

Lab-2 Vacuum Cleaning Agent (Reflex-based)

→ Initial Conditions:

Setting up a 2-room space with a 1x2

array

arr = ["None", "None"]

Setting up a randomized environment
~~for i in range(2)~~

~~arr[i] = random.choice("C", "D")~~

for i ← 0 to 2

arr[i] = random.choice("C", "D")

and p-sequence

Setting up a position for the vacuum cleaner,

pos = 0; pseq = []

→ Algorithm for cleaning the sequence of rooms def clean(pos, arr):

~~if pos pseq.append~~

while True: # Infinite loop

pseq.append(arr[pos], (pos, arr[pos]))

if arr[pos] == "D"

print("Cleaning room {}".format(pos)); arr[pos] = "C"

else arr[pos] == "C":

print("Room is clean")

Randomize environment again

~~for i in range(2)~~

~~arr[i] = random.choice("C", "D")~~

Increment position

pos = (pos + 1) % 2

arr[pos] = random.choice("C", "D")

pos = (pos + 1) % 2

→ Utility Function:

→ display():

```
def display(is arr, pseq):
    print("Percept Sequence: ", pseq)
def print("Current env: ", arr)
```

→ Sample Percept Sequence:

Initial env: pseq = []
arr = [{"C", "D"}]
pos = 0

Showing 5 Steps

S1: pseq = [(0, "C")]

~~arr = [{"C", "D"}] (0, "C"), (1, "D")~~

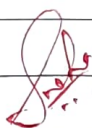
S2: pseq = [(0, "C"), (1, "D")]

Clear in this car

S3: ~~pseq~~ pseq = [(0, "C"), (1, "D"), (0, "D")]

S4: pseq = [(0, "C"), (1, "D"), (0, "D"), (1, "C")]

S5: pseq = [(0, "C"), (1, "D"), (0, "D"), (1, "C"), (0, "C")]



→

Program:

import random

class Cleaner:

def __init__(self):

self.env = [None, None]

for i in range(2):

self.env[i] = random.choice(["C", "D"])

self.pos = 0

self.pseq = []

self.clean()

def display(self):

print("Current percept seq: ", self.pseq)

print("Current env: ", self.env)

def clean(self):

while True:

self.pseq.append((self.pos,
self.env[self.pos]))

if self.env[self.pos] == "D":

print(f"Room {self.pos} is dirty, ^{cleaning...}")

self.env[self.pos] = "C"

else:

print("Room is clean...")

if self.pos == 1:

print("Moving left...")

else:

print("Moving right...")

self.display()

self.env[self.pos] =

random.choice(["C", "D"])

self.pos = (self.pos + 1) % 2

c = Cleaner()

Output:

Room 0 is dirty, cleaning...

Current percept seq: [(0, 'D')]

Current env: ['C', 'D']

Room 1 is dirty, cleaning...

Current percept seq: ~~[(0, 'D')]~~, [(0, 'D'), (1, 'D')]

Current env: ['D', 'C']

Room 0 is dirty, cleaning...

Current percept seq: [(0, 'D'), (1, 'D'), (0, 'D')]

Current env: ['C', 'C']

Room ~~1~~ is clean...

Moving left...

Current percept seq: [(0, 'D'), (1, 'D'), (0, 'D'), (1, 'C')]

Current env: ['D', 'C']

~~Suckoff B~~
~~11/10/24~~