**Virtual Initio Programming: WS25 Sample Answers**

**Question 1:** The program writes the contents of the dictionary to a file called “policy\_file”. Each line in this file consists of a dictionary key, followed by “::” followed by the value for that key in the dictionary.

**Exercise 1:** The key bit is the final few lines of this program – the rest is my solution to WS24.

import simclient.simrobot as initio

import time, random

initio.init()

actions = ['forward','backward','left','right']

def action\_reward(action\_list, default):

action\_rewards = {}

for i in range(0, 2):

for j in range(0, 2):

for k in (action\_list):

action\_rewards[((i, j), k)] = default

return action\_rewards

def execute\_action(action):

if (action == "forward"):

initio.forward(10)

elif (action == "backward"):

initio.reverse(10)

elif (action == "left"):

initio.spinLeft(10)

elif (action == "right"):

initio.spinRight(10)

else:

initio.stop()

time.sleep(3)

def best\_action(state):

max\_reward = 0

for act in actions:

if (reward\_dictionary[(state, act)] > max\_reward):

action = act

max\_reward = reward\_dictionary[(state, act)]

return action

reward\_dictionary = action\_reward(actions, 1)

rewards = {(1, 1):1, (1, 0):2, (0, 1):0, (0, 0):1}

epsilon = 1

epsilon\_reduce = 0.05

learning\_rate = 0.5

while (epsilon > 0):

explore = random.random()

state = (initio.irLeftLine(), initio.irRightLine())

if (explore < epsilon):

action = random.choice(actions)

print("Random Action: " + action)

else:

action = best\_action(state)

print("Best Action: " + action)

execute\_action(action)

reward = rewards[(initio.irLeftLine(), initio.irRightLine())]

reward\_dictionary[(state, action)] = reward\_dictionary[(state, action)] + (reward - reward\_dictionary[(state, action)])\*learning\_rate

if (reward == 2):

epsilon = epsilon - epsilon\_reduce

print("New epsilon: " + str(epsilon))

initio.stop()

f = open('policy\_file', 'w')

for key in reward\_dictionary:

f.write(str(key))

f.write('::')

f.write(str(reward\_dictionary[key]))

f.write('\n')

f.close()

**Exercise 2:**

f = open('policy\_file', 'r')

for line in f:

print(line)

**Question 2:**

s[0] returns ( and s[1] return 0 – these are the first and second characters in the string.

**Question 3:**

s[1:4] returns 0, 1 – these are the 2nd-5th characters in the string.

**Question 4:**

Python prints: ['(0', ' 1)'] – this is an array of two elements consisting of the part of the string before the comma and the part of the string after the comma.

**Question 5:** This reads in a file – assumed to be formatted as the output for the first example program in the worksheet. It first splits each line around :: into the key and the value from the dictionary. It then splits the key around the commas to extract strings that contain the values of the left and right line sensors and the action. It extracts the values of the left and right line sensors from these strings by knowing at which character they will occur. It then splits the action string around the ‘ to extract the string containing the action name. It then prints out the sensor values, the action name and the value string.

**Exercise 3:**

import simclient.simrobot as initio

import time, random

initio.init()

actions = ['forward','backward','left','right']

def action\_reward(action\_list, default):

action\_rewards = {}

for i in range(0, 2):

for j in range(0, 2):

for k in (action\_list):

action\_rewards[((i, j), k)] = default

return action\_rewards

def execute\_action(action):

if (action == "forward"):

initio.forward(10)

elif (action == "backward"):

initio.reverse(10)

elif (action == "left"):

initio.spinLeft(10)

elif (action == "right"):

initio.spinRight(10)

else:

initio.stop()

time.sleep(3)

def best\_action(state):

max\_reward = 0

for act in actions:

if (reward\_dictionary[(state, act)] > max\_reward):

action = act

max\_reward = reward\_dictionary[(state, act)]

return action

reward\_dictionary = action\_reward(actions, 1)

f = open('policy\_file', 'r')

for line in f:

[key, value] = line.split('::')

[a, b, c] = key.split(',')

irR = a[2]

irL = b[1]

[x, action, y] = c.split('\'')

state = (int(irR), int(irL))

reward\_dictionary[(state, action)] = float(value)

f.close()

while (initio.getDistance() > 50):

state = (initio.irLeftLine(), initio.irRightLine())

action = best\_action(state)

execute\_action(action)

initio.stop()



University of Liverpool, 2020

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).