- · Engineer: Shaun Pritchard
- Date: 1/27/2023
- · Task: Reflex,goal, and reflex model agents
- Artifical Intellgence CAP 6635
- 1 !pip install·messagebox

Looking in indexes: <a href="https://pypi.org/simple">https://us-python.pkg.dev/colab-wheels/public/simple/</a> Requirement already satisfied: messagebox in /usr/local/lib/python3.8/dist-packages (0.1.0)

```
1
    class Node:
 2
        def init (self, x, y, parent=None):
 3
            self.x = x
 4
            self.y = y
 5
            self.parent = parent
 6
 7
        def set x(self, x):
 8
            self.x = x
 9
10
        def set y(self, y):
            self.y = y
11
12
13
        def get_x(self):
14
            return self.x
15
16
        def get_y(self):
17
            return self.y
18
19
        def set_parent(self, parent):
20
            self.parent = parent
21
        def get parent(self):
22
            return self.parent
23
 1 import matplotlib.pyplot as plt
 2 import random
 3 # from Node import *
 4 from tkinter import messagebox
 5 from copy import copy, deepcopy
 6 import matplotlib.pyplot as plt
 7 import numpy as np
8 import random
```

## Model 1

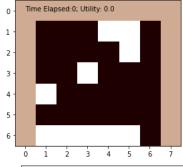
```
1 # CAP 6635 Artificial Intelligence; X. Zhu; 01/13/2022
 2 # model.py" is a reflex agent with a model to ensure agent walking through all locations
 3 # Code adopted from https://github.com/mawippel/python-vacuum. Changes are made to reflect agent moves following predef
 4 # Reflex Vacuum Cleaner Agent. Agent makes random move (-1 for each move, and +10 for clean a spot)
 6 # 0 -> clean
7 # 1 -> wall
 8 # 2 -> dirt
9 # The original matrix contains probablty values which will be used to generte the environment.
10 # if you want to make a spot to have dirt for sure, set the value as 1.0
11 # if you do NOT want to make a spot to have dirt, set the value as 0
12 matrix = [
13
      [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
14
      [1.0, 0.1, 0.1, 0.1, 0.4, 0.4, 0.1, 1.0],
15
      [1.0, 0.1, 0.1, 0.1, 0.6, 0.5, 0.1, 1.0],
16
      [1.0, 0.1, 0.4, 0.1, 0.1, 0.1, 0.1, 1.0],
17
      [1.0, 0.4, 0.6, 0.4, 0.1, 0.1, 0.1, 1.0],
      [1.0, 0.1, 0.4, 0.1, 0.1, 0.1, 0.1, 1.0],
```

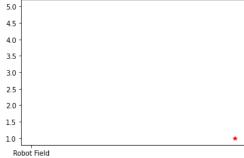
```
[1.0, 1.0, 1.0, 1.0, 1.0, 0.4, 1.0]
 1 # Actions Matrix -> represents the action for each position
 2 # Actions = up (0), down (1), left (2), right (3), clean(4), end (5)
 3 actionsMatrix = [
      [9, 9, 9, 9, 9, 9, 9],
 5
      [9, 1, 3, 1, 3, 1, 0, 9],
 6
      [9, 1, 0, 1, 0, 1, 0, 9],
 7
      [9, 1, 0, 1, 0, 1, 0, 9],
 8
      [9, 1, 0, 1, 0, 1, 1, 9],
9
      [9, 3, 0, 3, 0, 5, 0, 9],
10
      [9, 9, 9, 9, 9, 9, 9]
11 ]
12
 1 def renderMatrix(matrix,x,y,utility,timeElapsed):
      plt.text(0,0,"Time Elapsed:%d; Utility: %.1f"%(timeElapsed,utility))
      plt.imshow(matrix, 'pink')
 3
 4
      plt.show(block=False)
      plt.plot(y,x,'r:',linewidth=1)
 5
      plt.plot(y[len(y)-1], x[len(x)-1], '*r', 'Robot Field', 5)
 6
 7
      plt.pause(0.5)
 8
      plt.clf()
9
10 def createWorld(m):
      for mI in range(1, 7):
11
          for aI in range(1, 7):
12
13
              if (random.random()<m[mI][aI]):</pre>
14
                  m[mI][aI] = 2
15
              else:
16
                  m[mI][aI] = 0
17
      #renderMatrix(matrix)
18
19 def findNextAction(x, y):
20 return actionsMatrix[x][y]
21
22 # decides which action will be done
23 # Actions = up (0), down (1), left (2), right (3), clean(4)
24 def modelAgentRobot(x, y):
25 if (matrix[x][y] == 2): # if it's dirty, return the clean action
26
27  return findNextAction(x, y)
28
29 def checkDirtSpots(matrix):
30 x=len(matrix)
31 totalones=2*x+(x-2)*2
32   sum=np.sum(matrix)-totalones*2
33 return(sum)
1 def main():
 2 createWorld(matrix)
 3 print("Environment (beginning)\r\n")
 4 print('\n'.join(['\t'.join([str(cell) for cell in row]) for row in matrix]))
 5 # The robot always starts at matrix[1][1]
 6 currLine = 1
 7 \quad currCol = 1
 8 Lines=[]
9 Cols=[]
10 Lines.append(currLine)
11 Cols.append(currCol)
12 utility=0
13 timeElapsed=0
14 renderMatrix(matrix,Lines,Cols,utility,timeElapsed)
15  totalDirt=checkDirtSpots(matrix)
16 print(totalDirt)
   while True:
17
      action = modelAgentRobot(currLine, currCol)
18
      if (action == 0): # go up
```

```
print("up")
       currLine = currLine - 1 # remove 1 line
       utility=utility-1
23
    elif (action == 1): # go down
     print("down")
25
      currLine = currLine + 1
26
      utility=utility-1
27
    elif (action == 2): # go left
     print("left")
28
29
      currCol = currCol - 1
30
      utility=utility-1
31
    elif (action == 3): # go right
     print("right")
32
33
      currCol = currCol + 1
      utility=utility-1
34
35
    elif (action == 4): # clean
     print("clean")
36
37
      matrix[currLine][currCol] = 0
38
      utility=utility+10
39
    else:
     print("end")
40
41
      break
42
    Lines.append(currLine)
     Cols.append(currCol)
43
     timeElapsed=timeElapsed+1
44
45
     renderMatrix(matrix,Lines,Cols,utility,timeElapsed)
46
    print("Environment (ending): %f\r\n"%utility)
47
    print('\n'.join(['\t'.join([str(cell) for cell in row]) for row in matrix]))
48
49
50
51
52 if __name__ == "__main__":
53 main()
```

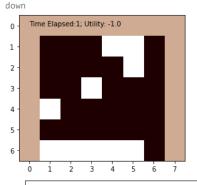
Environment (beginning)

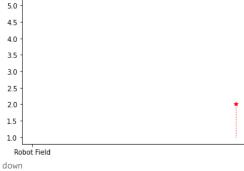
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	0	0	0	2	2	0	1.0
1.0	0	0	0	0	2	0	1.0
1.0	0	0	2	0	0	0	1.0
1.0	2	0	0	0	0	0	1.0
1.0	0	0	0	0	0	0	1.0
1.0	2	2	2	2	2	0	1.0





-8.0





0 - Time Elapsed:2; Utility: -2.0

1 - 2 - 3 - 4 - 5 - 6 - 6 - 6 - 7

```
1
```

## - GOAL

```
1
 1 # CAP 6635 Artificial Intelligence; X. Zhu; 01/13/2022
 3 # Code adopted from https://github.com/mawippel/python-vacuum. Changes are made to reflect agent moves following search
4 # Goal Based Vacuum Cleaner Agent. Agent repetitively searches closest dirt, and walks to clean the dirt. After that, s
 5 # dirt again. -1 for each move, and +10 for clean a dirt.
 6 # -1 -> clean
 7 # 0 -> wall
 8 # 1 -> dirt
9 # The original matrix contains probablty values which will be used to generte the environment.
10 # if you want to make a spot to have dirt for sure, set the value as 0.0
11 # if you do NOT want to make a spot to have dirt, set the value as -1
12 matrix = [
13
      [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
       [1.0, 0.1, 0.1, 0.1, 0.4, 0.4, 1.0],
14
15
       [1.0, 0.1, 0.1, 0.1, 0.6, 0.1, 1.0],
       [1.0, 0.1, 0.4, 0.1, 0.1, 0.1, 1.0],
16
      [1.0, 0.4, 0.6, 0.4, 0.1, 0.1, 1.0],
17
      [1.0, 0.1, 0.4, 0.1, 0.1, 0.1, 1.0],
18
       [1.0, 0.1, 0.4, 0.1, 0.1, 0.1, 1.0],
19
20
       [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
21 ]
22 presentationMatrix = [
      [1, 1, 1, 1, 1, 1, 1, 1],
24
       [1, 0, 0, 0, 0, 0, 0, 1],
25
       [1, 0, 0, 0, 0, 0, 0, 1],
26
      [1, 0, 0, 0, 0, 0, 0, 1],
27
      [1, 0, 0, 0, 0, 0, 0, 1],
28
      [1, 0, 0, 0, 0, 0, 0, 1],
29
      [1, 0, 0, 0, 0, 0, 0, 1],
30
       [1, 1, 1, 1, 1, 1, 1],
31 ]
33 # The robot always starts at matrix[1][1]
34 currLine = 1
35 \text{ currCol} = 1
36 \text{ stack} = [Node(1, 1)]
37 \text{ solution} = [Node(1, 1)]
38 process_map = []
39
40
41 def mapNotClean():
42
       for i in range(1, len(matrix) - 1):
43
           for j in range(1, len(matrix[i]) - 1):
44
               if (matrix[i][j] == 2):
45
                   return True
46
      return False
47
48
49 #def renderMatrix(matrix):
       plt.imshow(matrix, 'pink')
50 #
51 #
       plt.show(block=False)
       plt.plot(currCol, currLine, '*r', 'LineWidth', 5)
52 #
53 #
       plt.pause(0.5)
       plt.clf()
55 def renderMatrix(matrix,x,y,utility,timeElapsed):
      plt.text(0,0,"Time Elapsed:%d; Utility: %.1f"%(timeElapsed,utility))
56
57
       plt.imshow(matrix, 'pink')
58
      plt.show(block=False)
59
      plt.plot(y,x,'r:',linewidth=1)
60
      plt.plot(y[len(y)-1], x[len(x)-1], '*r', 'Robot Field', 5)
61
       plt.pause(0.5)
       plt.clf()
```

```
63
 64
 65
 66 def createWorld(m):
       for mI in range(1, 6):
            for aI in range(1, 6):
 69 #
                 if (mI == 1 and aI == 1):
 70 #
                     continue
 71 #
                number = random.randint(0, 3)
 72 #
                m[mI][aI] = 2 \text{ if number} == 1 \text{ else } 0
 73
                if (random.random()<m[mI][aI]):</pre>
 74
                    m[mI][aI] = 2
 75
                else:
 76
                    m[mI][aI] = 0
 77
       #renderMatrix(matrix)
 78
       global process_map
 79
       global presentationMatrix
 80
       process_map = deepcopy(matrix)
 81
       presentationMatrix = deepcopy(matrix)
 82
 83
 84 def hasPosition(x, y):
       if (matrix[x][y] == 1):
 86
           return False
 87
       return True
 88
 89
 90 def lookLeft(x, y, node):
 91
      if (hasPosition(x - 1, y)):
 92
           new_node = Node(x - 1, y, node)
 93
           if (process_map[x - 1][y] == 2):
 94
                return new_node
 95
           if (process_map[x - 1][y] != 4):
 96
                stack.append(new_node)
 97
                process_map[x - 1][y] = 4
 98
 99
100 def lookRight(x, y, node):
     if (hasPosition(x + 1, y)):
101
           new_node = Node(x + 1, y, node)
102
103
           if (process_map[x + 1][y] == 2):
104
               return new_node
105
           if (process_map[x + 1][y] != 4):
106
                stack.append(new_node)
107
                process_map[x + 1][y] = 4
108
109
110 def lookAbove(x, y, node):
111
      if (hasPosition(x, y - 1)):
           new_node = Node(x, y - 1, node)
112
           if (process_map[x][y - 1] == 2):
113
114
                return new_node
           if (process_map[x][y - 1] != 4):
115
116
                stack.append(new_node)
117
                process_map[x][y - 1] = 4
118
119
120 def lookDown(x, y, node):
121
       if (hasPosition(x, y + 1)):
122
            new_node = Node(x, y + 1, node)
123
            if (process_map[x][y + 1] == 2):
124
                return new_node
125
           if (process_map[x][y + 1] != 4):
126
                stack.append(new_node)
127
                process_map[x][y + 1] = 4
128
129
130 def discoverPath():
131
       while (len(stack) != 0):
132
            node = stack.pop(0)
```

```
x = node.get_x()
134
           y = node.get_y()
135
136
           auxNode = lookLeft(x, y, node)
137
           if (auxNode):
138
                return auxNode
139
140
           auxNode = lookAbove(x, y, node)
141
           if (auxNode):
                return auxNode
142
143
           auxNode = lookRight(x, y, node)
145
           if (auxNode):
146
                return auxNode
147
148
           auxNode = lookDown(x, y, node)
149
           if (auxNode):
150
                return auxNode
151
152
153 def main():
154
     global matrix
155
       global process_map
156
       global stack
157
       global currCol
158
       global currLine
159
       createWorld(matrix)
160
161
        print("Environment (beginning)\r\n")
162
       print('\n'.join(['\t'.join([str(cell) for cell in row]) for row in matrix]))
163
       # The robot always starts at matrix[1][1]
164
165
       currLine = 1
166
       currCol = 1
167
       utility=0
168
       Lines=[]
169
       Cols=[]
170
       Lines.append(currLine)
171
       Cols.append(currCol)
172
       timeElapsed=0
173
       renderMatrix(matrix,Lines,Cols,utility,timeElapsed)
174
175
       while (mapNotClean()):
176
           path = discoverPath()
177
           x = path.get_x()
178
           y = path.get_y()
179
180
           aux_list = []
           while (path.get_parent() is not None):
181
182
                process_map[path.get_x()][path.get_y()] = 3
183
                aux_list.append(path)
184
                path = path.get_parent()
185
           aux_list.reverse()
186
           solution.extend(aux_list)
187
188
           matrix[x][y] = 0
189
            stack = [Node(x, y)]
190
            process_map = deepcopy(matrix)
191
192
       for path in solution:
193
           currCol = path.get_y()
194
            currLine = path.get_x()
195
            Lines.append(currLine)
196
            Cols.append(currCol)
197
           timeElapsed=timeElapsed+1
            render {\tt Matrix} (presentation {\tt Matrix}, {\tt Lines}, {\tt Cols}, {\tt utility}, {\tt timeElapsed})
198
199
            if (presentationMatrix[currLine][currCol] == 2):
                presentationMatrix[currLine][currCol] = 0
200
201
                utility=utility+10
202
```

Environment (beginning)

1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	0	0	0	0	2	1.0
1.0	0	0	0	0	0	1.0
1.0	0	2	0	0	2	1.0
1.0	2	2	2	0	2	1.0
1.0	0	0	2	0	0	1.0
1.0	0.1	0.4	0.1	0.1	0.1	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0

