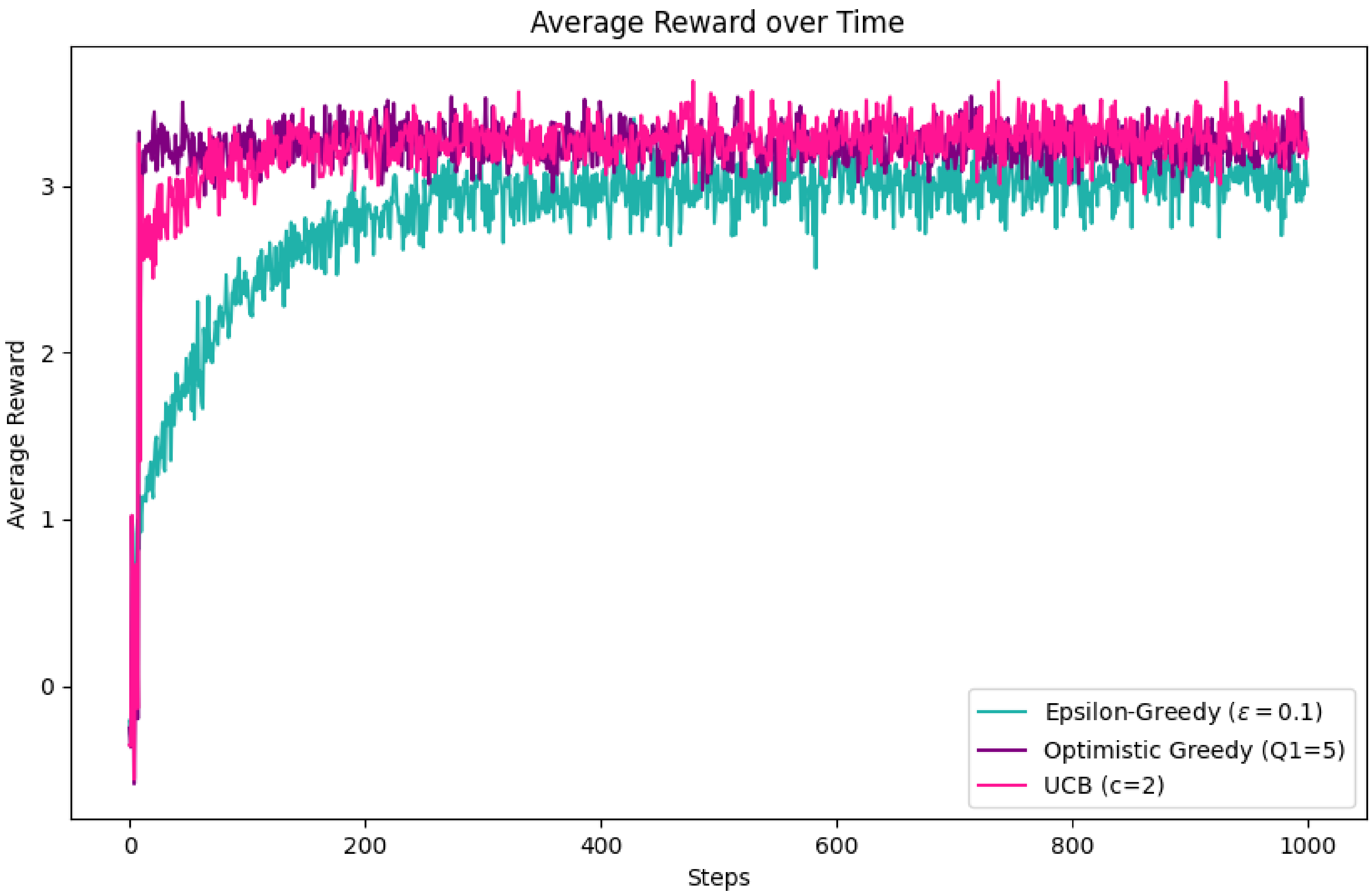
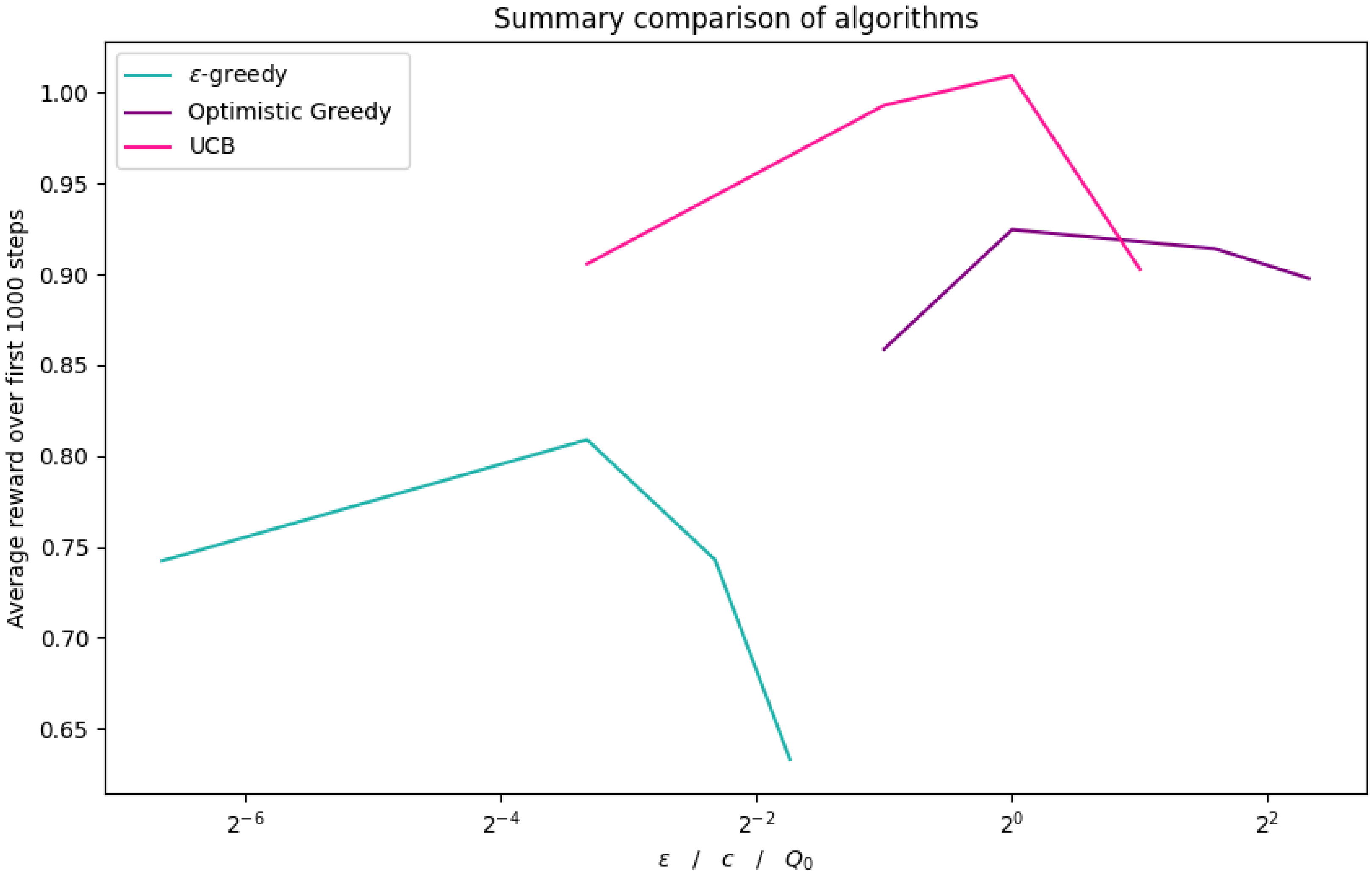
```python
# Nihal Ranchod - 2427378
# Lisa Godwin - 2437980

import numpy as np
import matplotlib.pyplot as plt

class Bandit:
    def __init__(self, k):
        self.k = k
        self.means = np.random.normal(0, np.sqrt(3), k)

    def pull(self, arm):
        return np.random.normal(self.means[arm], 1)

def epsilon_greedy(bandit, epsilon, steps):
    k = bandit.k
    Q = np.zeros(k)
    N = np.zeros(k)
    rewards = np.zeros(steps)

    for t in range(steps):
        if np.random.rand() < epsilon:
            arm = np.random.choice(k)
        else:
            arm = np.argmax(Q)

        reward = bandit.pull(arm)
        N[arm] += 1
        Q[arm] += (reward - Q[arm]) / N[arm]
        rewards[t] = reward

    return rewards

def greedy_optimistic(bandit, Q1, steps):
    k = bandit.k
    Q = np.ones(k) * Q1
    N = np.zeros(k)
    rewards = np.zeros(steps)

    for t in range(steps):
        arm = np.argmax(Q)
        reward = bandit.pull(arm)
        N[arm] += 1
        Q[arm] += (reward - Q[arm]) / N[arm]
        rewards[t] = reward

    return rewards

def ucb(bandit, c, steps):
    k = bandit.k
    Q = np.zeros(k)
    N = np.zeros(k)
    rewards = np.zeros(steps)

    for t in range(steps):
        if t < k:
            arm = t
        else:
            ucb_values = Q + c * np.sqrt(np.log(t + 1) / (N + 1e-5))
            arm = np.argmax(ucb_values)

        reward = bandit.pull(arm)
        N[arm] += 1
        Q[arm] += (reward - Q[arm]) / N[arm]
        rewards[t] = reward

    return rewards

def run_simulation(algorithm, bandit, param, steps, runs):
    all_rewards = np.zeros((runs, steps))
    for run in range(runs):
        rewards = algorithm(bandit, param, steps)
        all_rewards[run] = rewards
    return np.mean(all_rewards, axis=0)

def main():
    # Parameters
    k = 10
    steps = 1000
    runs = 100

    # Initialize bandit
    bandit = Bandit(k)

    # Run simulations
    epsilon_rewards = run_simulation(epsilon_greedy, bandit, 0.1, steps, runs)
    optimistic_rewards = run_simulation(greedy_optimistic, bandit, 5, steps, runs)
    ucb_rewards = run_simulation(ucb, bandit, 2, steps, runs)

    # Plot results
    plt.figure(figsize=(10, 6))
    plt.plot(epsilon_rewards, label='Epsilon-Greedy ($\epsilon=0.1$)', color='lightseagreen')
    plt.plot(optimistic_rewards, label='Optimistic Greedy (Q1=5)', color='purple')
    plt.plot(ucb_rewards, label='UCB (c=2)', color='deeppink')
    plt.xlabel('Steps')
    plt.ylabel('Average Reward')
    plt.legend()
    plt.title('Average Reward over Time')
    plt.savefig('Avarege_Reward_over_Time.png')
    plt.show()


    # Part 2: Summary of Comparsion Plot with different hyperparameters

    # Hyperparameter values
    epsilon_values = [0.01, 0.1, 0.2, 0.3]
    Q1_values = [5, 3, 1, 0.5]
    c_values = [0.1, 0.5, 1, 2]

    # Average rewards for different hyperparameters
    epsilon_rewards = [np.mean(run_simulation(epsilon_greedy, bandit, epsilon, steps, runs)) for epsilon in epsilon_values]
    Q1_rewards = [np.mean(run_simulation(greedy_optimistic, bandit, Q1, steps, runs)) for Q1 in Q1_values]
    c_rewards = [np.mean(run_simulation(ucb, bandit, c, steps, runs)) for c in c_values]

    # Plot results
    plt.figure(figsize=(10, 6))
    plt.plot(epsilon_values, epsilon_rewards, label='$\epsilon$-greedy', color='lightseagreen')
    plt.plot(Q1_values, Q1_rewards, label='Optimistic Greedy ', color='purple')
    plt.plot(c_values, c_rewards, label='UCB', color='deeppink')
    plt.xscale('log', base=2)
    plt.xlabel('$\epsilon \quad / \quad c \quad / \quad Q_0$')
    plt.ylabel('Average reward over first 1000 steps')
    plt.legend()
    plt.title('Summary comparison of algorithms')
    plt.savefig('Summary_comparison_of_algorithms.png')
    plt.show()


if __name__ == '__main__':
    main()
```


Average Reward over Time
Average Reward
Steps
0
1
2
3
0
200
400
600
800
1000
Epsilon-Greedy ($\varepsilon = 0.1$)
Optimistic Greedy (Q1=5)
UCB (c=2)


Summary comparison of algorithms
ε-greedy
Optimistic Greedy
UCB
Average reward over first 1000 steps
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
2^{-6}
2^{-4}
2^{-2}
2^{0}
2^{2}
ε / c / Q_0