Οικονομικό Πανεπιστήμιο Αθηνών –
Τμήμα Πληροφορικής

Τεχνητή Νοημοσύνη

Εργασία πρώτη

Μαριάνθη Μηνδρινού 3150110
Μιχαήλ Ρούσσος 3150148
Χρήστος Τασιόπουλος 3150170

Ακολουθεί περιληπτική περιγραφή των κλάσεων ,των μεθόδων και της λογικής της πρώτης εργασίας του μαθήματος «Τεχνητή Νοημοσύνη».

- **Κλάση Evaluation**: Η κλάση αυτή χρησημοποιείται για την υλοποίηση των συναρτήσεων αξιολόγησης καταστάσεων . Η κλάση περιέχει ένα κατασκευαστή με το αντικείμενο State που δέχεται προς αξιολόγηση.
  Ακολουθεί περιληπτική περιγραφή των λειτουργιών των μεθόδων της κλάσης.
  - **evaluatePositions:** Σε αυτή την μέθοδο υπόλογίζουμε το άθροισμα των αξιών των θέσεων των πουλιών,προσθέτοντας την αξία των δικών μας και αφαιρώντας του αντιπάλου.
  - **evaluatePercentage:** Σε αυτήν την μέθοδο αξιολογούμε το ποσοστό των πουλιών μας σε σχέση με των αντιπάλων μας στον πίνακα, μετρώντας την διαφορά του πλήθους των δικών από το πλήθος του αντιπάλουμας προς το σύνολο.
  - **Complete_evaluation:** Πρόκειται για την τελική αξιολόγηση, η οποία προκύπτει από το γινόμενο των δυο προηγούμενων (δηλαδή γραμμικός συνδυασμός τους) πολλαπλασιασμένες επί το βάρος της καθεμίας.

- **Κλάση MiniMax_DFS:** Η κλάση αυτή υλοποιεί τον αλγόριθμο MiniMax με DFS.Η κλάση αυτή περιέχει τις εξής παρακάτω μεθόδους:
  - **creatingDFS:** Η μέθοδος αυτή καλεί την μέθοδο creatingDFSrecursive.
  - **creatingDFSrecursive:** Η μέθοδος αυτή υλοποιει αναδρομικά την λογική του αλγορίθμου MiniMax με πριόνισμα.
  - **min , max:** Οι μέθοδοι αυτές υπολογίζουν το σκορ του κάθε κόμβου ανάλογα αν είναι min ή max.

- **Κλάση Main:** Η κύρια κλάση του πρόγραμματος μας. Περιλαμβάνει τα μυνήματα επικονωνίας με το χρήστη,τις απαραίτητες εισόδους από τον χρήστη καθώς και τους απαραίτητους ελέγχους για την σωστή εισαγωγή των εισόδων.(Σε περίπτωση που ο τύπος δεδομένων που θα δοθεί σαν είσοδος δεν ταιριάζει με τον επιθυμητό τύπο, εμφανίζεται ανάλογο μήνυμα και το πρόγραμμα τερματίζει.) Αρχικά ελέγχουμε ποιος θα κάνει την πρώτη κίνηση και στην συνέχεια μαθαίνουμε το επιθυμητό βάθος αναζήτησης. Στην συνέχεια καλούμε την μέθοδο checkRules ,που κάνει τις απαρίτητες αλλαγές.Έπειτα αν παίζει πρώτος ο παίκτης, του ζητούμε να μας δώσει την θέση που θέλει να εισάγει το πούλι του, ρωτώντας τον για την γραμμή και την στήλη της θέσης(εμφανίζοντας ανάλογο μήνυμα για λανθασμένες εισαγωγές),και αν η θέση που εισήχθη ήταν δυνατή τότε εισάγουμε το πούλι στην αντίστοιχη θέση του πίνακα,αλλάζοντας και τα απαραίτητα πούλια του αντιπάλου. Διαφορετικά αν παίζει πρώτα ο υπολογιστής,κάλουμε την creatingDFS και κάνουμε τις απαραίτητες αλλαγές,σύμφωνα με την λογική του MiniMax, και στην συνέχεια ρωτάμε τον παίκτη για την δική του κίνηση. Τέλος εμφανίζουμε τα ανάλογα μηνύματα ανάλογα το αποτέλεσμα και ρωτάμε τον πάικτη αν θέλει να ξαναπαίξει ή όχι.

- **Κλάση State:** Στην αρχή δηλώνουμε τα πεδία της κλάσης:
    - **reversi_table**: ένας πίνακας χαρακτήρων που ουσιαστικά είναι το ταμπλό του παιχνιδιού σε κάθε κατάσταση
    - **father**: ο κόμβος από τον οποίο προέκυψε ο κόμβος στον οποίο είμαστε
    - **children:** μία λίστα με τις καταστάσεις που προκύπτουν από αυτή που είμαστε τώρα
    - **possible_columns, possible_rows**: δύο λίστες που περιέχουν τις πιθανές τοποθεσίες των πουλιών στις οποίες θα μπουν τα πούλια για να σχηματιστούν οι καταστάσεις των παιδιών
    - **current_depth**: το βάθος στο οποίο βρισκόμαστε
    - **player**: το χρώμα των πουλιών αυτουνού που παίζει τώρα ('B', 'W' για black και white αντίστοιχα)
    - **score**: ένας double που περιέχει την βαθμολογία τις κάθε κατάστασης
    - **x, y**: δύο ακέραιοι για της συντεταγμένες του πουλιού που παίχτηκε

Έπειτα έχουμε κάποιους **κατασκευαστές** για τα αντικείμενα μας με διάφορα ορίσματα, καθώς και **setters και getters** για τα ορίσματά μας .
Άλλες μέθοδοι που έχουμε είναι οι εξής :

- **add:** Δέχεται μια κατάσταση και την προσθέτει στα παιδιά της κατάστασής στην οποία είμαστε.
- **tableInitialization**: Αρχικοποιεί την ρίζα του δέντρου έτσι ώστε να αντιπροσωπεύει το αρχικό ταμπλό.
- **printTable:** Τυπώνει το ταμπλό στην κονσόλα με B τα μαύρα και W τα λευκά πούλια και * όπου δεν υπάρχει κανένα πούλι.
- **printWithPossibleMoves:** Τυπώνει το ταμπλό στην κονσόλα με B τα μαύρα και W τα λευκά πούλια και * όπου δεν υπάρχει κανένα πούλι ,αλλά στα σημεία που μπορούμε να παίξουμε τυπώνει + αντί για *.
- **countChar:** Μετράει τα B ,W και τα * στο ταμπλό κάθε κατάστασης.
- **sameStates:** Ελέγχει αν η κατάστασή στην οποία καλέστηκε και αυτή που πήρε σαν όρισμα είναι ίδιες.
- **checkRules:** Κοιτάει σε όλες τις θέσεις του πίνακα που είναι άδειες ('*') , άμα έχουν πούλι του αντιπάλου δίπλα και καλεί την changeTiles για εκείνη την κατεύθυνση. Οι κατευθύνσεις φαίνονται στους παρακάτω πίνακες . Για κάθε θέση μπορεί να χρειαστεί να κοιταχτούν πάνω από μία κατευθύνσεις . Αν σε μία κατάστασή δεν μπορούμε να προσθέσουμε πούλι του παίχτη ,του οποίου ήταν η σειρά ,τότε σαν παιδί της κατάστασης αυτής βάζουμε μια κατάσταση που έχει ίδιο ταμπλό με την τωρινή απλά για x,y βάζουμε -1 ,αυξάνουμε το βάθος κατά 1 και οι υπόλοιπες τιμές παίρνουν default τιμές.

1	2	3
4		5
6	7	8

(i-1,j-1)	(i-1,j)	(i-1,j+1)

(i,j-1)	(i,j)	(i,j+1)
(i+1,j-1)	(i+1,j)	(i+1,j+1)

Π.χ.:(* Ο πίνακας δεν αποτελεί απαραίτητα δυνατή κατάσταση απλά χρησιμοποιείται για επεξηγηματικούς λόγους)

	0	1	2	3	4	5	6	7
A								
B								
C	B	*	*					
D	*	B	W					
E	*	B	*					
F								
G								
H								

Εδώ αν θέλαμε να βάλουμε ένα άσπρο πούλι (W) στο (E , 2) θα πρέπει να κοιταχτούν οι κατευθύνσεις 1 και 4 για το άμα μπορούμε να βάλουμε το πουλί και ποιες αλλαγές θα γίνουν.

- **changeTiles:** Δέχεται δύο καταστάσεις ,μια θέση μια κατεύθυνση και το χρώμα του παίχτη που παίζει και ελέγχει αν σε εκείνη την κατεύθυνση υπάρχει πούλι του ιδίου χρώματος και αν ενδιάμεσα παρεμβάλλονται μόνο πούλια του αντιπάλου . Τότε ξέρουμε ότι μπορεί να γίνει η κίνηση ,οπότε αρχικοποιούμε μια κατάσταση αν δεν είναι ήδη αρχικοποιημένη από άλλη κλήση της changeTiles και κάνουμε τις αλλαγές στα ενδιάμεσα πούλια, που γίνονται πούλια του ίδιου χρώματος. Τέλος επιστρέφεται η κατάσταση αυτή που ή θα είναι null αν δεν έγινε κίνηση ή θα είναι κάποια ενδιάμεση κατάστασή αν δεν έχουν γίνει ακόμη όλοι οι έλεγχοι ή θα είναι η κατάσταση που προκύπτει από την κίνηση.

Π.χ.:(* Ο πίνακας δεν αποτελεί απαραίτητα δυνατή κατάσταση απλά χρησιμοποιείται για επεξηγηματικούς λόγους)

	0	1	2	3	4	5	6	7
A								
B	*	*	B	*				
C	B	*	*	*				

D	*	W	W	B	
E	*	B	*	*	
F		W	B	*	
G	*	*	*	*	
H					

Εδώ αν θέλουμε να παίξουμε ένα μαύρο πούλι (B) στο (E,2) τότε θα ελεγχθούν όλες οι κατευθύνσεις γύρω του και ενώ θα γίνει η κλήση της changeTiles για τις κατευθύνσεις 1,2,6 αφού μόνο προς εκείνες τις κατευθύνσεις υπάρχει πούλι του αντίθετου χρώματος , θα γίνουν αλλαγές μόνο στην κατεύθυνση 1 όπου η changeTiles θα συναντήσει πούλι του ίδιου χρώματος χωρίς να παρεμβάλλεται κάτι άλλο εκτός από πούλι του αντίθετου χρώματος.

- **skip_turn:** Χρησιμοποιείται για να δούμε αν μπορεί ο παίχτης που είναι να παίξει να κάνει κάποια κίνηση ή όχι και επιστρέφει true και false αντίστοιχα . Αν καλεστεί για μία κατάσταση και επιστρέψει false αναφέρθηκε πιο πριν και στην checkRules ότι για τις καταστάσεις που δεν έχουν παιδιά βάζουμε στο παιδί το ίδιο ταμπλό και κάνουμε κάποιες αλλαγές στα πεδία το παιδιού. Αν και στο παιδί η skip_turn βγάλει false σημαίνει ότι η αρχική ήταν τερματική κατάσταση (δηλαδή ο “παππούς” της),αφού κανένας από τους παίχτες δεν μπορεί να παίξει.