Γεώργιος Χείρμπος
3130230
Τεχνητή Νοημοσύνη
Χειμερινό Εξάμηνο 2018-2019
Εργασία 1 : Reversi

Εισαγωγή

Σαν θέμα 1ης εργασίας ζητήθηκε η υλοποίηση του παιγνίου Reversi με χρήση του αλγορίθμου minimax pruning a-b.
Όλες οι μέθοδοι υλοποίησης περιλαμβάνονται στα αρχεία που βρίσκονται στο φακελο src.
Η υλοποίηση έγινε σε περιβαλλον windows 10 και με χρήση του atom IDE σε γλώσσα προγραμματισμού python 3.6.

Ο αλγόριθμος miniMax.

Ο minimax ειναι ένας αναδρομικός αλγόριθμος για την επιλογή κίνησης σε παίγνίδια n-αντιπάλων. Για κάθε καταστάση του παιγνίου υπάρχει μια αξια που αντιστοιχίζεται με αυτήν, ανάλογα αν κάποιος θέλει να μεγιστοποιήση ή να ελαχιστοποιήση την αξια της νέας κατάστασης μετά από κίνηση του, λέγεται αυτή η τιμή καλή.

Alpha-beta pruning

Λόγω του τεράστιου ποσού καταστασεων που μπορούν να παραχθούν από παιχνίδια όπως (reverse, σκάκι, go), η αποκοπή αποτελεί μια βελτιστοποίση πάνω στον minimax.
Χρησιμοποιώντας pruning παίρνουμε το ίδιο αποτέλεσμα που θα παίρναμε με τον απλό minimax, χωρίς να χρειαστεί να αναλύσουμε όλες τις πιθανές καταστάσεις που παράγονται. Για παράδειγμα αν σε ενα κομβο ελαχίστου έχουμε μια τιμή Α, και από τον κομβο μεγιστοποίησης πάρουμε μια τιμή Β, και Α < Β τότε θα αγνοήσουμε όλες τις καταστάσεις που θα παραχθούν από τον κόμβο μεγιστοποίησης.

Αρχεία

- abminimax.py

Περιλαμβάνονται οι μέθοδοι υλοποίηση του minimax.

- board.py

Περιλαμβάνονται οι μέθοδοι ελέγχου του ταμπλό, κινήσεων, score.

- game.py

Αρχικοποίση βασικών μεταβλητών και εκίνηση παιγνίου.

- stateNode.py

Αναπαράσταση με χρήση κόμβων των παιδιών της εκάστοτε κατάσταση παιχνιδιού.

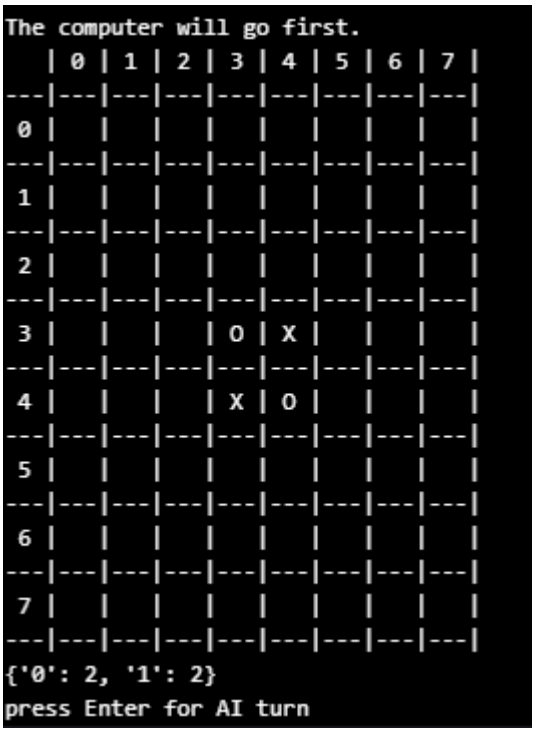
Οδηγίες και μέθοδοι

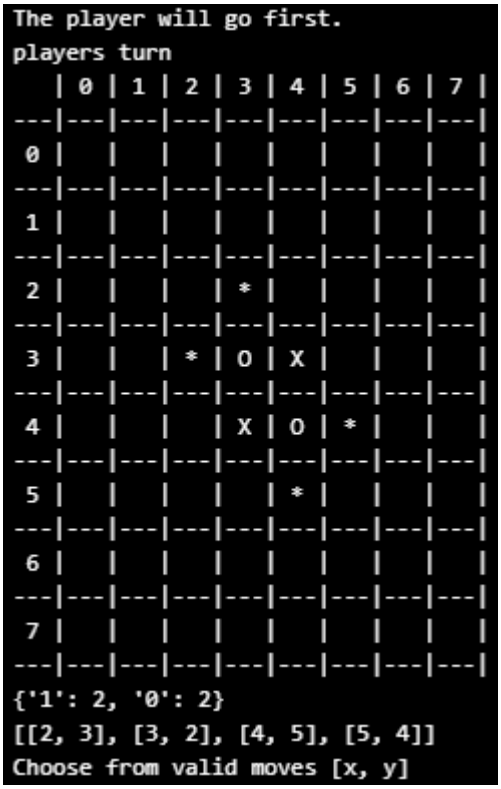
Για την εκτέλεση που παιχνιδιού χρησιμοποιούμε την εντολή “python .\game.py” βρισκόμενοι στο directory των αρχείων.

Κατά την έναρξη του παιχνιδιού αρχικοποιήται ο βασικός πίνακας που περιέχει τις τιμές -1 για άδεια θέση, 0 για τα μαύρα, 1 για τα άσπρα. Τα μαύρα συμβολίζονται ως X και τα άσπρα ως O.

Τυχαία αρχικοποιήται σε 0 ή 1 το tile του παίχτη (Τα μαύρα παντα παίζουν πρώτα κατα κανόνα).

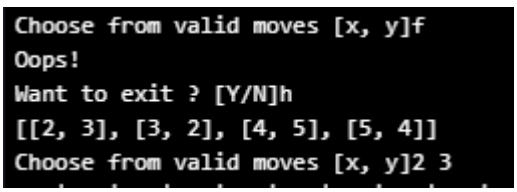
Ανάλογα τον παίχτη έχουμε αρχική εικόνα



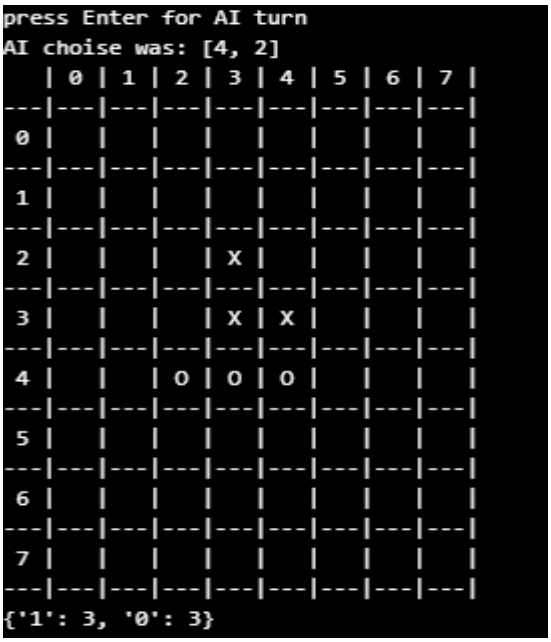


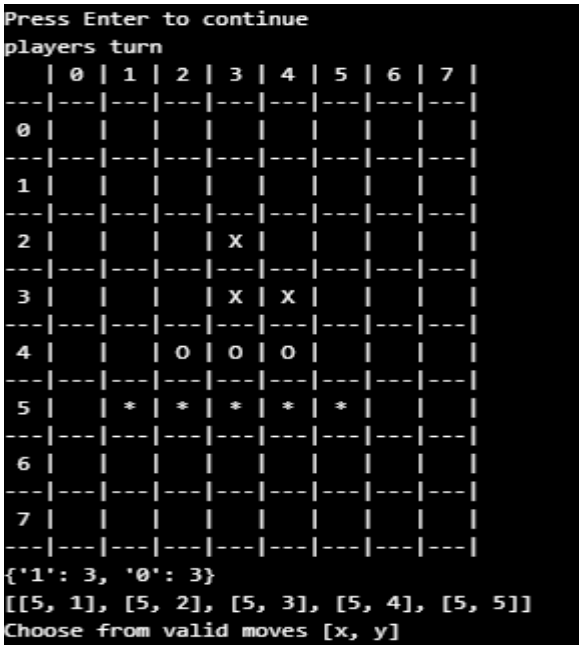
Για βοήθεια προς τον παίχτη τυπώνονται οι κινήσεις που μπορουν να γίνουν ως λίστα και συμβολίζονται ως * στο ταμπλό του. Για να παρουμε την κίνηση του παίχτη γράφουμε αυστηρά βάση το παρακάτω μοτίβο. “X[space]Y”.

Αν δεν δωθεί το σωστό το σωστό μοτίβο ή τιμή διάφορη αριθμού, τυπώνεται μήνυμα λάθος και ερωτείται ο παίχτης για έξοδο. Αν δεν δωθεί ο χαρακτήρα Y απλά ξαναερωτούμαστε για κίνηση.



Αφού δωθεί σωστή κίνηση. Πατάμε Enter και έχει σειρά ο υπολογιστής. Αφού χρησιμοποιηθεί ο minimaxAB, επιλέγεται η κίνηση και τυπώνεται ο πίνακας και το σκορ. Ξαναπατώντας enter παίρνει σειρά ο παίχτης και τυπώνεται ο πίνακας με τις βοήθειες και η λίστα των δυνατών κινήσεων.





Τελικά καταλήγουμε στην τελική κατάσταση και τυπώνεται ο νικητής

Επεξήγηση μεθόδων

board.py

def starterBoard() : αρχικοποιούμε το βασικό ταμπλό με -1,0,1

printBoard(board) : εκτύπωση του board με σύμβολα στη γραμμή εντολών. Όρισμα το ταμπλό.

copyBoards(originalBoard) : δημιουργία νέου ταμπλό για επεξεργασία χωρίς να πειραχτει το αρχικό. Όρισμα το αρχικό ταμπλό.

getBoardWithValidMoves(validBoard, tile) : βρίσκουμε όλες τις έγκυρες κινήσεις και τις επιστρέφουμε για έλεγχο της εισόδου του παίχτη. Επίσης τυπώνεται ο βοηθητικός πίνακας. Όρισμα το ταμπλό, και το tile του παίχτη.

getPlayerMove(board, tile, validMoves) : συνάρτηση επεξεργασίας κίνησης του παίχτης. Όρισμα το ταμπλό, το tile, και η λίστα με τις έγκυρες κινήσεις.

getComputerMove(board, playerTile, computerTile) : συνάρτηση επιλογής κίνησης του υπολογιστή. Όρισμα το ταμπλό, το tile του παίχτη και του υπολοστική.

makeMove(board, tile, xstart, ystart) : εφαρμογή της κίνησης στο ταμπλό. Καλείται από τις συναρτήσεις ***getPlayerMove, getComputerMove.*** Επιστρέφει το ανανεωμένο ταμπλό. Όρισμα το ταμπλό, το tile του παίχτης και οι συντεταγμένες εκίνησης.

getScoreOfBoard(board) : συνάρτηση για μέτρηση σκορ στο ταμπλό. Όρισμα το ταμπλό.

isValidMove(originalBoard, tile, xstart, ystart) : συνάρτηση αναζήτησης ελεγχου εγκυρότητας κίνησης καθώς και tiles προς εναλλαγή αν η κίνηση είναι έγκυρη. Όρισμα το ταμπλό, το tile του παίχτης και οι συντεταγμένες εκίνησης.

game.py

if __name__ == "__main__": αρχικοποίση tile παιχτων, δημιουργία αρχικού ταμπλού.

game_start(game_board, playerTile, computerTile, turn) : εκκίνηση παιχνιδιού. Όρίσματα το ταμπλό, tile παιχτών.

end_game(game_board, playerTile, computerTile) : Μετά το πέρας του παιχνιδιού τύπωση σκορ βάση του τελικού ταμπλού.

abminimax.py

execute(game_board, playerTile, computerTile) : αρχικές τιμες και εκτέλεση του minimax. Ορίζουμε βάθος αναζήτησης, αλφα, βετα και επιστρέφουμε την βέλτιστη κίνηση. Δημιουργόυμε ρίζα για την κατάσταση με ορίσματα (self, curBoard, move, player, value=0, parentBoard=None, children=[]).

def minimax(state, depth, alpha, beta, mizing): εκτέλεση αλγορίθμου. Για καθε κατάσταση δημιουργούμε κόμβο που περιέχει (self, curBoard, move, player, value=0, parentBoard, children=[]) και βρίσκουμε τις έγκυρες κινήσεις για αυτό. Για όλες τις έγκυρες κινήσεις δημιουργούμε νέο κόμβο με νέα κατάσταση ταμπλού, το προσθέτουμε στο πατέρα ως παιδί και κάνουμε minimax σε αυτό. Η ίδια διαδικασία μέχρι μεγιστο βάθος. Στο μεγιστο βάθος αξιολογούμε την κατάσταση και την επιστρέφουμε στον πατέρα. Γίνεται ο έλεγχος ανάλογα αν κάνουμε minimize ή maximize και είτε pruning είτε εξαντλούνται όλες οι καταστάσεις και επιστρέφεται η βέλτιστη κίνηση και καλύτερο σκορ.

def stateScore(state, mizing) αξιολόγηση του state με βάση των παρακάτω συναρτήσεων

def getTileDifference(state, mizing) : εύρεση διαφοράς σε tiles.

def isCornerMove(move) : έλεγχος αν η κίνηση είναι σε γωνία.

def isNearEdge(move) : έλεγχος αν η κίνηση είναι ή πλησιάζει πλευρά.

def isOnCorner(x, y) : βοηθητική στην isCornerMove.