9/11/20

AI lab Test-1.

IDS algo.

Shavan Ram D.B.
1BM18CS091.

```
import numpy as np.
~~from PuzzleNode import *.~~
import copy.
import time.
from queue import PriorityQueue.
from itertools import count.
class PuzzleNode:
def __init__(self, init=None);
Self.goal = np.array([[-1,1,2],[3,4,0],[0,0,0]])
Self.node = self.goal.copy() if init is None else
init.copy()
self.io = 2.
self.jo = j.
self.parent = None.
def down(self):
 assert self.io > 0.
 io = self.io
 jo. = Self.jo.
self.node[io][jo], self.node[io-1][jo] = self.node[io self.io -= 1.
 def up(self):
 assert self.io < 2.
 io = self.io.
 jo = self.jo
 self.node[io][jo], self.node[io+1][jo] = self.node[io self.io += 1.
 def right(self):
 assert self.jo > 0.
 io = self.io.
 jo = self.jo.
 Self.node[io][jo] self.node[io][jo-1] = Self.node[io] self.jo -= 1
```


Shavan Ram D.B

Scanned with CamScanner

Shaevan Ram D.B.
1BM18CS097.

```
def left(self): assert self.jo < 2
io = self.io
jo = self.jo
self.node[io][jo], self.node[io][jo+1] = self.node[io][jo+1],
self.node[io][jo]
self.jo += 1
def getValidMoves(self):
validDir = []
if self.io > 0:
validDir.append('d')
if self.io < 2:
valid Dir.append('u')
if self.jo > 0:
validDir.append('r')
if self.jo < 2:
validDir.append('l')
return validDir
def doMove(self, moveChar):
if moveChar == 'd':
self.down()
if moveChar == 'u':
self.up()
if moveChar == 'r':
self.right()
if moveChar == 'l':
self.left()
def randomStep(self):
validDir = self.getValidMoves()
dirNum = len(validDir)
randomDir = validDir[np.random.randint(0, dirNum)]
```


ShaevanRam D.B.

Scanned with CamScanner

```
self.doMove(randomDir)
def shuffle(self, shuffleTime=20):
for i in range(shuffleTime):
self.randomStep()
def isGoal(self):
return (self.node == self.goal).all()
def numOfWrong(self):
return 9 - np.sum(self.node == self.goal)
def manhattanDist(self):
dist = 0
for i in range(3):
for j in range(3):
num = self.node[i][j]
if num != 0:
iGoal = (num-1) // 3
jGoal = (num-1) % 3
dist += abs(iGoal - i) + abs(jGoal - j)
return dist
def show(self):
print(self.node)
# print solution trace
test = PuzzleNode()
test.shuffle()
test.show()
step, trace = iterativeDeepingSearch(test)
print(step)
while len(trace) != 0:
n = trace.pop()
print(n)
```


Shavan Ram D.B

Scanned with CamScanner