

Balaji Senthilkumar

CS570-Advanced Intelligence Systems

Program #1: Fred Flintstone problem-solving: part 1

January 29, 2021

Output of running project1_test.py:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
balajisenthilkumar@Balajis-MacBook-Air program 1 - Boggle % python3 interpreter.py project1_test1.py

>>> from project1_funs import *

>>>

>>> myBoard = loadBoard("board.txt")

>>> printBoard(myBoard)
J O P Y
C M P V
X F E G
P G V U

>>> possibleMoves((3,3),myBoard)
{(2, 3), (3, 2), (2, 2)}
[(2, 2), (2, 3), (3, 2)]

>>> possibleMoves((2,1),myBoard)
{(1, 2), (3, 1), (1, 1), (2, 0), (3, 0), (2, 2), (1, 0), (3, 2)}
[(3, 1), (2, 2), (3, 2), (3, 0), (1, 0), (1, 2), (1, 1), (2, 0)]

>>> legalMoves( possibleMoves((1,2), myBoard), ( (1,0),(2,0),(2,1),(2,2) ))
{(0, 1), (2, 1), (1, 1), (0, 3), (2, 3), (0, 2), (2, 2), (1, 3)}
[(2, 2), (1, 3), (2, 3), (2, 1), (0, 1), (0, 3), (0, 2), (1, 1)]

>>> legalMoves( possibleMoves((2,2), myBoard), ( (1,1),(1,2),(1,3),(2,3),(3,2) ) )
{(1, 2), (2, 1), (3, 1), (1, 1), (2, 3), (3, 3), (3, 2), (1, 3)}
[(3, 2), (2, 3), (3, 3), (3, 1), (1, 1), (1, 3), (1, 2), (2, 1)]

>>> myDict = frozenset(word.strip() for word in open("twl06.txt"))

>>> examineState(myBoard,(0,3),( (1,1), (0,1),(0,2) ),myDict)
('mopy', 'Yes')

>>> examineState(myBoard,(0,0),( (3,3), (2,2), (1,1) ),myDict)
('uemj', 'No')

>>> examineState(myBoard,(3,3),( (2,2),(2,1),(2,0),(3,0),(3,1),(3,2) ),myDict)
('efxpgvu', 'No')
```

project1_funs.py (source code)

- loadBoard(), and printBoard()

```

project1_funs.py X project1_test1.py interpreter.py
project1_funs.py > possibleMoves
1  """
2      Programming Assignment 1 - Part1
3      Submitted by Balaji Senthilkumar
4  """
5  #The funtion to load the board (.txt to 2Dlist/array)
6  def loadBoard(board):
7      inputFile = open(board, 'r')
8      myBoard = []
9      for row in inputFile:
10         myBoard.append(row.split())
11     return myBoard
12
13
14 #THE FUNCTION TO PRINT 'myBoard'
15 def printBoard(myBoard):
16     for row in myBoard:
17         print (" ".join(map(str,row)))
18     # print('***'+myBoard[2][3])
19
20

```

- possibleMoves()

```

20
21 def possibleMoves(currentPosition,myBoard):
22     #So here I have used a graph based approach, for figuring out the solution,
23     # using x_co_ordinates and y_co_ordinates as the positional arguments.
24
25     x_co_ordinate,y_co_ordinate = currentPosition #spreading operation
26
27     possibleMovesArray=[]
28
29     limit=len(myBoard) # ensuring that the program does not cause an outOfBounds or any negative array positioning
30
31     if(x_co_ordinate+1<limit and y_co_ordinate<limit): #left to right
32         possibleMovesArray.append((x_co_ordinate+1,y_co_ordinate))
33
34     if(x_co_ordinate<limit and y_co_ordinate+1<limit): #Top-Bottom
35         possibleMovesArray.append((x_co_ordinate,y_co_ordinate+1))
36
37     if(x_co_ordinate+1<limit and y_co_ordinate+1<limit): #diagonal 1
38         possibleMovesArray.append((x_co_ordinate+1,y_co_ordinate+1))
39
40     if(x_co_ordinate+1<limit and ((y_co_ordinate-1<limit)and y_co_ordinate-1>=0)): #diagonal 2
41         possibleMovesArray.append((x_co_ordinate+1,y_co_ordinate-1))
42
43     if(((x_co_ordinate-1<limit)and x_co_ordinate-1>=0) and ((y_co_ordinate-1<limit)and y_co_ordinate-1>=0)): #diagonal 3
44         possibleMovesArray.append((x_co_ordinate-1,y_co_ordinate-1))
45
46     if(((x_co_ordinate-1<limit)and x_co_ordinate-1>=0) and (y_co_ordinate+1<limit)): #diagonal 4
47         possibleMovesArray.append((x_co_ordinate-1,y_co_ordinate+1))
48
49     if(((x_co_ordinate-1<limit)and x_co_ordinate-1>=0) and y_co_ordinate<limit): #right to left
50         possibleMovesArray.append((x_co_ordinate-1,y_co_ordinate))
51
52     if(x_co_ordinate<limit and ((y_co_ordinate-1<limit)and y_co_ordinate-1>=0)): #Bottom Up
53         possibleMovesArray.append((x_co_ordinate,y_co_ordinate-1))
54     #print(set(possibleMovesArray))
55     return(possibleMovesArray)
56
57

```

- legalMoves() , examineState()

```

56
57
58
59 def legalMoves(possibleMovesArg,pathArg):
60     legalMovesArray=[]           #creating a list to store the legal moving positions(co_ordinates)
61     for pos in possibleMovesArg:
62         if pos not in legalMovesArray: #checking for non-repetion of positions, to comply the rules of the game
63             legalMovesArray.append(pos)
64     return legalMovesArray
65
66
67 def examineState(myBoard,currentPosition,path,myDict):
68     wordList = []
69     for i in path:
70         x_co_ordinate, y_co_ordinate = i #spreading x_co_ordinate and y_co_ordinate out of the path
71         wordList.append(myBoard[x_co_ordinate][y_co_ordinate])
72     x_co_ordinate, y_co_ordinate = currentPosition
73     wordList.append(myBoard[x_co_ordinate][y_co_ordinate])
74     finalList = ''.join([str(i) for i in wordList])
75     # print(finalList)
76     if finalList.lower() in myDict: #Checking whether the word is in the given dictionary
77         outputTuple=(finalList.lower(),'Yes')
78         print(outputTuple)
79     else:
80         outputTuple=(finalList.lower(),'No')
81         print(outputTuple)
82

```