import numpy as np

# Q-learning algoritması için gerekli parametreler

learning\_rate = 0.1

discount\_factor = 0.9

num\_episodes = 100000

# Oyun durumu ve aksiyon sayıları

num\_states = 3

num\_actions = 2

# Q-table'ı başlangıçta rastgele değerlerle doldur

q\_table = np.random.rand(num\_states, num\_actions)

# Oyuncu 1 için eylem seçimi

def select\_action\_player1(state):

return np.argmax(q\_table[state])

# Oyuncu 2 için eylem seçimi

def select\_action\_player2(state):

return np.argmax(q\_table[min(state, num\_states - 1), :])

# Oyunda bir adım gerçekleştirme

def take\_step(state\_player1, state\_player2):

action\_player1 = select\_action\_player1(state\_player1)

action\_player2 = select\_action\_player2(state\_player2)

# Oyuncuların eylemleri üzerinden ödeme hesaplama

if action\_player1 == 0 and action\_player2 == 0:

payoff\_player1 = 3

payoff\_player2 = 3

elif action\_player1 == 0 and action\_player2 == 1:

payoff\_player1 = 1

payoff\_player2 = 4

elif action\_player1 == 1 and action\_player2 == 0:

payoff\_player1 = 4

payoff\_player2 = 1

else:

payoff\_player1 = 2

payoff\_player2 = 2

# Q-table güncelleme

q\_table[state\_player1, action\_player1] += learning\_rate \* (payoff\_player1 + discount\_factor \* np.max(q\_table[state\_player1]) - q\_table[state\_player1, action\_player1])

q\_table[state\_player2, action\_player2] += learning\_rate \* (payoff\_player2 + discount\_factor \* np.max(q\_table[state\_player2]) - q\_table[state\_player2, action\_player2])

# Q-learning algoritmasıyla oyunu eğitme

for episode in range(num\_episodes):

state\_player1 = np.random.randint(num\_states)

state\_player2 = np.random.randint(num\_states)

take\_step(state\_player1, state\_player2)

# Her 1000 episode'de bir sonuçları yazdırma

if (episode + 1) % 1000 == 0:

strategy\_player1 = np.argmax(q\_table, axis=1)

strategy\_player2 = np.argmax(q\_table, axis=0)

print(f"Episode: {episode + 1}")

print("Player 1 Strategy:", strategy\_player1)

print("Player 2 Strategy:", strategy\_player2)

print()

# Eğitim sonucunda Nash dengesini elde etmek için stratejileri belirleme

strategy\_player1 = np.argmax(q\_table, axis=1)

strategy\_player2 = np.argmax(q\_table, axis=0)

# Sonuçları yazdırma

print("Final Player 1 Strategy:", strategy\_player1)

print("Final Player 2 Strategy:", strategy\_player2)