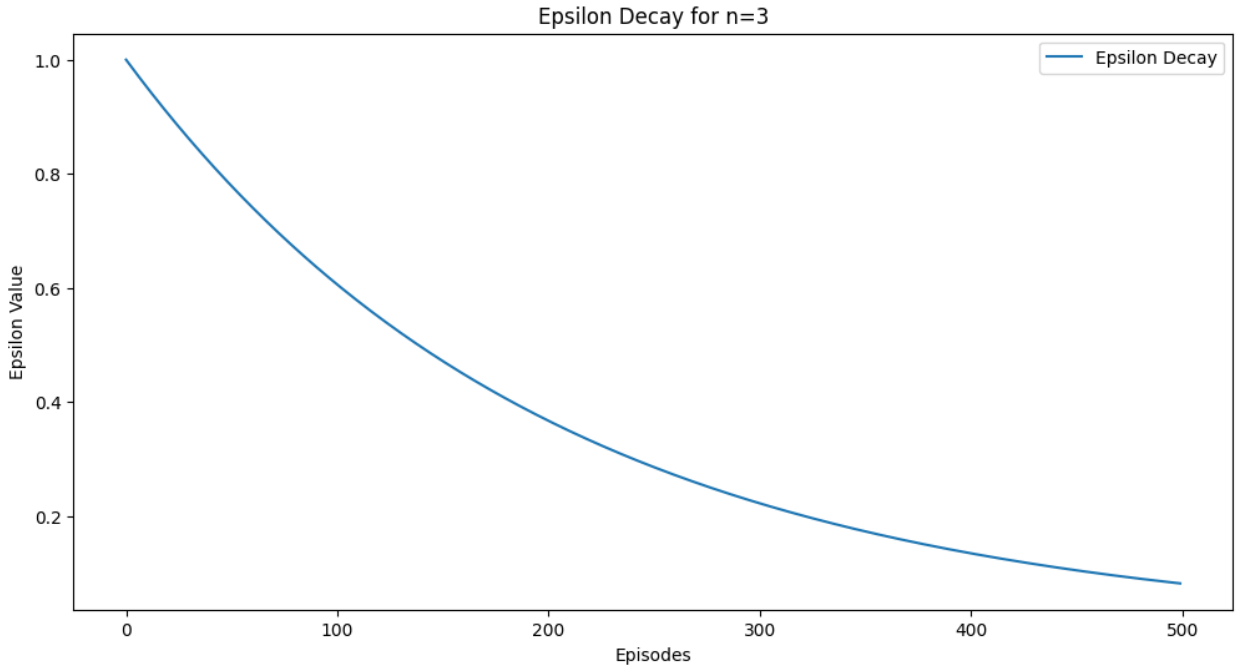
[[3.25757683e+01 5.09926271e+01 3.25138411e+01 3.82308595e+01]
 [3.24581408e+01 3.85722202e+01 5.42158523e+01 3.73312619e+01]
 [9.69004670e+00 6.13375454e+00 2.20708787e+01 4.50410335e+01]
 [1.70762672e+00 -9.93094935e-01 6.48811671e+00 1.97990692e+01]
 [1.28627468e-02 -9.78163729e-01 4.53943316e+00 7.44366727e-02]]

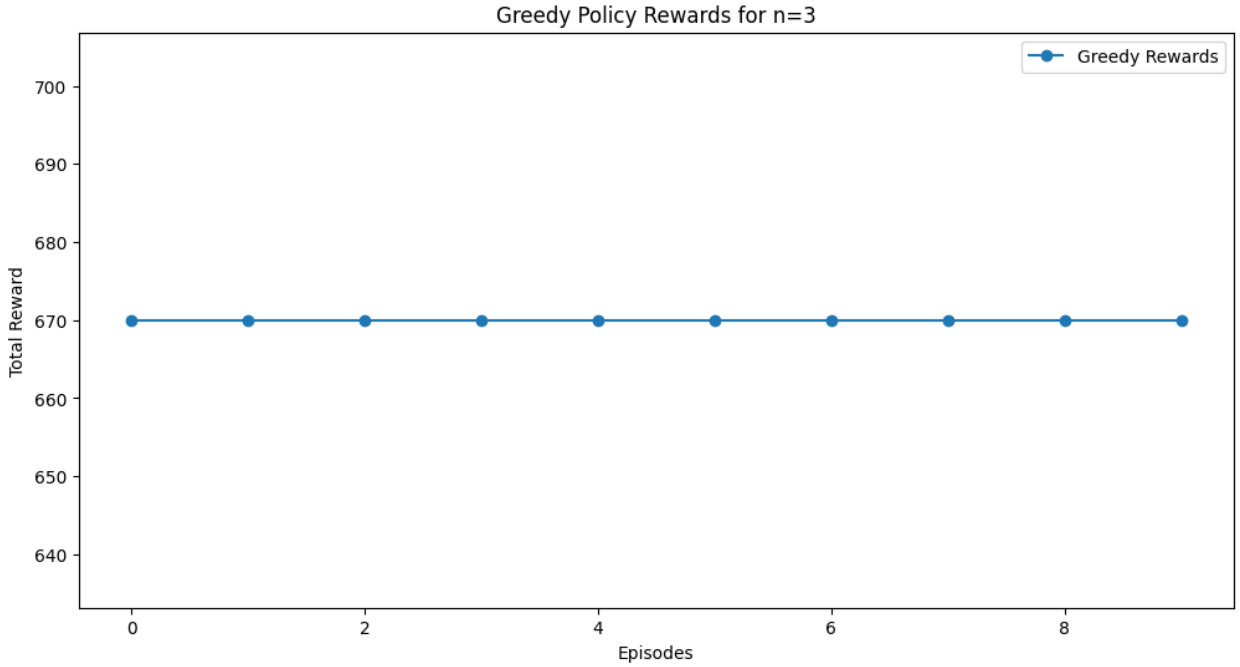
[[4.02005897e+01 1.54212615e+01 1.81672878e+01 6.12429830e+00]
 [4.05175199e+01 4.41946625e+01 6.04907778e+01 3.61283798e+01]
 [4.14780007e+01 4.00072679e+01 6.69153498e+01 4.72392920e+01]
 [1.45146689e+00 3.59906177e+00 1.57753026e+01 5.44845678e+01]
 [1.11844963e+00 1.88860946e+00 0.00000000e+00 1.12100775e+01]]

[[9.97742444e+00 5.59351943e+01 2.12258650e+01 2.31564788e+01]
 [5.10043697e+01 6.67703489e+01 5.66464286e+01 4.79749296e+01]
 [5.85072713e+01 5.96608862e+01 6.82426342e+01 6.02503248e+01]
 [4.43321240e+01 0.00000000e+00 5.12074627e+01 6.74797960e+01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]]

For n =3







n=3, Average Greedy Reward: 670.0
Initial Q-table for n=3 (Q1):
[[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]

[[0. 0. 0. 0.]

```
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [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 for n=3 (Q2):
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]
```

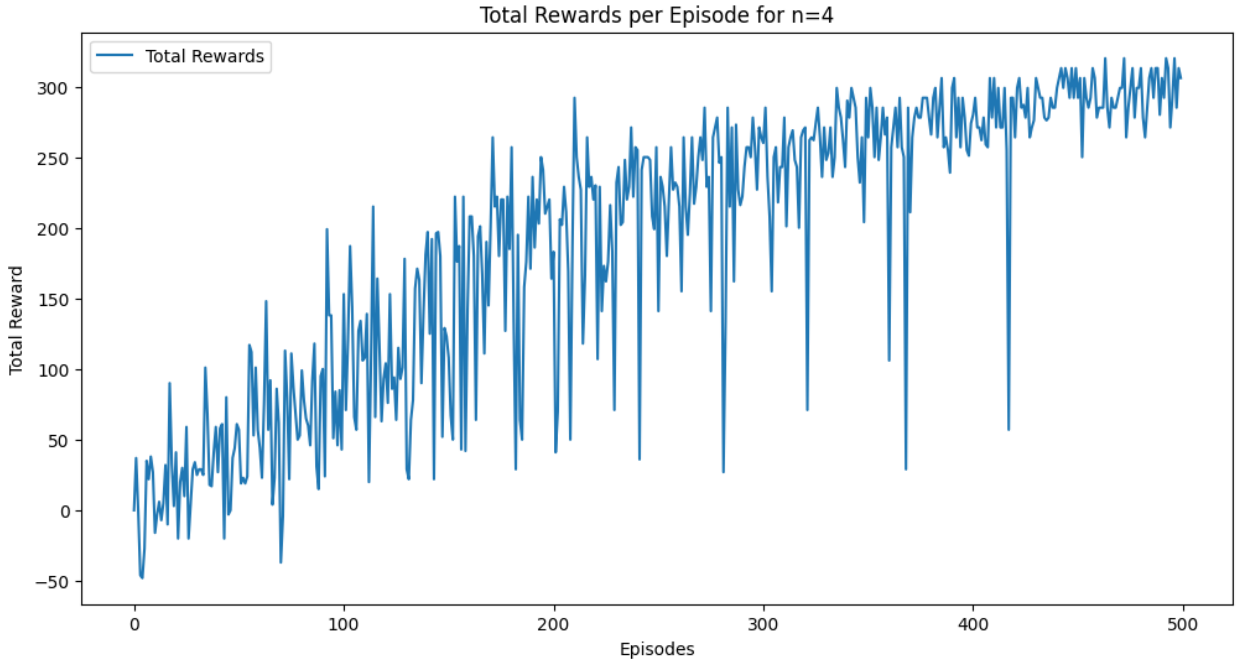
```
 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]]
Trained Q-table for n=3 (Q1):
[[[ 2.71091683e+01  2.60019301e+01  4.11919908e+01  2.24080454e+01]
  [ 4.43915614e+00 -1.43182110e+00  3.28113085e+01  7.93700953e+00]
  [-9.03675351e-01 -3.30576228e-01 -5.93263117e-01  1.33827462e+01]
  [-8.60666236e-01  1.52754389e+00 -1.35003511e+00 -2.46744432e+00]
  [-1.14051502e-02  3.08662911e-01  3.46546925e+00 -5.04490588e-01]]

 [[ 1.85777438e+01  2.65062858e+01  4.73576515e+01  2.92495070e+01]
  [ 9.86241559e+00  6.45935895e+00  4.65464592e+01  1.37635913e+01]
  [-2.92772326e+00  7.46729222e-01  5.90341919e+00  1.57977436e+01]
  [-1.10550305e+00  1.74756930e+00 -2.30586186e+00 -5.41655149e-01]
  [-2.11453278e-01  3.94387938e-01  4.61847847e+00  9.47546380e-01]]

 [[ 2.29895958e+01  5.26230382e+01  3.42802734e+01  3.13879345e+01]
  [ 3.47896236e+01  3.11179049e+01  5.08573845e+01  3.00597511e+01]
  [ 6.58762817e+00  6.91738339e+00  1.78566895e+01  4.58995153e+01]
  [ 1.58729086e+00  1.10416553e+00  1.65320250e+01  2.95586764e+00]
  [ 3.37654280e+00  3.83384463e-01  1.41347242e+01 -3.06821008e+00]]

 [[ 1.25376713e+01  4.75949408e+01  1.58453093e+01  2.15598561e+01]
  [ 3.96145550e+01  4.57814098e+01  5.92863498e+01  3.79189172e+01]
  [ 3.41963098e+01  3.04778777e+01  6.64609326e+01  4.34112573e+01]
  [ 8.78361981e+00  1.29126297e+01  5.13864959e+01  2.98481169e+01]
  [ 4.83849532e-01  7.53837074e+00  0.00000000e+00  2.44716888e+01]]

 [[ 1.40690866e+01  5.05647274e+01  1.70201980e+01  1.56269493e+01]
  [ 4.41953495e+01  6.59993205e+01  4.81763375e+01  4.55193463e+01]
  [ 5.51060460e+01  5.93784399e+01  6.80249516e+01  5.89472245e+01]
  [ 3.85342885e+01  0.00000000e+00  4.93873730e+01  6.62082373e+01]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]
Trained Q-table for n=3 (Q2):
[[[ 2.23896716e+01  2.16881111e+01  4.21325317e+01  2.29485398e+01]
  [ 2.12746547e+00  2.38431777e+00  2.80240427e+01  7.81145700e+00]
  [-1.74183894e-02  5.85370404e-01 -1.81062171e-01  3.92201713e+00]
  [-8.11496618e-03  2.68496903e-01 -8.60814789e-01 -2.10128317e+00]
  [ 4.04626338e-01  2.29577145e-01  2.82209275e+00 -9.65755880e-01]]

 [[ 3.00993814e+01  2.73036482e+01  4.56293171e+01  2.74727054e+01]
  [ 2.62624215e+00  1.15024440e+01  4.76830059e+01  1.17072442e+01]
  [-1.99510148e+00 -2.82962827e-01  1.23345953e+00  1.24172176e+01]
```

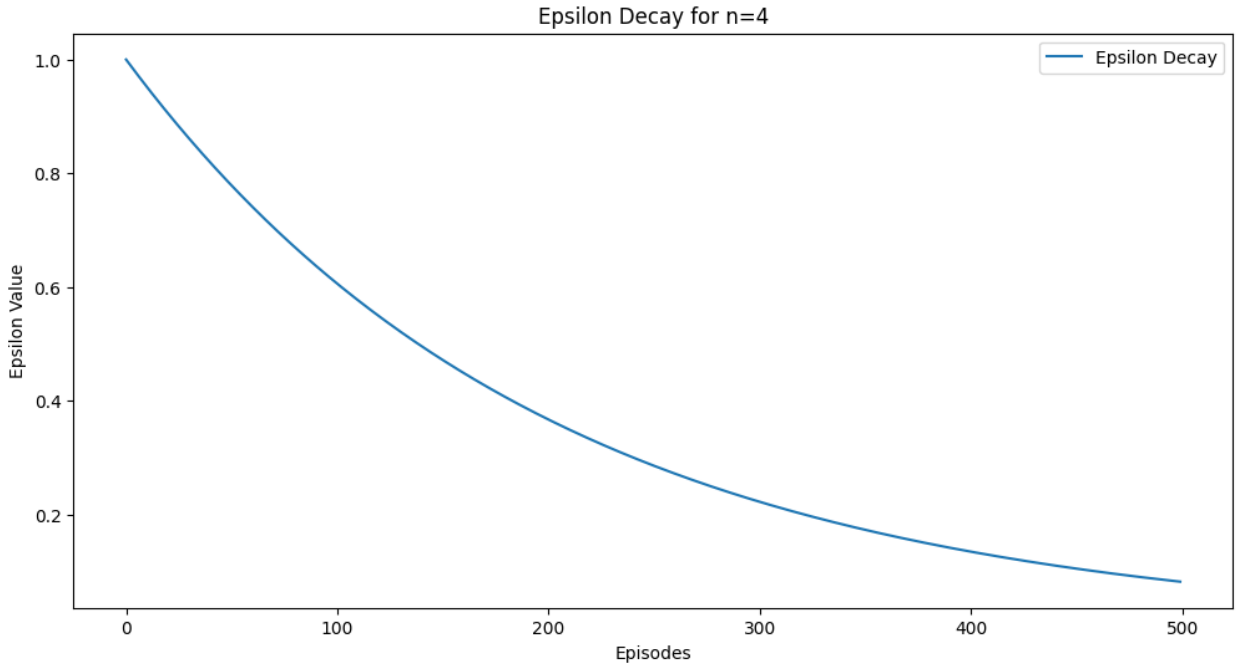
[-4.71642020e-01 1.08777160e+00 -5.49884244e-01 1.56989423e+00]
 [5.40121742e-01 -6.16199105e-01 1.00390271e+01 -3.37543496e-01]]

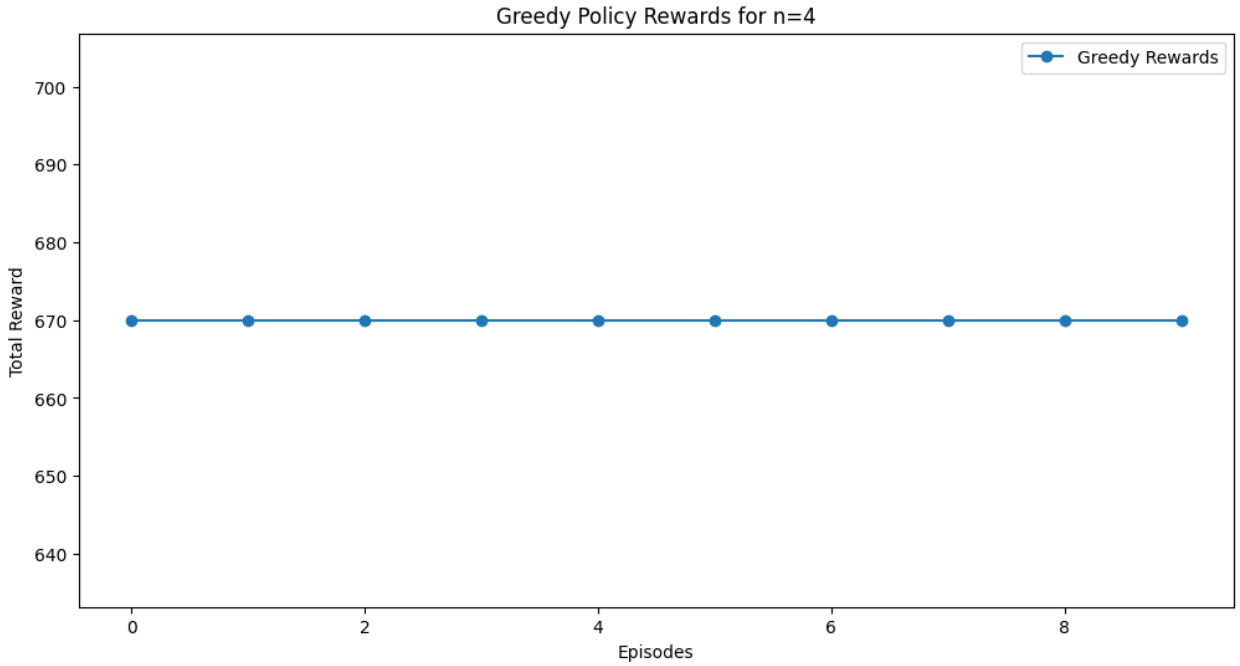
[[2.93459556e+01 5.26233549e+01 3.23670350e+01 3.26040207e+01]
 [2.55373202e+01 3.08675271e+01 5.41572109e+01 2.52331079e+01]
 [6.77632920e+00 2.47422646e+00 1.62604451e+01 4.42321407e+01]
 [-4.25735802e-01 3.18169322e+00 2.18929167e+01 2.74725337e+00]
 [4.14620756e-01 4.52985336e+00 9.79864677e+00 -3.04798762e+00]]

[[2.07471758e+01 4.51672684e+01 1.06313654e+01 1.31107004e+01]
 [4.12222735e+01 4.52798593e+01 5.66905494e+01 3.68155958e+01]
 [3.60644176e+01 3.80634199e+01 6.62465147e+01 4.16299625e+01]
 [7.41760379e+00 1.39470424e+01 5.22801627e+01 2.14557323e+01]
 [-8.35939917e-02 6.08295128e+00 0.00000000e+00 1.76087083e+01]]

[[1.81851891e+01 5.29900234e+01 1.26920484e+01 2.06715626e+01]
 [4.71358777e+01 6.65513590e+01 5.17384776e+01 4.68738066e+01]
 [5.91312595e+01 5.89388770e+01 6.78508408e+01 5.97985208e+01]
 [4.19747754e+01 0.00000000e+00 4.35094886e+01 6.67684463e+01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]]







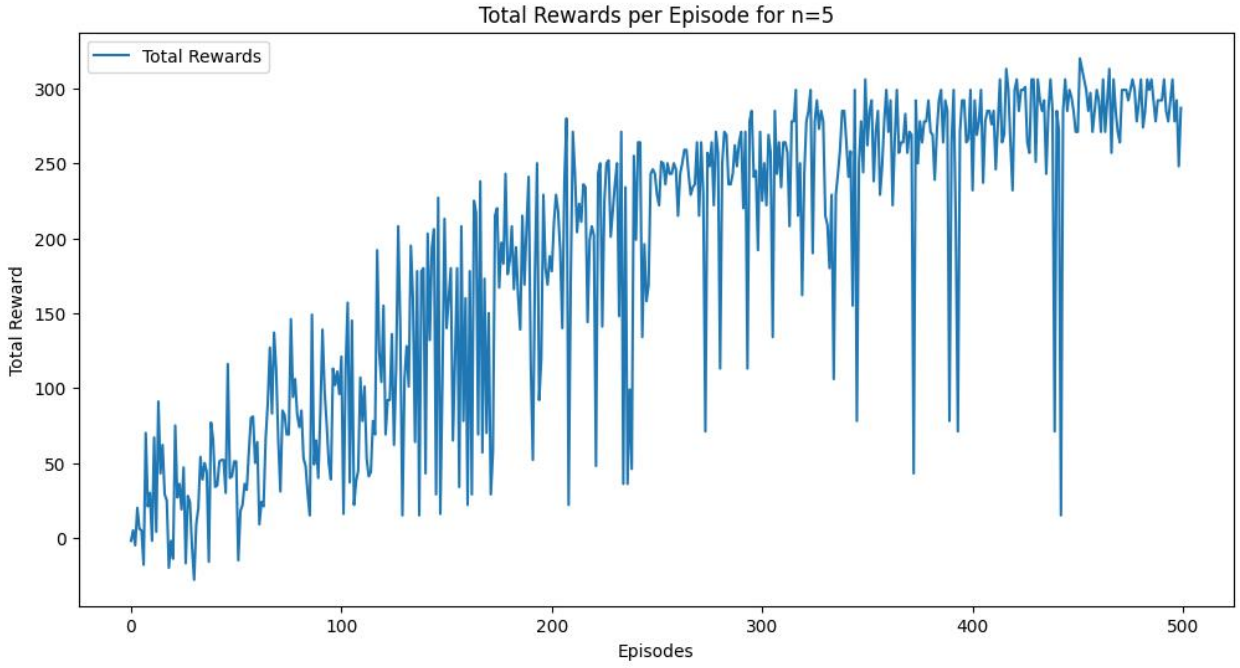
n=4, Average Greedy Reward: 670.0
Initial Q-table for n=4 (Q1):
[[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]

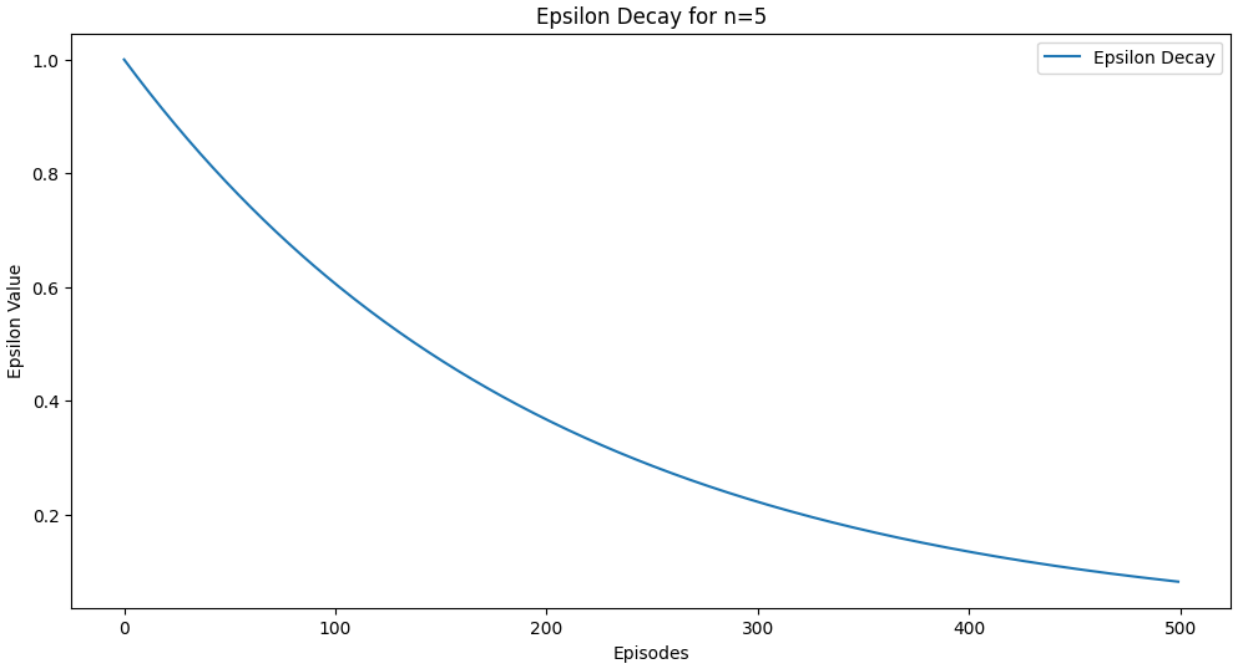
[[0. 0. 0. 0.]

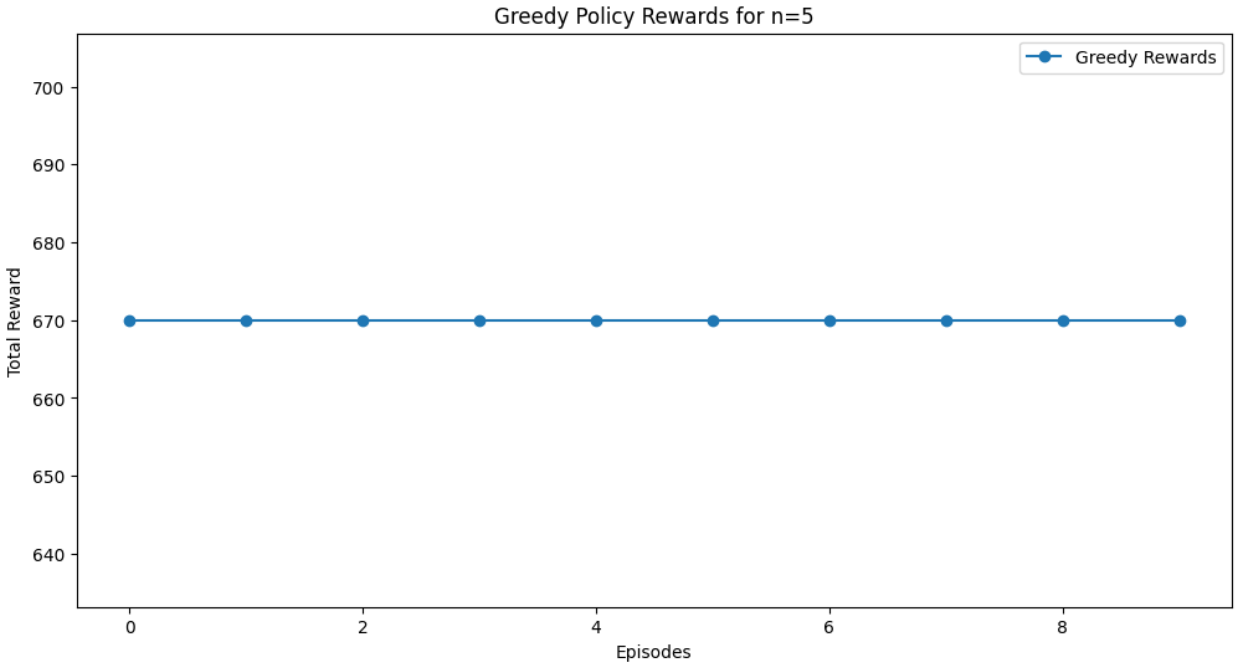
```
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [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 for n=4 (Q2):
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [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 for n=4 (Q1):
[[[ 2.62665427e+01  2.35083778e+01  4.09289842e+01  2.49409750e+01]
  [ 1.26928397e+01 -6.18168205e-01  3.21358765e+01  7.48770343e+00]
  [-2.79779776e+00 -7.04420210e-01 -6.26822580e-01  5.55990560e+00]
  [-3.30236483e-01  1.93227282e-02 -1.82298070e-01 -3.65409847e+00]
  [-4.79772946e-01  4.10346958e-01 -4.10742708e-01 -2.24172337e-01]]

 [[ 1.85205430e+01  2.90332871e+01  4.59263793e+01  2.41723463e+01]
  [ 1.09824972e+01  1.35538492e+01  4.62201388e+01  1.18307432e+01]
  [-3.82066791e+00  3.48047543e+00  6.65442490e+00  1.75405002e+01]
  [-2.79554800e-01 -7.25706389e-01 -2.78073462e+00  5.00301654e+00]
  [-4.18752495e-01  1.97867270e+00 -7.13833280e-01  1.38791931e+00]]

 [[ 2.68928024e+01  5.16822399e+01  3.15152027e+01  3.02254764e+01]
  [ 3.08897989e+01  3.50630300e+01  5.28759107e+01  2.82424165e+01]
  [ 1.04363454e+01  9.70847170e+00  4.89314469e+01  2.19575073e+01]
  [ 1.50386066e+00  1.19463261e+00  3.14039128e+01  8.97319935e+00]
  [-4.97951002e-02 -3.08535948e-01  9.18411101e+00  3.05080978e-01]]

 [[ 1.09292946e+01  4.49426087e+01  1.56518026e+01  1.00821487e+01]
  [ 3.79706533e+01  5.81296975e+01  3.97238457e+01  3.41396479e+01]
  [ 3.76572885e+01  4.53066969e+01  6.68929940e+01  4.47770466e+01]
  [ 1.67666941e+01  1.13724810e+01  5.48595300e+01  3.08294341e+01]
  [ 4.46623904e+00  0.00000000e+00  0.00000000e+00  1.48925734e+01]]

 [[ 1.64394507e+01  4.55416795e+01  1.27294156e+01  1.25586183e+01]
  [ 4.58563148e+01  6.70813577e+01  4.72077531e+01  4.18238714e+01]
  [ 5.84488397e+01  5.79284949e+01  6.76443490e+01  5.96967529e+01]
  [ 4.55314567e+01  0.00000000e+00  4.82579595e+01  6.56328389e+01]
  [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]
Trained Q-table for n=4 (Q2):
[[[ 2.44357020e+01  2.90785858e+01  4.16473517e+01  2.21817264e+01]
  [ 9.83665774e+00  1.56087442e+00  3.34477463e+01  5.07959692e+00]
  [-4.79736891e+00 -6.89465669e-01  1.40974829e+00  6.75631525e+00]
  [-7.13466224e-01 -5.09872200e-01 -1.22622838e+00 -2.50900971e+00]
  [-4.64401600e-01  6.05975857e-01 -8.03914390e-01 -1.89360875e+00]]

 [[ 2.25067377e+01  3.53980682e+01  4.70196400e+01  2.73206061e+01]
  [ 1.10080644e+01  1.10371120e+01  4.70634594e+01  1.40104277e+01]
  [-4.66221539e+00  7.27594353e-01  1.01514458e+01  2.38298044e+01]
```

```
[-5.85720393e-01  7.44476453e-01 -1.44287449e-01  1.09741822e+01]
 [-4.47008558e-02  5.30592832e-01  1.62522686e+00  5.17412403e-01]]

[[ 2.94900227e+01  5.21952194e+01  2.92827774e+01  3.37197880e+01]
 [ 3.47705414e+01  3.26865889e+01  5.33105587e+01  3.39171607e+01]
 [ 1.03389967e+01  4.81960272e+00  5.34973807e+01  1.61438116e+01]
 [ 3.47928575e+00  6.40230961e+00  2.19256751e+01  3.64909489e+00]
 [ 1.19893651e+00  1.54910028e+00  7.52170077e+00  4.00559755e+00]]

[[ 1.20713346e+01  4.31059582e+01  1.67785138e+01  1.19885577e+01]
 [ 3.88533747e+01  5.98649688e+01  4.74649023e+01  3.34832938e+01]
 [ 4.44390089e+01  4.46783235e+01  6.68445210e+01  4.76256362e+01]
 [ 9.34881447e+00  1.47762168e+01  5.33389976e+01  3.01344618e+01]
 [ 1.71287416e+00  9.17509255e+00  0.00000000e+00  2.17228939e+01]]

[[ 1.80092482e+01  4.96113043e+01  7.79689226e+00  2.23019767e+01]
 [ 4.14041903e+01  6.56921483e+01  4.79088326e+01  4.38815560e+01]
 [ 5.94859602e+01  5.86518398e+01  6.75508550e+01  5.88681875e+01]
 [ 3.64305651e+01  0.00000000e+00  4.95589807e+01  6.57943201e+01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]
```







n=5, Average Greedy Reward: 670.0
Initial Q-table for n=5 (Q1):
[[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]

[[0. 0. 0. 0.]

```
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [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 for n=5 (Q2):
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [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 for n=5 (Q1):
[[[23.11909147 25.91106862 40.92038181 22.94374029]
  [ 5.58921117  4.57802467 28.94669721  5.49507406]
  [-1.73684635 -0.08896497 11.6417538   1.40192304]
  [ 0.88566856 -0.66149569 -0.64843387 -1.91119024]
  [-0.10346734 -0.86999879  2.22896287 -1.14457778]]

 [[23.17801907 21.25715862 45.8763625  29.62869733]
  [12.28275246 16.75017001 44.43620431 18.45088286]
  [ 1.94757266  3.14558676 10.90345046 29.18209269]
  [-0.26394569  4.71046593 -3.3626573   8.29616897]
  [ 0.2079893  -0.26181959  2.75231708  2.2645598 ]]

 [[24.56250425 51.16872472 31.85777481 29.73204825]
  [29.35340623 38.22827262 52.65779549 25.735773  ]
  [10.42241218 17.51921957 54.71082439 23.8940738 ]
  [ 2.48263448 -0.3278525  23.5346639   7.24110099]
  [ 1.47873635  4.78632929  4.70540907 -1.19760831]]

 [[11.71447113 49.25033574 22.97358023 20.71558155]
  [41.224283   57.93799041 41.36794257 32.9311705 ]
  [44.14117587 42.90629829 64.57217061 44.72926956]
  [ 7.17475788  8.37818634 22.07846854 51.75451951]
  [ 2.6906202   2.39727769  0.         12.87853533]]

 [[21.55996951 43.52697971 10.52169221  9.56086522]
  [39.57518149 64.69775443 45.74291246 39.59830437]
  [56.80230384 57.87010416 62.10267584 58.32594825]
  [34.23296822  0.         46.12254577 65.88147517]
  [ 0.         0.         0.         0.        ]]]
Trained Q-table for n=5 (Q2):
[[[ 2.44620404e+01  2.27153489e+01  4.06744186e+01  1.60649438e+01]
  [ 7.37716407e+00 -1.36987847e+00  3.19358006e+01  3.84838891e+00]
  [-1.54062588e+00 -4.06810382e-01  1.28223966e+01  1.30961783e+00]
  [ 5.50399469e-01  1.25559235e-01  4.82369539e-01 -1.94243169e+00]
  [ 9.27175977e-01 -2.51600941e-02 -4.12572084e-01 -3.64495202e-01]]

 [[ 2.81921121e+01  3.02795525e+01  4.52006882e+01  2.27380362e+01]
  [ 1.34181601e+01  1.57521366e+01  4.53136987e+01  1.24683669e+01]
  [ 5.48923230e-01  7.74227304e+00  2.55921572e+00  2.43663900e+01]
```

```
 [-6.47157365e-01  1.33244537e+00  6.92169653e-01  1.18022233e+01]
 [-3.71867331e-01 -2.25855561e-01  6.03053880e-01  6.14687217e+00]]

[[ 2.99651290e+01  5.24562503e+01  3.38703647e+01  3.31281749e+01]
 [ 2.95494319e+01  3.33160837e+01  5.09115832e+01  2.81397917e+01]
 [ 1.38568496e+01  4.89739276e+00  5.34024859e+01  1.79231254e+01]
 [-8.74396647e-01  3.39438591e+00  2.35253526e+01  5.00460710e+00]
 [ 1.81650281e+00  1.68228934e+00  1.03658874e+01 -3.08704031e+00]]

[[ 1.52405101e+01  3.99990272e+01  1.22919248e+01  8.53603159e+00]
 [ 4.35880295e+01  5.86055334e+01  3.85208183e+01  3.43537981e+01]
 [ 4.64584450e+01  4.25175954e+01  6.49098581e+01  4.53542376e+01]
 [ 7.83001874e+00  9.20921831e+00  2.49254941e+01  5.24389176e+01]
 [ 2.19240271e+00  3.26736613e+00  0.00000000e+00  1.27693341e+01]]

[[ 1.84937068e+01  4.65743589e+01  1.70863740e+01  1.27360981e+01]
 [ 3.91430294e+01  6.55379755e+01  4.86322508e+01  3.99440287e+01]
 [ 5.61576482e+01  5.91854916e+01  6.31207518e+01  5.82280661e+01]
 [ 4.11675182e+01  0.00000000e+00  4.81528794e+01  6.56666722e+01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]]
```

5. Compare the performance of SARSA and n-step Double Q-learning algorithms on the same environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.

Initial Q-Tables:

Both SARSA and n-step Double Q-learning start with Q-tables initialized to zero, which means the agent begins with no knowledge of the environment. Over time, these tables are updated as the agent learns the values of different state-action pairs.

SARSA Results:

The Q-table from SARSA shows the values it has learned for each state-action pair. Positive values indicate actions that lead to higher cumulative rewards. For example:

- The state (4, 2) has high Q-values for certain actions, showing that SARSA has identified this state as part of the path to the goal.
- Similarly, (5, 3) is close to the goal, so it has some of the highest Q-values in the table.

However, the values for states further from the goal are lower, and in some cases negative. This is expected because SARSA is an **on-policy** algorithm—it learns from the actions it takes based on its current policy. While effective, it sometimes struggles to fully explore or optimize for all states, especially those less visited.

n-step Double Q-learning Results (n=1):

In n-step Double Q-learning, two separate Q-tables (Q1 and Q2) are maintained. Initially, both are set to zero. During training, the two tables alternate updates, which helps prevent overestimating action values.

After training:

- Both Q1 and Q2 tables show high positive values for states near the goal.
- For example, states in key regions close to the terminal state show consistent and high Q-values, reflecting a well-learned policy.
- The **average greedy reward** achieved (670.0) confirms that the agent has learned an effective policy for maximizing rewards in this environment.

Comparison: SARSA vs. n-step Double Q-learning

1. **Learning Quality**
    a. SARSA produces a functional policy, with good Q-values near the goal. However, some values are suboptimal or uneven, especially in less-explored states.
    b. n-step Double Q-learning generates more consistent and balanced Q-values across the board. The dual Q-table setup helps reduce bias and results in better policy optimization.
2. Reward Maximization
    a. The trained Double Q-learning policy leads to higher average rewards compared to SARSA, as shown by the **670.0 greedy reward** value.
    b. SARSA performs well but may leave some potential rewards untapped due to its tendency to learn more cautiously.
3. **Exploration and Stability**
    a. SARSA's on-policy nature means it explores as it learns, making it stable in dynamic environments but slower to converge in deterministic ones.
    b. Double Q-learning, on the other hand, focuses on exploitation once trained, which helps it optimize faster and achieve better results in deterministic environments like this one.

To sum up:

- SARSA is reliable for environments with variability or when exploration is critical.

- n-step Double Q-learning excels in deterministic settings, producing a more optimal policy with higher cumulative rewards.
- For this specific scenario, **n-step Double Q-learning** clearly performs better, as evidenced by the higher average reward and more balanced Q-values.

Bonus - Grid-World Environment Visualization

Scenario Overview

1. **Grid Environment**:
   a. The environment is a 5x5 grid.
   b. The agent starts at (0, 0) and must reach the goal at (4, 4).
   c. The grid contains specific cells with rewards, penalties, and obstacles:
      i. Rewards: Cells like (2, 1) and (4, 2) offer positive points (+5, +7).
      ii. Penalties: Some cells, such as (0, 2), impose negative points (-5).
      iii. Obstacles: Cells like (1, 1) and (3, 3) block the agent's movement.
2. **Agent Actions**:
   d. The agent can move up, down, left, or right. Movement is restricted by grid boundaries and obstacles.
3. **Visualization**:
   e. The grid is visually rendered after each step. Custom icons represent:
      i. The agent (a black figure).
      ii. The goal (a flag icon).
      iii. Rewards (coins or stars).
      iv. Obstacles (prohibited symbols).

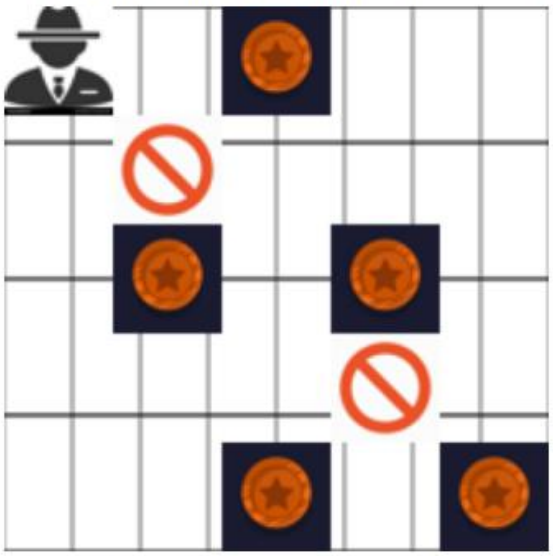
Execution

The agent takes random actions within the grid and for each step:

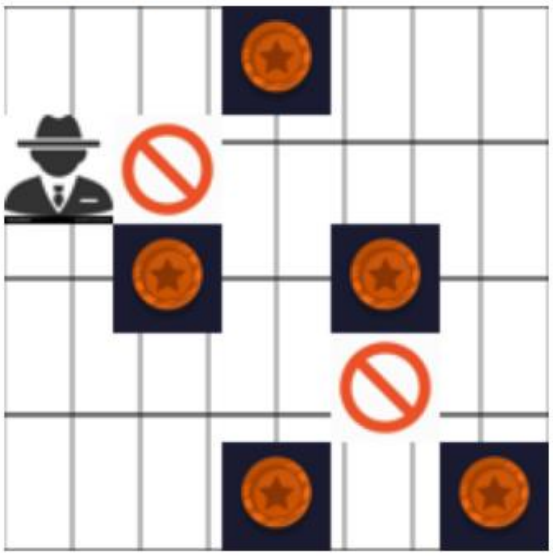
- The agent’s position is updated.
- Any rewards or penalties are applied based on the cell it moves to.
- A check determines whether the agent has reached the goal.
- Visualizations show the agent’s progress on the grid.

Visualizations:

```
State: [0 0], Action: 0, Reward: 0, Next State: [0 0]
```



```
State: [0 0], Action: 2, Reward: 0, Next State: [1 0]
```



```
State: [1 0], Action: 0, Reward: 0, Next State: [0 0]
```

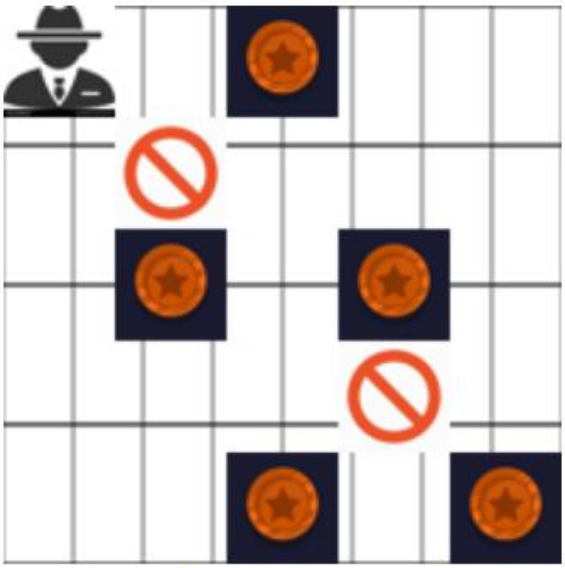
The agent starts at the top-left corner (0, 0) and aims to reach the bottom-right corner (4, 4) on a 5x5 grid. Along the way, rewards (stars) and obstaclesare placed throughout the grid.

Agent's movements:

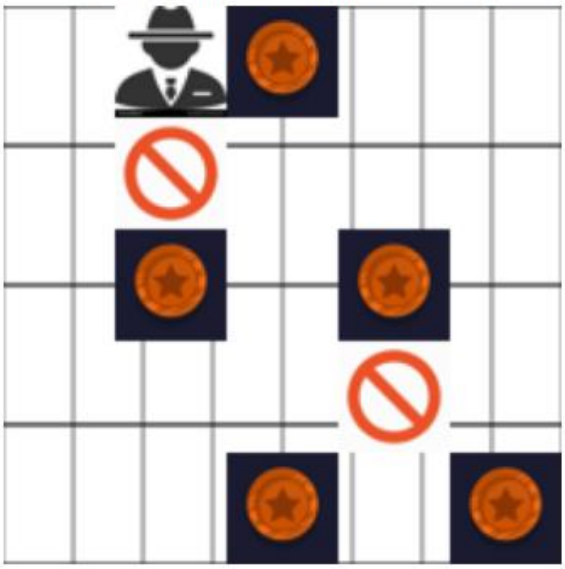
- From (0, 0), Action 0 (Up) keeps the agent in the same spot.
- Action 2 (Down) moves the agent to (1, 0).
- At (1, 0), Action 0 (Up) takes the agent back to (0, 0).

The results below show the agent’s transition:

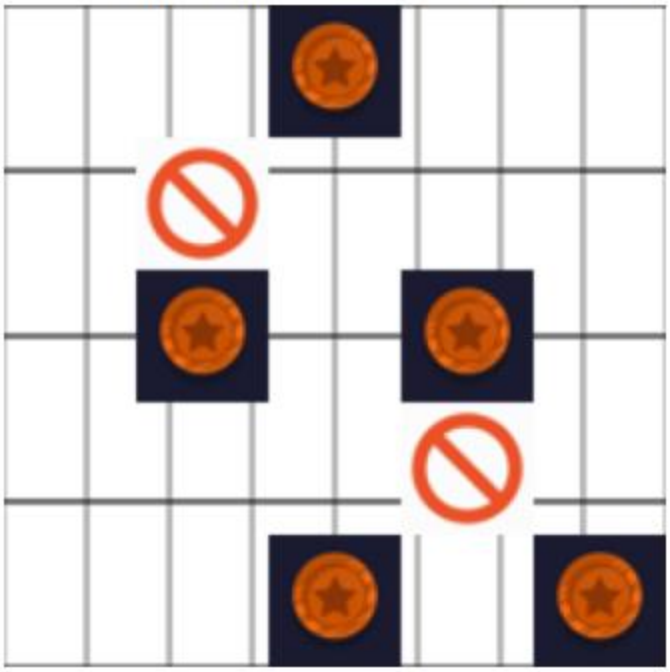
State: [1 0], Action: 0, Reward: 0, Next State: [0 0]



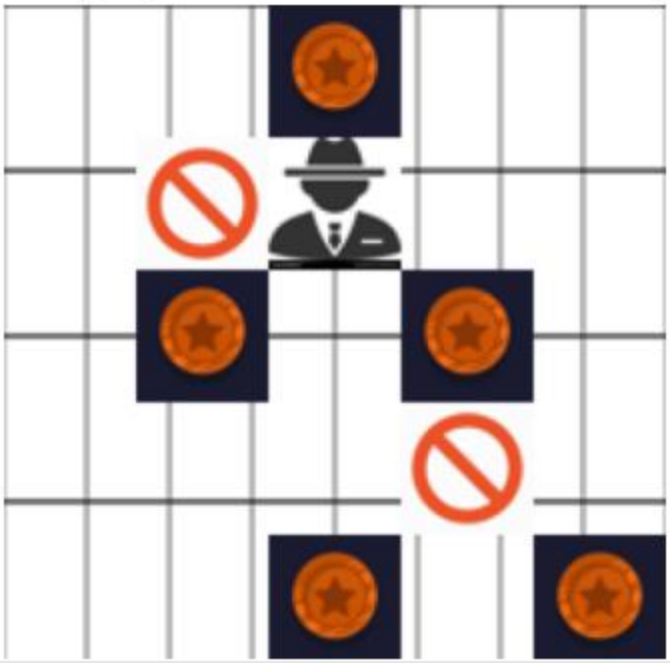
State: [0 0], Action: 1, Reward: 0, Next State: [0 1]



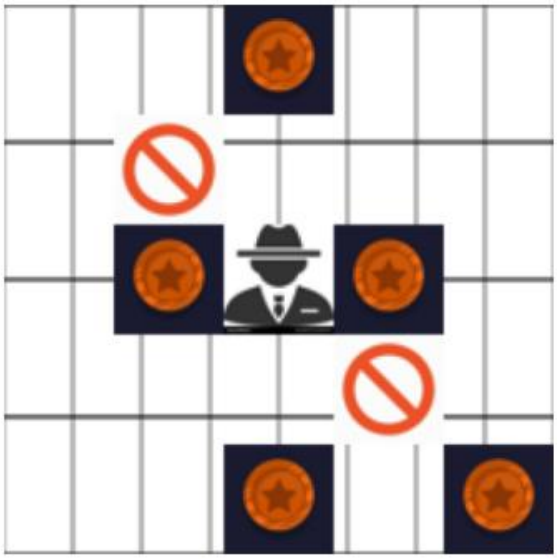
State: [0 1], Action: 2, Reward: 0, Next State: [1 1]



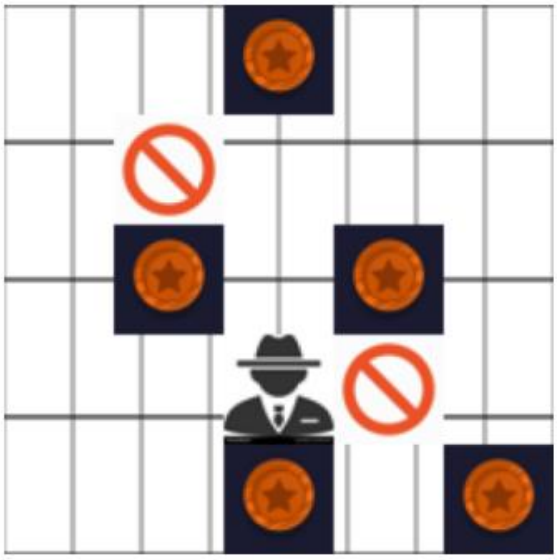
State: [1 1], Action: 1, Reward: 0, Next State: [1 2]



```
State: [1 2], Action: 2, Reward: 0, Next State: [2 2]
```



```
State: [2 2], Action: 2, Reward: 0, Next State: [3 2]
```



```
State: [3 2], Action: 1, Reward: 0, Next State: [3 3]
```

The above transitions indicate the movement of agent avoiding obstacles, collecting rewards and moving towards the goal state.

References:

- https://www.geeksforgeeks.org/sarsa-reinforcement-learning/
- https://www.geeksforgeeks.org/q-learning-in-python/
- https://www.geeksforgeeks.org/what-is-reinforcement-learning/