1η Εργασία

Οδυσσέας Σπυρόπουλος p3200183

Μαρία - Ιωάννα Μουζουράκη p3200105

Στην 1η εργασία του μαθήματος, υλοποιήσαμε ένα πρόγραμμα μέσω του οποίου ο χρήστης μπορεί να παίξει το παιχνίδι Othello.

Για την δημιουργία του προγράμματος , αναπτύξαμε τις εξής τέσσερις κλάσεις :

Move, **Player**, **Board** & **Main.**

Move:

Η κλάση **Move** χαρακτηρίζει μία κίνηση στο ταμπλό του παιχνιδιού.

Γνωρίσματα κάθε κίνησης είναι το χρώμα της (0:μαύρο ,1:άσπρο), οι συντεταγμένες της x και y , και η αξία της η οποία καθορίζεται από την μέθοδο evaluate (υλοποιείται στην κλάση **Board**).

Player:

Η κλάση **Player** χαρακτηρίζει έναν παίκτη του Othello.

Γνωρίσματα κάθε παίκτη είναι το χρώμα του , και το μέγιστο βάθος αναζήτησης του αλγορίθμου MiniMax. Παίρνουν τιμές στην **Main** , οι οποίες εξαρτώνται από τις επιλογές του χρήστη.

Όταν έρχεται η στιγμή για τον υπολογιστή να παίξει, επιλέγει κίνηση χρησιμοποιώντας τον αλγόριθμο MiniMax. Για να είναι αυτό εφικτό , δημιουργήσαμε τις εξής μεθόδους:

MiniMax: Αν ο παίκτης έχει το μαύρο χρώμα, καλείται η μέθοδος Max, ενώ αν έχει το άσπρο χρώμα καλείται η Min. Καλείται στην **Main** όταν είναι σειρά του υπολογιστή να παίξει.

Max: Καλεί τις μεθόδους Max και Min, την μία μετά την άλλη, μέχρι να φτάσει το μέγιστο βάθος αναζήτησης ή τον τερματικό κόμβο (το ελέγχουμε καλώντας την μέθοδο isTerminal που υλοποιείται στην κλάση Board). Αν έχουμε φτάσει στην προαναφερόμενη κατάσταση , επιστρέφει την τελευταία κίνηση. Διαφορετικά δημιουργεί την κίνηση maxMove δίνοντάς της μία ελάχιστη αξία, και καλεί την getChildren (υλοποιείται στην κλάση **Board**) για την δημιουργία μίας λίστας με τα παιδιά-κινήσεις του παίκτη. Για κάθε παιδί καλείται η Min,και συγκρίνεται η τιμή που επιστρέφει η Min με την maxMove,ώστε να επιστραφεί η κίνηση με την μεγαλύτερη αξία. Τέλος , κάνει πριόνισμα α-β , δίνοντας στο α (δίνεται ως παράμετρος) την αξία της κίνησης που επέστρεψε η Min αν αυτή είναι μεγαλύτερή του. Ύστερα συγκρίνει το α με το β (δίνεται ως παράμετρος),και αν το α δεν είναι μικρότερο του β σταματάει την εκτέλεση της Max.

Min:

Είναι αντίστοιχη της Max.

Board :

Η κλάση **Board** χαρακτηρίζει το ταμπλό του παιχνιδιού μέσω των παρακάτω μεθόδων:

isMoveValid: Ελέγχει πρώτα αν είναι κενή η θέση που θέλουμε να βάλουμε το δίσκο. Αν είναι, ελέγχουμε και τις 8 γειτονικές θέσεις για να βρούμε *τουλάχιστον ένα* δίσκο αντίθετου χρώματος. Αν τον βρούμε τότε ακολουθάμε τη γραμμή (κάθετη, οριζόντια ή διαγώνιος) και ελέγχουμε αν καταλήγει σε δίσκο ίδιου χρώματος με εκείνον της δοθείσας κίνησης. Αν ισχύει, τότε η κίνηση είναι έγκυρη, διαφορετικά δεν είναι. Καλείται στις μεθόδους possibleMoves, getChildren, isTerminal και στη **Main**.

makeMove: Όπου καλείται, η κίνηση η οποία δίνεται ως παράμετρος, έχει ήδη ελεγχθεί για την εγκυρότητά μέσω της isMoveValid. Επομένως μπορεί να εκτελεστεί. Αρχικά, βάζει το χρώμα της κίνησης (0:μαύρο ,1:άσπρο) στη θέση του ταμπλό με τις συντεταγμένες της κίνησης. Ύστερα διατρέχει το ταμπλό με τον ίδιο τρόπο της μεθόδου isMoveValid, όμως εδώ, ελέγχει και ακολουθεί *όλους* τους δίσκους αντίθετου χρώματος. Αποθηκεύει σ ένα πίνακα τις θέσεις που έχουν γίνει out-flank και τους δίνει το χρώμα της κίνησης. Καλείται στην **Main**, όταν κάνει κάποιος παίκτης κάνει κίνηση, και στην getChildren.

possibleMoves: Διατρέχει όλο το ταμπλό και ελέγχει για κάθε θέση του , αρχικά αν είναι κενή , και ύστερα μέσω της isMoveValid ελέγχει αν μία κίνηση σε αυτήν την θέση είναι έγκυρη. Οι έλεγχοι αυτοί γίνονται με βάση το χρώμα του παίκτη , το οποίο δίνεται ως παράμετρος. Επιστρέφει το αριθμό των κινήσεων που πληρούν τις παραπάνω προϋπόθεση. Καλείται στην evaluate για να πάρουμε τον αριθμό των πιθανών κινήσεων για τον κάθε παίκτη , και στην Main για να ελέγξουμε αν απομένουν έγκυρες κινήσεις στον κάθε παίκτη.

getChildren: Επιστρέφει μια ArrayList με όλα τα παιδιά (=τις διαθέσιμες κινήσεις) για τον παίκτη με το χρώμα που δίνεται ως παράμετρος. Οποιοδήποτε κενή θέση στο ταμπλό είναι διαθέσιμη κίνηση. Καλείται στις μεθόδους Max και Min (υλοποιούνται στην κλάση **Player**).

isTerminal: Ελέγχει μέσω της μεθόδου isMoveValid για κάθε άδεια θέση του ταμπλό, αν μία κίνηση σε αυτήν την θέση είναι έγκυρη. Επιστρέφει true εάν το ταμπλό είναι γεμάτο ή δεν έχουν απομείνει έγκυρες κινήσεις. Αν υπάρχει τουλάχιστον μία κενή θέση και μια έγκυρη κίνηση ,επιστρέφει false.Καλείται στην while loop της **Main** ,όπου όσο επιστρέφει την τιμή false συνεχίζεται το παιχνίδι.

evaluate: Χρησιμοποιεί τις παρακάτω 3 συναρτήσεις, για να υπολογίσει την αξία μίας κίνησης.

f1 : αριθμός μαύρων δίσκων - αριθμός άσπρων δίσκων

f2 : αριθμός μαύρων δίσκων που βρίσκονται στις γωνίες του πίνακα - αριθμός άσπρων δίσκων που βρίσκονται στις γωνίες

f3 : αριθμός μαύρων δίσκων που βρίσκονται σε μη γωνιακές ακραίες θέσεις - αριθμός άσπρων δίσκων που βρίσκονται σε μη γωνιακές ακραίες θέσεις

Η τελική αξία που επιστρέφει η evaluate είνα η εξής :

f1 + 3*f2 + 2*f3

Χρησιμοποιείται στις μεθόδους Max & Min (υλοποιούνται στην κλάση **Player**).

printBoard: Τυπώνει το ταμπλό , ως εξής: όπου είναι κενή η θέση του ταμπλό τυπώνει _ , όπου βρίσκεται δισκίο του παίκτη που έχει το μαύρο χρώμα τυπώνει X και αντίστοιχα τυπώνει O για το άσπρο. Καλείται στην αρχή του παιχνιδιού και ύστερα από την εκτέλεση κάθε κίνησης.

getScore: Διατρέχοντας όλο το ταμπλό μετράει και επιστρέφει τον αριθμό των μαύρων (X) ή των άσπρων (O) δισκίων, ανάλογα με τον αριθμό που δίνεται ως παράμετρος (0: μαύρο ,1: άσπρο). Καλείται στο τέλος του παιχνιδιού για την εμφάνιση των σκορ των δύο παικτών.

Main:

Στην κλάση **Main** ξεκινάει η εκτέλεσης του προγράμματος.

Ξεκινώντας το παιχνίδι , ο χρήστης ζητείται αρχικά να επιλέξει

χρώμα και μέγιστο βάθος αναζήτησης του αλγορίθμου MiniMax. Αν έχει επιλέξει το μαύρο χρώμα παίζει πρώτος , διαφορετικά η πρώτη κίνηση είναι του υπολογιστή.

Πριν επιλέξει κίνηση ο κάθε παίκτης, ελέγχεται μέσω της μεθόδου possibleMoves (υλοποιείται στην κλάση **Board**) αν μπορεί να εκτελέσει έγκυρες κινήσεις. Αν δεν μπορεί, χάνει την σειρά του και παίζει ο επόμενος.

Δημιουργούμε δύο αντικείμενα της κλάσης **Player** , το human (χρήστης) και το computer (υπολογιστής) . Δίνουμε τιμές στις μεταβλητές color και depth των αντικειμένων ανάλογα με τις επιλογές που έκανε ο χρήστης.

Όταν είναι η σειρά του χρήστη, ζητείται να δώσει τις συντεταγμένες της κίνησης που θέλει να εκτελέσει. Ελέγχεται αν η κίνηση είναι έγκυρη μέσω της μεθόδου isMoveValid (υλοποιείται στην κλάση **Board**) καθώς επίσης και αν οι συντεταγμένες της κίνησης δεν υπερβαίνουν τα όρια του ταμπλό. Αν η κίνηση που επέλεξε πληρεί τις προαναφερόμενες προϋποθέσεις, τότε εκτελείται. Αν όχι, ζητείται από τον παίκτη να δώσει διαφορετικές συντεταγμένες.

Όταν είναι η σειρά του υπολογιστή, θα γίνει η επιλογή κίνησης μέσω της μεθόδου MiniMax (υλοποιείται στην κλάση **Player**).

Μετά από κάθε κίνηση, εμφανίζεται το αποτέλεσμά της στο ταμπλό , και δίνουμε την τιμή του επόμενου παίκτη στην μεταβλητή nextPlayer μέσω της μεθόδου setNextPlayer (υλοποιείται στην κλάση **Board**).

Το παιχνίδι τελειώνει αν κανένας από τους δύο παίκτες δεν μπορεί να εκτελέσει έγκυρη κίνηση ,ή αν είναι γεμάτες όλες οι θέσεις στο ταμπλό (όταν βγούμε από το while loop).Με την λήξη του παιχνιδιού υπολογίζονται τα σκορ κάθε παίκτη μέσω της μεθόδου getScore (υλοποιείται στην κλάση **Board**) , και ανακοινώνεται ο νικητής.

Παράδειγμα :

```
Which color do you choose? (Consider that black playes first!) (Answer with : 0 for Black / 1 for White)
0
What would you like the maximum search depth be?
3

  Y 1 2 3 4 5 6 7 8
X  ________________
1 | _ _ _ _ _ _ _ _ | 1
2 | _ _ _ _ _ _ _ _ | 2
3 | _ _ _ _ _ _ _ _ | 3
4 | _ _ _ X O _ _ _ | 4
5 | _ _ _ O X _ _ _ | 5
6 | _ _ _ _ _ _ _ _ | 6
7 | _ _ _ _ _ _ _ _ | 7
8 | _ _ _ _ _ _ _ _ | 8
   ________________
    1 2 3 4 5 6 7 8

Blacks: 2  /  Whites: 2

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

It's your turn!

Enter X,Y coordinations:
X:6
Y:4

BOARD:

  Y 1 2 3 4 5 6 7 8
X  ________________
1 | _ _ _ _ _ _ _ _ | 1
2 | _ _ _ _ _ _ _ _ | 2
3 | _ _ _ _ _ _ _ _ | 3
4 | _ _ _ X O _ _ _ | 4
5 | _ _ _ X X _ _ _ | 5
6 | _ _ _ X _ _ _ _ | 6
7 | _ _ _ _ _ _ _ _ | 8
8 | _ _ _ _ _ _ _ _ | 8
   ________________
    1 2 3 4 5 6 7 8
```

```
Blacks: 4  /  Whites: 1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

It's AI's turn!

Calculating move...
Move: 6, 5

BOARD:

  Y 1 2 3 4 5 6 7 8
X  ________________
1 | _ _ _ _ _ _ _ _ | 1
2 | _ _ _ _ _ _ _ _ | 2
3 | _ _ _ _ _ _ _ _ | 3
4 | _ _ _ X O _ _ _ | 4
5 | _ _ _ X O _ _ _ | 5
6 | _ _ _ X O _ _ _ | 6
7 | _ _ _ _ _ _ _ _ | 7
8 | _ _ _ _ _ _ _ _ | 8
   ________________
    1 2 3 4 5 6 7 8

Blacks: 3  /  Whites: 3

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

(...μετά από κάποιες κινήσεις)