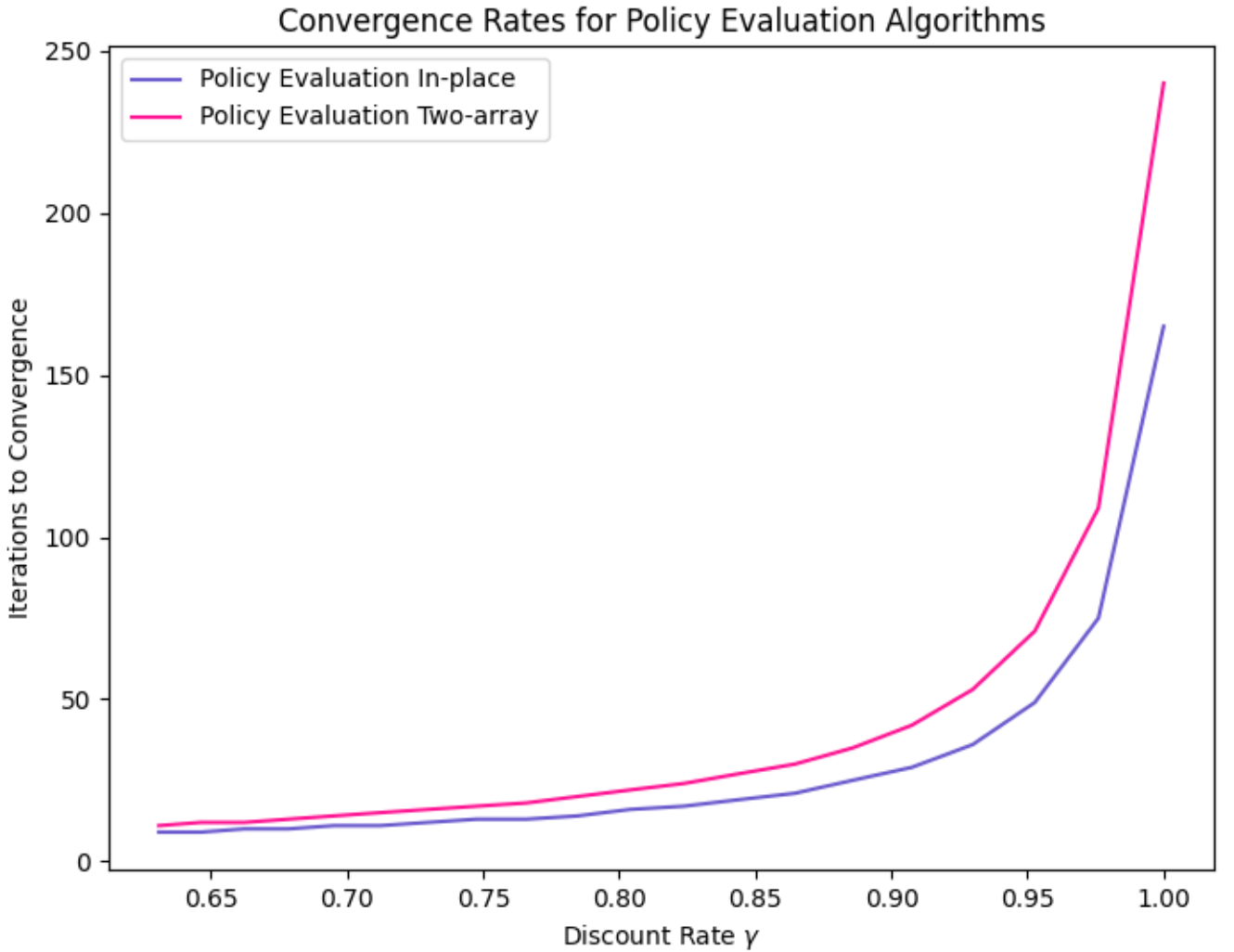
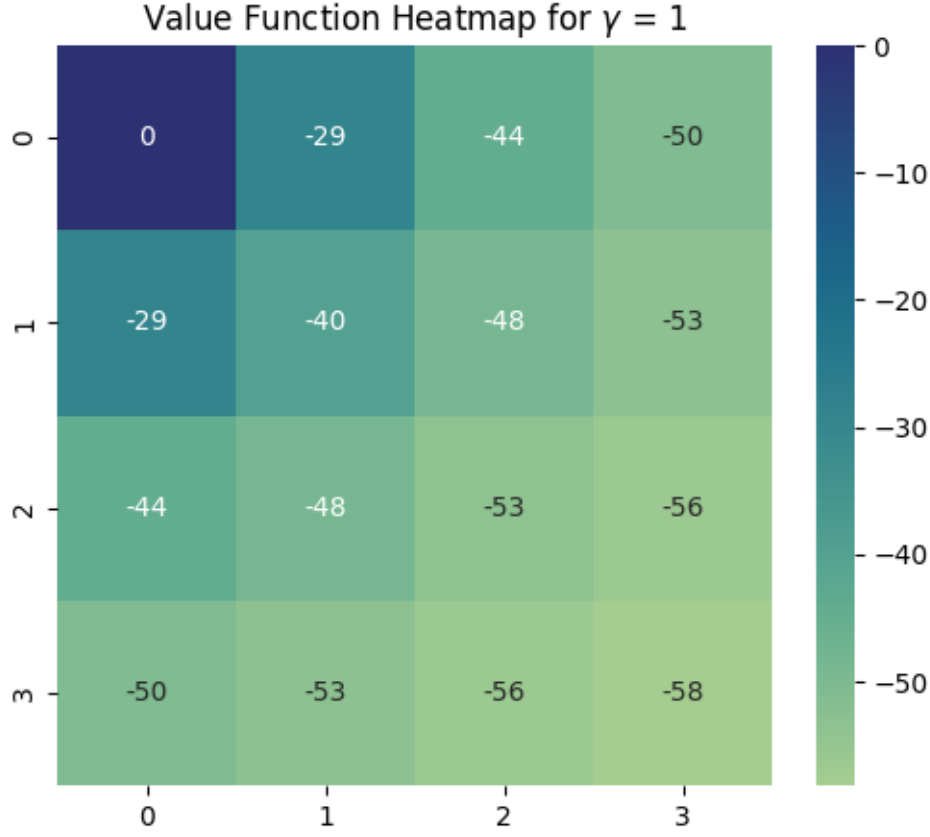
```python
#Nihal Ranchod - 2427378
#Lisa Godwin -2437980

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

class GridworldMDP:
    def __init__(self):
        self.grid_size = 4
        self.initial_state = (3, 0)
        self.goal_state = (0, 0)
        self.actions = ['up', 'down', 'left', 'right']
        self.state = self.initial_state

    def step(self, state, action):
        if action not in self.actions:
            raise ValueError("Invalid action")

        x, y = state
        if action == 'up':
            x = max(x - 1, 0)
        elif action == 'down':
            x = min(x + 1, self.grid_size - 1)
        elif action == 'left':
            y = max(y - 1, 0)
        elif action == 'right':
            y = min(y + 1, self.grid_size - 1)

        new_state = (x, y)
        reward = -1
        if new_state == self.goal_state:
            reward = 0

        return new_state, reward

    def is_terminal(self, state):
        return state == self.goal_state

# Policy Evaluation in Place:
# The value function is updated immediately after evaluating each state.
# The new value of a state is used in the subsequent evaluations of other states within the same iteration.
def policy_evaluation_in_place(env, gamma, theta):
    V = np.zeros((env.grid_size, env.grid_size))
    iterations = 0
    while True:
        delta = 0
        for x in range(env.grid_size):
            for y in range(env.grid_size):
                state = (x, y)
                if env.is_terminal(state):
                    continue
```

```
                v = V[state]
                V[state] = sum(1/len(env.actions) * (reward + gamma * V[new_state])
                               for action in env.actions
                               for new_state, reward in [env.step(state, action)])
                delta = max(delta, abs(v - V[state]))
        iterations += 1
        if delta < theta:
            break
    return V, iterations

# Policy Evaluation with Two Arrays:
# A temporary array (V_new) is used to store the updated values of the states.
# The value function V is only updated after all states have been evaluated in the current iteration
def policy_evaluation_two_array(env, gamma, theta):
    V = np.zeros((env.grid_size, env.grid_size))
    iterations = 0
    while True:
        delta = 0
        V_new = np.copy(V)
        for x in range(env.grid_size):
            for y in range(env.grid_size):
                state = (x, y)
                if env.is_terminal(state):
                    continue
                V_new[state] = sum(1/len(env.actions) * (reward + gamma * V[new_state])
                                   for action in env.actions
                                   for new_state, reward in [env.step(state, action)])
                delta = max(delta, abs(V_new[state] - V[state]))
        V = V_new
        iterations += 1
        if delta < theta:
            break
    return V, iterations

def main():
    env = GridworldMDP()
    theta = 0.01
    gammas = np.logspace(-0.2, 0, num=20)

    iterations_in_place = []
    iterations_two_array = []

    for gamma in gammas:
        _, it_in_place = policy_evaluation_in_place(env, gamma, theta)
        _, it_two_array = policy_evaluation_two_array(env, gamma, theta)
        iterations_in_place.append(it_in_place)
        iterations_two_array.append(it_two_array)

    V, _ = policy_evaluation_in_place(env, 1, theta)
```

```
    plt.figure(figsize=(6, 5))
    sns.heatmap(V, annot=True, cmap='crest', cbar=True)
    plt.title('Value Function Heatmap for $\gamma$ = 1')
    plt.show()

    plt.figure(figsize=(8, 6))
    plt.plot(gammas, iterations_in_place, label='Policy Evaluation In-place', color="slateblue")
    plt.plot(gammas, iterations_two_array, label='Policy Evaluation Two-array', color="deeppink")
    plt.xlabel('Discount Rate $\gamma$')
    plt.ylabel('Iterations to Convergence')
    plt.title('Convergence Rates for Policy Evaluation Algorithms')
    plt.legend()
    plt.show()

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


Convergence Rates for Policy Evaluation Algorithms
Policy Evaluation In-place
Policy Evaluation Two-array
Iterations to Convergence
Discount Rate γ
0
50
100
150
200
250
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00


Value Function Heatmap for $\gamma = 1$
0
-29
-44
-50
-29
-40
-48
-53
-44
-48
-53
-56
-50
-53
-56
-58
0
1
2
3
0
−10
−20
−30
−40
−50