***Εργασία Προχωρημένα Θέματα***

***Η/Υ***

Συμμετέχοντες: Κωνσταντίνος Παπάζογλου 16731

**Εισαγωγή**

Στην εργασία χρησιμοποιήθηκαν οι ζητούμενες μέθοδοι. Δημιουργήθηκαν 2 κλάσεις, η πρώτη ονομαζόμενη Cell και η δεύτερη Proximity25. Και οι δύο είναι αναγκαίες για τη λειτουργία του αλγορίθμου αφού η Cell δίνει χαρακτηριστικά στα κελιά του ταμπλό μας. Η κλάση Proximity25 περιέχει τη παραγωγή του ταμπλό αλλά και τον τρόπο που αγωνίζεται ο παίκτης μας. Στο συγκεκριμένο παιχνίδι, λόγω κανόνων είναι πολύ σημαντική η παρακολούθηση των γειτονικών κελιών. Αυτό αποτελούσε ένα από τα βασικότερα ζητούμενα που καλούμασταν να αντιμετωπίσουμε. Γενικά, οι κανόνες του παιχνιδιού είναι:

1. Κάθε παίκτης τοποθετεί μία τιμή στη σειρά του σε ένα κελί
2. Κάθε γειτονικό κελί με μικρότερη τιμή από το δικό μας υπό ιδιοκτησία του αντιπάλου, το μετατρέπουμε σε δικό μας και το σκορ μας αυξάνει αναλόγως.
3. Κάθε γειτονικό κελί σε αυτό που τοποθετούμε τη νέα τιμή και είναι στη κατοχή μας, αυξάνει τη τιμή του κατά 1.

Νικητής ανακηρύσσεται ο παίκτης με το μεγαλύτερο σκορ όταν έχουν καλυφθεί όλα τα κελιά στο ταμπλό. Επίσης, οι τιμές που δίνονται για να τοποθετηθούν από τους παίκτες είναι μεταξύ του 1 και του 20. Οι εντολές που χρησιμοποιήθηκαν ήταν ακριβώς οι ζητούμενες που έχουν αναλυθεί στην εργασία ενώ χρησιμοποιήσαμε και ως ομάδα κάποιες βοηθητικές. Θεωρούσαμε ευκολότερο, να αντιλαμβανόμαστε το ταμπλό ως πίνακα και να συγκρίνουμε γραμμές και στήλες αντί να στηριζόμαστε στη χρήση των index.

**Extra Μέθοδοι που χρησιμοποιήθηκαν**

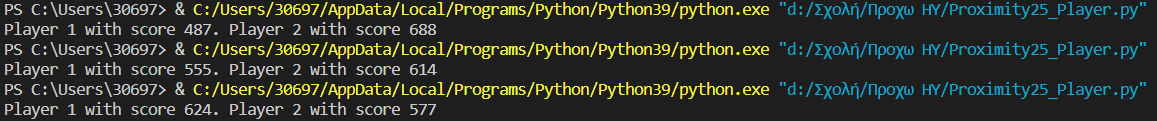
* Συνάρτηση wrap : Χρησιμοποιήθηκε με τη προοπτική να εμφανίζει index των κελιών, όταν δίνονται γραμμή και στήλη
* Συνάρτηση unwrap : Η δημιουργία της έγινε για το αντίθετο της wrap. Ουσιαστικά, επιστρέφει τη γραμμή και τη στήλη όταν δίνεται το index. Αυτές οι δύο βοηθήσαν στη διαδικασία εύρεσης γειτόνων.
* Συνάρτηση getScore : Αυτή μας επιστρέφει τη διαφορά του σκορ, ούτως ώστε να ξέρουμε αν προηγούμαστε ή όχι στο παιχνίδι.
* Συνάρτηση applyRules : Βασικό πρόβλημα στο παιχνίδι ήταν ότι δε ζητήθηκε μέθοδος που να εφαρμόζει τους κανόνες. Για αυτό φτιάξαμε αυτή την εξτρά μέθοδο, σύμφωνα με την οποία ο παίκτης υπακούει στους κανόνες , αφού καλείται μέσα σε αυτή η FindMyNeighbours για έλεγχο των γειτόνων και κάνει τις απαραίτητες αλλαγές στο ταμπλό μας και βοηθά και στην applyChanges αφού τοποθετεί τη τιμή στο κελί.
* Συνάρτηση isgridempty : Δέχεται το ταμπλό και ελέγχει αν είναι κενό σε περίπτωση που παίζουμε πρώτοι.

**Μέθοδος placeTile() :**

Η στρατηγική, σύμφωνα με την οποία ο παίκτης μας επιλέγει το κελί είναι πολύ απλή. Μέσα στη συνάρτηση καλούμε την FindNeighbours για να έχουμε τους γείτονες ενώ καλούμε και τη βοηθητική getScore για να αποθηκεύσουμε το υπάρχων σκορ. Στην if ελέγχουμε αν η κατάκτηση ενός από τα μαρκαρισμένα κελιά από τους γείτονες θα μας αυξήσει το σκορ. Αν αυτή η κίνηση δεν μας συμφέρει, γίνεται έλεγχος καλώντας την isgridempty σε περίπτωση που παίζουμε πρώτοι και τοποθετούμε τη τιμή στο κεντρικό κελί. Αν δεν ισχύει τίποτα από τα παραπάνω, απλά επιλέγουμε ένα τυχαίο κενό κελί. Το move ουσιαστικά είναι το i κελί που επιλέγεται από το grid και το ορίζουμε ως -1 για να είναι εκτός του ταμπλό αρχικά ούτως ώστε να εξεταστούν όλα τα κελιά.

**Κυρίως πρόγραμμα που αποδεικνύει την ορθή λειτουργία.**

Στο κυρίως πρόγραμμα υπάρχει και έλεγχος για το αν το ταμπλό είναι γεμάτο ούτως ώστε να τελειώσει το παιχνίδι και να αποδοθεί το τελικό σκορ. Η getScore σε αυτό το σημείο έχει άλλη δομή αλλά δεν παίζει ρόλο αφού είναι εκτός των κλάσεων. Παρακάτω παρατίθενται αποτελέσματα από διάφορες παρτίδες του παίκτη μας εναντίον τον εαυτό του.



Παρατηρούμε διαφορετικά αποτελέσματα αφού σε κάθε παρτίδα οι επιλογές είναι διαφορετικές για κάθε παίκτη αλλά βασίζονται και οι δύο στη placeTile.