ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

(ΕΡΓΑΣΙΑ 1)

ΘΕΜΑ: REVERSI
ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΓΕΩΡΓΟΠΟΥΛΟΥ ΚΩΝΣΤΑΝΤΙΝΑ
ΑΜ: 3180029

-ΤΡΟΠΟΣ ΧΡΗΣΗΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ:

Θα ξεκινήσει το πρόγραμμα από την κλάση start η οποία περιέχει και τη main. Θα ζητήσει από το χρήστη να επιλέξει εάν θα παίξει ο υπολογιστής πρώτος (και θα αποθηκευτεί η επιλογή του στη boolean μεταβλητή c_moves) και ζητά και το βάθος στο οποίο θα κάνει αργότερα ο υπολογιστής αναζήτηση για την καλύτερη επιλογή των κινήσεων του μέσω της minimax (και το αποθηκεύει στη μεταβλητή my_depth. Ύστερα ξεκινά το παιχνίδι με τη δημιουργία του αντικειμένου game_reversi. Το παιχνίδι συνεχίζει μέχρις ότου ο παίχτης και ο υπολογιστής φτάσουν τα 64 πλακίδια, δηλαδή να γεμίσει όλος ο πίνακας ((Integer)game_reversi.b.disks(1)).intValue() + ((Integer)game_reversi.b.disks(2)).intValue() == 64) ή μέχρι να μην έχει κανένας από τους δύο διαθέσιμες κινήσεις ((b.ValidMoves(1)).size() == 0) && ((b.ValidMoves(2)).size() == 0). Εάν τερματίσει η επανάληψη εμφανίζει τα κατάλληλα μηνύματα με τον νικητή.

Κατά την έναρξη του παιχνιδιού καλείται ο constructor της start και φτιάχνει νέο πίνακα μέσω της κλάσης board και εμφανίζει τον πίνακα θέσεων με την μέθοδο printGame, έτσι όπως έχει διαμορφωθεί από την initialState.

Σε κάθε κίνηση (που εκτελείται από της reversiGame) υπάρχουν δύο περιπτώσεις ανάλογα με την επιλογή του χρήστη για το αν θα παίξει ο υπολογιστής ή οχι. Στην πρώτη περίπτωση καλέιται η computerMoves που έχει όρισμα το βάθος που επέλεξε ο χρήστης για να δωθεί η καλύτερη επιλογή κίνησης για τον υπολογιστή μέσω της minimax.

Πιο συγκεκριμένα η computerMoves ελέγχει τη διαθεσιμότητα των κινήσεων του υπολογιστή μέσω της ValidMoves, και αν δεν υπάρχουν εκτυπώνει αντίστοιχο μήνυμα και χάνει τη σειρά του ο υπολογιστής και αν δεν λήξει το παιχνίδι από τις συνθήκες που ελέγχει η main εκτελείται πάλι η reversiGame και παίζει αύτη τη φορά ο παίχτης. Αν υπάρχουν διαθέσιμες κινήσεις ξεκινά η καταγραφή του χρόνου εκτέλεσης στη μεταβλητή startTime, ο μέγιστος χρόνος που θα επιτρέπεται η αναζήτηση στη μεταβλητή maxTime, και η μεταβλητή cutOff που θα σταματήσει την περαιτέρω αναζήτηση σε περίπτωση που ξεπεραστεί ο χρόνος της maxTime. Ύστερα δημιουργείται κλώνος του πίνακα κινήσεων μέσω της cloneGrid της κλάσης board ώστε να υπάρχει βαθύ αντίγραφο του πίνακα αλλά χωρίς να έχουν την ίδια αναφορά και οι πιθανές μετατροπές του πίνακα αυτού να μην επιρεάσουν τις ήδη υπάρχουσες κινήσεις που έχουν καταγραφεί. Μετά ξεκινά η επιλογή της κίνησης του υπολογιστή μεχρι να φτάσει το βάθος που έδωσε ο χρήστης ή μέχρι να λήξει ο χρόνος. Σε κάθε επανάληψη καταχωρείται από τη reversi.best η καλύτερη επιλογή στον πίνακα ώστε αν λήξει ο χρόνος να έχει αποθηκευτεί η επιλογή της προηγούμενης επανάληψης. Τέλος εμφανίζει στο χρήστη μήνυμα για το βάθος που κάλυψε κατά την αναζήτηση και ρωτά αν θα πραγματοποιηθεί από το χρήστη η επόμενη κίνηση. (η MiniMax θα αναληθεί παρακάτω)

Όταν παίζει ο χρήστης του ζητείται στήλη και γραμμή στην οποία θέλει να τοποθετήσει το πούλι του. Μέσω της playerMoves ελέγχεται αν έχει

διαθέσιμες κινήσεις (από τη ValidMoves) και αν μπορεί να πραγματοποιήσει την κίνηση που επέλεξε μέσω της μεθόδου move. Στην πρώτη περίπτωση εμφανίζει μήνυμα και ενημερώνει το χρήστη για την τη μη διαθεσιμότητα κινήσεων και ότι η επόμενη κίνηση θα πραγματοποιηθεί από τον υπολογιστή, ενώ στη δευτερη αν δεν είναι έγκυρη επιτρέπει την αλλαγή της επιλογής του. Η μέθοδος move της κλασης board επιστρέφει μια boolean τιμή για το εάν επιτρέπεται η κίνηση που επέλεξε ο χρήστης ή όχι. Συγκεκριμένα ελέγχει αν η θέση είναι ήδη καλλυμένη και μετά ελέγχει ποιά πούλια που ανήκουν στον αντίπαλο πρέπει να αντικατασταθούν από του παίχτη μέσω της flip (σε αυτή την περίπτωση ποια πούλια του υπολογιστή θα αντικατασταθόυν από του χρήστη ανάλογα με την κίνηση που επέλεξε να κάνει).
Τέλος, μετά από κάθε κίνηση είτε του υπολογιστή είτε του χρήστη εκτυπώνει τον πίνακα μέσω της printGame που χρησιμοποιεί την SetBlock ώστε να του βάλει “X”, “O”, και “-”.

-ΔΥΝΑΤΟΤΗΤΕΣ ΠΡΟΓΡΑΜΜΑΤΟΣ:

Εντοπίζει τις διαθέσιμες κινήσεις του κάθε παίχτη.

Δείχνει στον χρήστη την εγκυρότητα της επιλογής κίνησης του.

Εμφανίζει τον πίνακα στον χρήστη ώστε να έχει εικόνα πριν από κάθε κίνηση του πως είναι διαμορφωμένος ο πίνακας.

Δίνει τη δυνατότητα στο χρήστη να επιλέξει αν θα παίξει σε κάθε γύρο και επιλέγει και το μέγεθος της αναζήτησης που θα κάνει ο υπολογιστής για τις δικές του κινήσεις.

Κάνει γνωστό στο χρήστη το βάθος που έκανε αναζήτηση ο υπολογιστής για την κίνησή του.

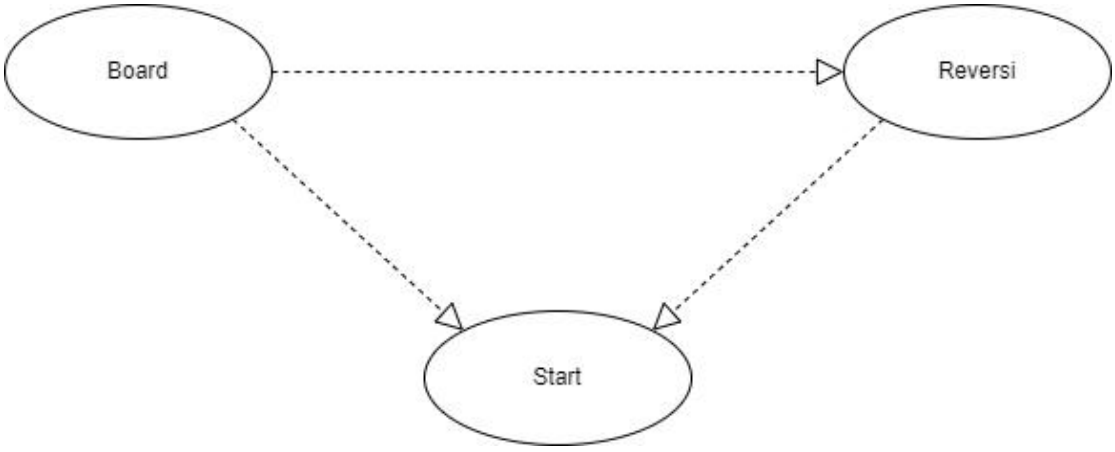
-ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ:

Class start: Αποτελείται από τη main και ξεκινά το παιχνίδι που ο χρήστης κάνει τις επιλογές κινήσεων του, εμφανίζεται ο πίνακας και τα αποτελέσματα με τη λήξη του παιχνιδιού.

Class board: Διαμορφώνει τον πίνακα ανάλογα με τις κινήσεις των παιχτών (αλλάζοντας τα πλακίδια), έχει τις μεθόδους που μετράνε τους δίσκους του κάθε παίχτη και τις έγκυρες κινήσεις τους και τέλος και την μέθοδο που φτιάχνει κλώνο του πίνακα.

Class reversi: Υλοποιείται η MiniMax.

Επικοινωνία κλάσεων:



-ΜΕΘΟΔΟΣ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ:

Για την εύρεση της λύσης, δηλαδή για την πιθανή καλύτερη κίνηση του υπολογιστή, χρησιμοποιείται MiniMax με πριόνισμα α-β. Συγκεκριμένα, από τη μέθοδο computerMoves, της κλάσης start, καλείται επαναληπτικά για κάθε βάθος (μέχρι να φτάσει το μέγιστοβάθος που επέλεξε ο χρήστης ή μέχρι να εξαντληθεί ο χρόνος) η MiniMax που δέχεται σαν ορίσματα το αν παίζει ο παίχτης Max, δηλάδη ο υπολογιστης, το επίπεδο από το οποίο θα ξεκινήσει την αναζήτηση, τα α και β που είναι οι οριακές τιμές που μπορεί να ελέγξει ώστε σύμφωνα με αυτές να γίνει το πριόνισμα (το α είναι η καλύτερη, δηλαδή η υψηλότερη, τιμή που μπορεί να "εγγυηθεί" για τον Max, ενώ το β είναι η καλύτερη, δηλαδή η χαμηλότερη, τιμή για τον Min), ο πίνακας των θέσεων πάνω στον οποίο θα γίνει το πριόνισμα (για αυτό και χρησιμοποιείται κλώνος του και όχι ο ίδιος ο πίνακας) και τέλος το μέγιστο επίπεδο βάθους που πρέπει να εξερευνήσει σε αυτή την επανάληψη.

Η MiniMax εφαρμόζεται μέσα στην κλάση reversi και ξεκινά με το εάν υπάρχει διαθέσιμος χρόνος ή επίπεδο για εξερεύνηση. Αν δεν υπάρχει ενημερώνει την cutOff ώστε να σταματήσουν οι επιπλεόν αναζητήσεις παρακάτω επιπέδων και επιστρέφει τη διαφορά των ήδη υπαρχόντων δίσκων.

Αν περάσει τον πρώτο έλεγχο μετά ελέγχει αν η Array List moves, που περιέχει τις διαθέσιμες κινήσεις του υπολογιστη (ValidMoves(1)) και του παίχτη (ValidMoves(2)). Αν η move είναι κενή τότε σημαίνει πως βρίσκεται σε φύλλο και επιτρέφει 64 αν ο παίχτης που εξετάζει είναι ο υπολογιστής ή -64 αν είναι ο χρήστης (διότι σημαίνει ότι έχουν καλύψει όλο τον πίνακα).

Μετά ο επόμενος έλεγχος αφορά το εάν η κίνηση αφορά τον υπολογιστή (δηλαδή τον Max), ή τον χρήστη (δηλαδή τον Min). Για τον Max θα επιλέξει την υψηλότερη τιμή από αυτές που βρίσκονται στο παρακάτω

επίπεδο (καλώντας τη minimax(false, level + 1, alpha, beta, moves.get(i), maxLevel)). Το κάτω επίπεδο επειδή θα είναι ο Min και θα έχει πάρει την ελάχιστη τιμή από το πιο κάτω επίπεδο (καλώντας τη minimax(true, level + 1, alpha, beta, i, maxLevel)) και αυτή η διαδικασία επαναλαμβάνεται μέχρις ότου είτε λήξει ο χρόνος είτε εξερευνηθεί το μέγιστο επιτρεπτό βάθος. Θα γίνει πιο εύκολα αντιληπτό εάν σκεφτούμε τη διαδικασία ανάποδα (όπως και πραγματικά εφαρμόζεται), δηλαδή αν για παράδειγμα στο προτελευταίο επίπεδο έχουμε Max παίχτη οι κόμβοι παίρνουν την max τιμή από τα παιδιά τους. Αντίστοιχα οι γονείς των max κόμβων επειδή είναι Min θα επιλέξουν την μικρότερη τιμή των παιδιών τους και έτσι συνεχίζεται η διαδικασία μέχρι να φτάσει στην κορυφή του δέντρου.
Το πριόνισμα α-β αποτρέπει τους γονείς να ψάξουν σε υποδέντρα τα οποία έχουν τιμές που δεν θα πάρουν ποτέ οι γονείς τελικά. Πιο συγκεκριμένα, στην αρχή ο πρώτος κόμβος που είναι ο Max θα έχει ως όρια τον μικρότερο και μεγαλύτερο ακέραιο. Αυτά τα όρια θα μεταφερθούν μέχρι τον πρώτο γονέα του μέγιστου επιτρεπτού βάθους. Αν αυτός ο κόμβος είναι Max τότε το το α παίρνει την max τιμή από τα παιδιά του και τα όρια διαμορφώνονται ως α <= επιλογή <= μέγιστος ακέραιος. Αν ο κόμβος είναι Min το β παίρνει την ελάχιστη τιμή από τα παιδιά του και τα όρια διαμορφώνονται ως ελάχιστος ακέραιος <= επιλογή <= β. Ο γονιός του κόμβου αυτού, στην περίπτωση που ο κόμβος είναι Max, είναι Min οπότε τα όρια διαμορφώνονται ως ελάχιστος ακέραιος <= επιλογή <= β. Τα ίδια όρια διαμορφώνονται και στο επόμενο παιδί που θα είναι Max, οπότε κοιτώντας και αυτό με τη σειρά του το πρώτο του παιδί διαμοργώνει ανάλογα και το α, αν είναι δηλαδή μεγαλύτερο από την ήδη υπάρχουσα τιμή του (που επειδή έχουμε στο α = ελάχιστος ακέραιος τότε θα αλλάξει) και έτσι δημιουργούνται τα όρια α >= επιλογή >= β. Αν τα υπόλοιπα παιδιά είναι μεγαλύτερα από το α δεν εξερευνούνται (πριόνισμα απο το α και πάνω). Η αντίθετη διαδικασία

γίνεται για το β, δηλαδή αν το παιδί κάποιου κόμβου έχει τιμή μικρότερη από το β.
Σε περίπτωση που γίνει πριόνισμα είτε στο άνω φράγμα (για τον Min) είτε στο κάτω (για τον Max), η μέθοδος minimax επιστρέψει το νέο όριο. Η καλύτερη επιλογή κίνησης για τον υπολογιστή της συγκεκριμένης επανάληψης αποθηκεύεται στην μεταβλητή best ανάλογα με το αν η τιμή που βρήκε είναι μεγαλύτερη από το κάτω φράγμα α. Με αυτό τον τρόπο συνεχίζεται ο αλγόριθμος μέχρι να βρει την καλύτερη επιλογή.
Γενικότερα αξίζει να σημειωθεί ότι το πριόνισμα α-β σε έναν τυπικό αλγόριθμο MiniMax επιστρέφει την ίδια κίνηση με τον τυπικό αλγόριθμο, αλλά αφαιρεί όλους τους κόμβους που δεν επηρεάζουν πραγματικά την τελική απόφαση αλλά κάνουν τον αλγόριθμο αργό. Ως εκ τούτου, με το πριόνισμα αυτών των κόμβων, καθιστά τον αλγόριθμο γρήγορο.

Η γενική ιδέα για το reversi είναι ότι είτε παίξει ο χρήστης πρώτος είτε όχι ο υπολογιστής είναι ο Max παίχτης και ο χρήστης θεωρεί ότι θα κάνει τη Min κίνηση. Στο πρόγραμμα ο χρήστης δεν μπορεί να ξέρει ποιά θα είναι η Min κίνηση για αυτό και επιλέγει ο ίδιος κίνηση και δεν τρέχει κάποιος αντίστοιχος αλγόριθμος για τη δική του επιλογή κίνησης.