Proximity Report

Ονοματεπώνυμο : Χριστόπουλος Πέτρος,

Ημ/νια : May 2024,

AEM: 16678,

Διεύθυνση ιδρυματικού mail: christopou@math.auth.gr,

Διεύθυνση Gmail: christopoulospetros16@gmail.com

Παρακάτω αναλύεται ο κώδικας που αφορά στο Proximity game:

Ορισμός της Cell Class

Cell Class και ζητούμενα methods αυτής:

```
import numpy as np
from hexalattice.hexalattice import *

class Cell:
    def __init__(self, value, owner):
        self.setValue(value)
        self.setOwner(owner)

    def setValue(self, value):
        if 0 <= value <= 20:
            self.value = value
        else:
            raise ValueError('Η τιμή(value) ενός κελιού πρέπει να είναι μεταξύ 0
και 20.')

    def getValue(self):
        return "Το κελί έχει λάβει τιμή: " + str(self.value)

    def setOwner(self, owner):
        if owner in (0, 1, 2, 3):
            self.owner = owner
        else:
            raise ValueError('Ο ιδιοκτήτης(owner) ενός κελιού μπορεί να πάρει μόνο
τις τιμές:\n0(χωρίς ιδιοκτήτη), 1(ιδιοκτήτης κόκκινος), 2(ιδιοκτήτης πράσινος),
3(υποψήφιο προς κατάληψη).')

    def getOwner(self):
        return self.owner


    def __repr__(self):
```

```
        return f"Cell({self.value}, {self.owner})"
```

Σχόλια:

1. Στο τέλευταίο method γίνεται χρήση formatted string για το επιθυμητό οπτικό αποτέλεσμα, όταν προσπαθώ να κάνω εκτυπώσω στην συέχεια το temp_board. source: https://stackoverflow.com/questions/1984162/purpose-of-repr-method

Παράδειγμα: Χρήση της κλάσης Cell

Στις εντολές που ακολουθούν με αντίστοιχη σειρά:

- Ορίζουμε pid (player_id) να είναι -> 1: Κόκκινος
- Θέτουμε τις διαστάσεις του ταμπλό να είναι (X=10) x (Y=8) = 80 cells
- Εκτυπώνουμε το pid
- Εκτυπώνουμε το όνομα του παίκτη

```
Proximity02.setPid(Proximity02, 1)

Proximity02.setBoardSize(Proximity02, 10, 8)

print(Proximity02.getPid(Proximity02))

print(Proximity02.getPlayerName())
```

Ορισμός της Proximity Class

Proximity02 Class και ζητούμενα methods αυτής:

```
class Proximity02:
    def __init__(self, pid, length_X, length_Y, board):
        self.setPid(pid)
        self.setBoardSize(length_X, length_Y)
        self.board = board

    def setPid(self, pid):
        if pid == 1 or pid == 2:
            self.pid = pid
        else:
            raise ValueError('Οι κωδικοί-αριθμοί των παικτών πρέπει να είναι 1:
ΚΟΚΚΙΝΟΣ, 2: ΠΡΑΣΙΝΟΣ')

    def getPid(self):
        player = ''
        if self.pid == 1:
            player = 'RED'
        elif self.pid == 2:
```

```
            player = 'GREEN'
        return "Ο παίκτης σας είναι ο " + str(self.pid) + ":" + player

    def setBoardSize(self, length_X, length_Y):
        if length_X > 0 and length_Y > 0:
            self.length_X = length_X
            self.length_Y = length_Y
        else:
            raise ValueError('Οι διαστάσεις του ταμπλό length_X και length_Y
πρέπει να είναι θετικές!')

    def getPlayerName():
        return 'Name of player: Christopoulos'

    def findNeighbours(self, board):
        temp_board = board.copy()
        length_X, length_Y = self.length_X, self.length_Y

        for i in range(len(board)):
            if board[i].getOwner() in (1, 2):  # Αν το κελί στην i-οστή θέση του
board είναι κατειλημμένο
                neighbours = self.getNeighbours(i, length_X, length_Y)
                for index in neighbours:
                    if board[index].getOwner() == 0:
                        temp_board[index] = Cell(0, 3)  # Θέτω το index στοιχείο
ως πιθανό προς κατάληψη

        return temp_board

    def getNeighbours(self, index, length_X, length_Y):
        neighbours = []
        x = index % length_X    #Ο αριθμός(δείκτης) της στήλης του κελιού του
οποίου τους neighbours ψάχνουμε. Υπολογίζεται με την βοήθεια του τελεστή % που
δίνει το υπόλοιπο διαίρεσης
        y = index // length_X   #Ο αριθμός της γραμμής στην οποία βρισκεται το
κελί που αναφέρω παραπάνω

        #possible neighbours sets
        posNeighboursSet = [(-1, 0), (1, 0), (0, -1), (0, 1), (-1, 1), (1, -1)] if
y % 2 == 0 else [(-1, 0), (1, 0), (0, -1), (0, 1), (-1, -1), (1, 1)]

        for dx, dy in posNeighboursSet:
            nx = x + dx # nx = neighbours of x ( = 'οριζόντιοι' neighbours)
            ny = y + dy # ny = neighbours of y ( = 'κατακόρυφοι' neighbours)
            if 0 <= nx < length_X and 0 <= ny < length_Y:
                neighbours.append(ny * length_X + nx)

        return neighbours

    def findMyNeighbors(self, tile_position):
        return self.getNeighbours(tile_position, self.length_X, self.length_Y)
```

Παράδειγμα χρήσης του παραπάνω κώδικα:

```
length_X = 10
length_Y = 8
board = [Cell(0, 0) for _ in range(length_X * length_Y)] # το board στην αρχη του
παιχνιδιού
board[0] = Cell(15, 1) #player:1 ->RED τοποθετεί το κελί με value=15 στη θέση:0
του board
board[79] = Cell(8, 2) #player:2 ->GREEN τοποθετεί το κελί με value=8 στη θέση:79
του board

player_name = Proximity02.getPlayerName()
print(player_name)
proximity_game = Proximity02(pid=2, length_X=length_X, length_Y=length_Y,
board=board)
temp_board = proximity_game.findNeighbours(board)
print('temp_board = ',temp_board)
print('Neighbours list: ', proximity_game.findMyNeighbors(2))
```

Σχόλια:

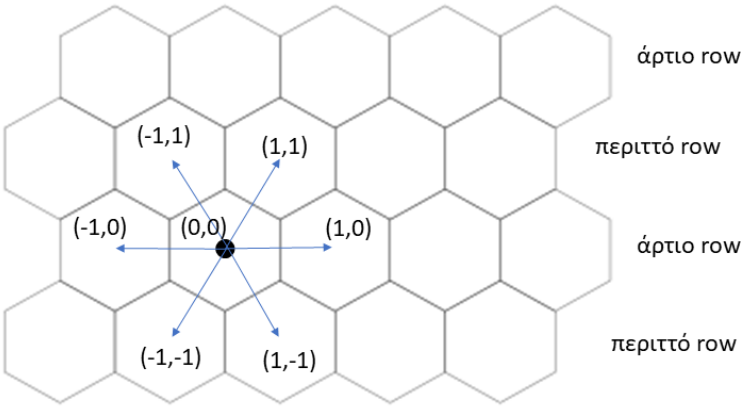
1. για κάθε κίνηση θα πρέπει να προσθέτω αντίστοιχα όπως στα παραπάνω:

   ```
   board[...] = Cell(..., ...)
   ```

2. method getNeighbours(): Δίνουμε 'διαισθητικά τη δομή ενός πίνακα' στο board (δηλαδή με γραμμές και στήλες). Προσπαθούμε να βρούμε το μοτίβο που κρύβεται πίσω από την εύρεση των neighbours. Έτσι καταλήγουμε στον παραπάνω ορισμό του method.

3. Στον ορισμό των posNeighboursSet βλέπουμε ότι τα y αναπαριστούν τις γραμμές του board, και ορίζουμε τις δυνατές μετατοπίσεις που μπορούν να πραγματοποιηθούν όταν, βρισκόμενοι ήδη σε κελιά, πρέπει να καταλήξουμε επίσης σε κελιά (κινούμενοι στα πλαίσια ενός συστήματος συντεταγμένων) Για παράδειγμα: αν index = 0,...,9 τότε το y = 0, αντίστοιχα αν index = 10,...,19 --> y = 1, αν index = 20,...,29 --> y = 2 κοκ. Βλέπουμε λοιπόν ότι τα y αναπαριστούν τις γραμμές (τα rows) του board και επίσης ότι υπάρχει 'διαμερισμός' κατηγοριών ως προς το ποιό σύνολο κινήσεων (για την εύρεση των neighbours) επιτρεπει το πλαίσιο στο οποίο αναφερόμαστε. Δηλαδή αν row = 0,2,4,6 υπάρχουν συγκεκριμένες μετατοπίσεις που μπορούν να πραγματοποιηθούν για την αναζήτηση γείτονα.Αντίστοιχα αν row = 1,3,5,7 (και στην προκειμένη περίπτωση είναι total_rows = 8 με αρίθμηση από 0 εώς 7). Παρακάτω παρουσιάζεται σχηματικά η έννοια της μετατόπισης για οποία έκανα λόγο:

Εικόνα: Δυνατοί γείτονες του κελιού με θέση την (0,0)



4. Η method findMyNeighbours() ορίστηκε ουσιαστικά, όταν και όρισα (στην προσπάθειά μου να ορισω την findNeighbours()) την getNeighbours(). Και άρα για την δημιουργά της απλώς καλώ με τον κατάλληλο τρόπο την getNeighbours().