ΤΕΧΝΗΤΉ ΝΟΗΜΟΣΎΝΗ

1Η ΕΡΓΑΣΊΑ

- ΧΑΣΑΠΗΣ ΝΙΚΟΛΑΟΣ, Α.Μ: 3170178
  ΚΥΡΑΓΙΑΝΝΗΣ ΠΕΤΡΟΣ, Α.Μ: 3170083
  ΜΠΟΥΜΠΑΛΗΣ ΠΕΤΡΟΣ-ΣΤΥΛΛΙΑΝΟΣ, Α.Μ: 3170119

1. Reversi

Στην 1η εργασία της ***Τεχνητής Νοημοσύνης*** αναπτύχθηκε ένα πρόγραμμα στο οποίο ο χρήστης μπορεί να παίξει Reversi απέναντι στον υπολογιστή. Το πρόγραμμα υλοποιήθηκε σε Python με τα ακόλουθα 3 αρχεία

1.) State.py
2.) SpaceSearcher.py
3.) Main.py

Το πρόγραμμα χρησιμοποιεί τον αλγόριθμο minimax με a-b pruning. Κατά την εκτέλεση ο χρήστης μπορεί να εισάγει το βάθος (depth) μέχρι το οποίο θα ψάχνει ο αλγόριθμος και στη συνέχεια να επιλέξει αν θέλει να παίξει πρώτος. Ο παίχτης που παίζει πρώτος παίρνει πάντα τα μαύρα πούλια.

2. Κώδικας Προγράμματος

2.1 State.py

Η ***State.py*** διατηρεί την κατάσταση του board δηλαδή τον πίνακα με τα πούλια και το score του board. Ο κατασκευαστής αρχικοποιεί την κατάσταση του board με τα πούλια στις σωστές θέσεις. Η μέθοδος ***isValidMove()*** δέχεται σαν όρισμα τη θέση (row και column) που θα γίνει η κίνηση και ελέγχει αν αυτή είναι εφικτή. Συγκεκριμένα, από τη θέση που θα μπει το νέο πούλι, κοιτάζουμε προς όλες τις κατευθύνσεις (διαγώνια, πάνω, κάτω, αριστερά, δεξιά) για αντίπαλα πούλια. Μέσα σε ένα loop όσο βρίσκουμε αντίπαλα πούλια τα κρατάμε σε μια προσωρινή δομή, coins. Αν στο loop βρούμε δικό μας πούλι, τότε βάζουμε τη δομή coins στη δομή flip, καθώς αυτό σημαίνει ότι η κίνηση είναι έγκυρη για την κατεύθυνση εκείνη. Επαναλαμβάνουμε τη μέθοδο για όλες τις κατευθύνσεις και γεμίζουμε τη λίστα flip. Αν στο τέλος η flip είναι κενή τότε η ***isValidMove()*** επιστρέφει None, αλλιώς κάνει return τη flip. Η μέθοδος ***makeMove()*** καλεί την ***isValidMove()*** και αν αυτή επιστρέψει στοιχεία από τη flip σημαίνει ότι η κίνηση είναι έγκυρη και αλλάζει τα πούλια για όλα τα coins της flip με το σωστό χρώμα. Η μέθοδος ***isFinal()*** ελέγχει αν ο παίχτης που παίζει έχει άλλες πιθανές κινήσεις, κοιτώντας όλα τα κενά κελιά. Στη συνέχεια έχουμε την ευρετική μας συνάρτηση η οποία βαθμολογεί με βάση τρία κριτήρια: 1) ποιος έχει τα περισσότερα πούλια, 2) ποιος έχει τις περισσότερες διαθέσιμες κινήσεις και 3) ποιος έχει πιάσει τις περισσότερες γωνίες στο board.

2.2 SpaceSearcher.py

Η ***SpaceSearcher.py*** υλοποιεί τον αλγόριθμο minimax με a-b pruning. Ο κατασκευαστής δέχεται ως ορίσματα τα a και b καθώς και το depth το οποίο έδωσε ο χρήστης στην αρχή του προγράμματος. Ο αλγόριθμος τρέχει μέχρι να εξαντληθεί το βάθος ή αν η κατάσταση που φτάσουμε είναι τελική (βλέπε ***isFinal()***). Στην περίπτωση του Reversi ο παίχτης με τα μαύρα πούλια είναι ο maximizer, ενώ ο παίχτης με τα άσπρα ο minimizer. Αυτό σημαίνει ότι αν ξεκινήσει ο χρήστης πρώτος το δέντρο θα ξεκινάει με ρίζα την κατάσταση μετά την κίνηση του χρήστη, δηλαδή με minimizer. Αν από την άλλη ξεκινήσει ο υπολογιστής πρώτος έχουμε maximizer στη ρίζα του δέντρου.

2.3 Main.py

Η ***Main.py*** αρχικοποιεί το board και ζητάει από τον χρήστη για το μέγιστο βάθος και αν επιθυμεί να παίξει πρώτος. Με βάση την απάντηση του χρήστη αρχικοποιούνται οι ανάλογες μεταβλητές για σειρά, πάσο, ανάθεση χρώματος. Το παιχνίδι τρέχει μέσα σε ένα loop και ξεκινάει ανάλογα με το τι επέλεξε ο χρήστης. Όταν παίζει ο χρήστης του ζητείται να εισάγει τις συντεταγμένες που θέλει να κάνει κίνηση και η ***makeMove()*** ελέγχει αν γίνεται και την πραγματοποιεί. Αν όχι τότε ζητάει νέες συντεταγμένες. Όταν παίζει ο CPU τρέχουμε ένα loop όπου καλούμε τον minimax για όλες τις διαθέσιμες κινήσεις και κρατάμε αυτή με το καλύτερο score, δηλαδή το μικρότερο όταν ο CPU είναι με τα άσπρα πούλια (minimizer) και το μεγαλύτερο όταν παίζει με τα μαύρα (maximizer). Έπειτα το πρόγραμμα καλεί την ***makeMove()*** για τη θέση αυτή. Το game τερματίζει όταν και οι δύο έχουν πάει «πάσο» δηλαδή όταν κανείς δεν έχει διαθέσιμη κίνηση. Στο τέλος εμφανίζεται το τελικό score καθώς και ένα μήνυμα για το ποιος νίκησε.