

Experiment – 7

```
# --- Imports ---
import random
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import clear_output
from collections import defaultdict
import pickle

# reproducibility
np.random.seed(0)
random.seed(0)
```

```
class QLearningAgent:
    def __init__(self, actions, alpha=0.5, gamma=0.99, epsilon=0.2):
        self.actions = actions
        self.alpha = alpha
        self.gamma = gamma
        self.epsilon = epsilon
        self.Q = defaultdict(lambda: np.zeros(len(actions), dtype=float))

    def act(self, state, greedy=False):
        if (not greedy) and (np.random.rand() < self.epsilon):
            return int(np.random.choice(self.actions))
        q = self.Q[state]
        return int(np.argmax(q))

    def update(self, s, a, r, s2, done):
        qsa = self.Q[s][a]
        if done:
            target = r
        else:
            target = r + self.gamma * np.max(self.Q[s2])
        self.Q[s][a] = qsa + self.alpha * (target - qsa)
```

```
# Define a sample grid (adjust size and layout as you like)
grid = [
    'S.....',
    '..R.H.',
    '...R..',
    '.H....',
    '..R...'
]

env = RecyclingGrid(grid, max_steps=200)
actions = [0, 1, 2, 3] # up, right, down, left

agent = QLearningAgent(actions, alpha=0.6, gamma=0.95, epsilon=0.2)
```

```

# Training parameters
episodes = 1000
rewards = []

for ep in range(episodes):
    state = env.reset()
    total_r = 0.0
    done = False
    while not done:
        a = agent.act(state)          # epsilon-greedy action
        s2, r, done, _ = env.step(a)  # take step
        agent.update(state, a, r, s2, done)
        state = s2
        total_r += r
    rewards.append(total_r)

# slowly decay epsilon
if ep % 50 == 0 and ep > 0:
    agent.epsilon = max(0.01, agent.epsilon * 0.995)

# progress print
if ep % 100 == 0:
    clear_output(wait=True)
    print(f"Episode {ep}/{episodes}, recent avg reward: {np.mean(rewards[-100:]).2f}")

# Plot training rewards
plt.figure(figsize=(8,4))
plt.plot(rewards, alpha=0.7)
plt.xlabel('Episode')
plt.ylabel('Total Reward')
plt.title('Training Rewards')
plt.grid(True)
plt.show()

```

OUTPUT:

Episode 900/1000, recent avg reward: 28.05



```
Episode 1: total reward = 29.50
S.....
....H.
.....
.H....
..A...
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (4, 2)]
-----
Episode 2: total reward = 29.50
S.....
....H.
.....
.H....
..A...
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (4, 2)]
-----
Episode 3: total reward = 29.50
S.....
....H.
.....
.H....
..A...
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (4, 2)]
-----
Episode 4: total reward = 29.50
S.....
....H.
.....
.H....
..A...
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (4, 2)]
-----
Episode 5: total reward = 29.50
S.....
....H.
.....
.H....
..A...
Path: [(0, 0), (0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 3), (4, 3), (4, 2)]
-----
```