Τεχνητή Νοημοσύνη (2022-2023)

Αλμπάνη Μάγδα (P3200005)

Βελισσαρίδης Γεώργιος (P3210255)

Ρηγάτος Διονύσιος (P3200262)

Εργασία 1

Εισαγωγή

Στην εργασία αυτή επιλέξαμε να υλοποιήσουμε το επιτραπέζιο παιχνίδι Othello στην γλώσσα προγραμματισμού Java. Ο αλγόριθμος τεχνητής νοημοσύνης που χρησιμοποιήθηκε είναι ο Minimax w/ a-b Pruning. Το πρόγραμμά μας επιτρέπει στον χρήστη να παίξει με αντίπαλο τον υπολογιστή, αφού επιλέξει το επιθυμητό του χρώμα (και συνεπώς με ποια σειρά θα παίξει). Επίσης, το πρόγραμμα παραλείπει την σειρά ενός παίκτη εάν δεν μπορεί να παίξει και την παραχωρεί στον αντίπαλό του, όπως απαιτεί η εκφώνηση.

Εκκίνηση του Παιχνιδιού

```
Welcome to Othello!
The game is played on an 8x8 board. You will be matched against the computer.
The player with the black disks goes first and the player with the white disks goes second.
```

```
Choose the game mode:
1. Human vs AI
2. AI vs AI
3. Human vs Human
```

Το παιχνίδι καλωσορίζει τον παίκτη και αμέσως του ζητάει να διαλέξει τύπο παιχνιδιού, δηλαδή εάν θα παίξει εναντίων του υπολογιστή, εναντίων άλλου παίκτη η θέλει να δει τον υπολογιστή να παίζει (simulation). Στην περίπτωση του Human vs AI και AI vs AI, ο παίκτης πρέπει να εισάγει το μέγιστο βάθος του Minimax, με κάποιες προτάσεις δυσκολίας.

```
Please pick the maximum depth for the AI.
Recommendations:
(2) Easy
(6) Medium
(10) Hard
```

```
You have chosen a depth of 2. Best of luck!
```

```
Do you want to play first with the black disks? (y/n)
y
```

Αφού διαλέξει ο παίκτης δυσκολία, τον ενημερώνει για το βάθος που επέλεξε και τον ρωτάει εάν θέλει να παίξει πρώτος στην περίπτωση που παίζει Human vs AI. Εάν επιλέξει y (ναι), τότε παίρνει τους μάυρους δίσκους και επιλέγει την πρώτη κίνηση, αλλίως παίζει δεύτερος με τους άσπρους δίσκους.

Στη συνέχεια, εμφανίζεται το αρχικό ταμπλό και το παιχνίδι ξεκινάει.

```
The game has started! This is the starting board:

    1 2 3 4 5 6 7 8
    - - - - - - - -
1 | . . . . . . . .
2 | . . . . . . . .
3 | . . . . . . . .
4 | . . . W B . . .
5 | . . . B W . . .
6 | . . . . . . . .
7 | . . . . . . . .
8 | . . . . . . . .
```

Διάρκεια Παιχνιδιού

Κατά τη διάρκεια του παιχνιδιού, παίζει ο παίκτης που έχει σειρά. Εάν είναι το AI, τότε κάνει την πρώτη κίνηση και εκτυπώνεται το καινούργιο board. Εάν είναι ο παίκτης, εισάγει την ΓΡΑΜΜΗ και την ΣΤΗΛΗ στις οποίες επιθυμεί να τοποθετήσει το δισκίο του, γίνεται έλεγχος εγκυρότητας, τοποθετείται το πιόνι, και γυρίζουμε όσα πιόνια «έπιασε» με αυτή την κίνηση. Ακολουθεί παράδειγμα Παίκτη (Μαύρο) vs AI (Άσπρο).

```
Black's Turn
Give row: 3
Give col: 4
Black's move: 3 4
    1 2 3 4 5 6 7 8
    - - - - - - - -
1 | . . . . . . . .
2 | . . . . . . . .
3 | . . . B . . . .
4 | . . . B B . . .
5 | . . . B W . . .
6 | . . . . . . . .
7 | . . . . . . . .
8 | . . . . . . . .
```

```
White's Turn
White's move: 3 3
    1 2 3 4 5 6 7 8
    - - - - - - - -
1 | . . . . . . . .
2 | . . . . . . . .
3 | . . W B . . . .
4 | . . . W B . . .
5 | . . . B W . . .
6 | . . . . . . . .
7 | . . . . . . . .
8 | . . . . . . . .
```

```
Black's Turn
Give row: 4
Give col: 3
Black's move: 4 3
    1 2 3 4 5 6 7 8
    - - - - - - - -
1 | . . . . . . . .
2 | . . . . . . . .
3 | . . W B . . . .
4 | . . B B B . . .
5 | . . . B W . . .
6 | . . . . . . . .
7 | . . . . . . . .
8 | . . . . . . . .
```

Ειδικές Περιπτώσεις

Το πρόγραμμα κάνει exception handling για όλα τα είδη input (από την αρχή έως και το τέλος) και συνεπώς δεν τερματίζει αντικανονικά ανεξάρτητα του πόσο λάθος κάνει ο χρήστης. Αντίθετα, του ζητάει να ξαναδοκιμάσει με ορθό input.

```
Black's Turn
Give row: a
Invalid input. Please try again.
```

Σε περίπτωση που ένας παίκτης δεν μπορεί να παίξει, η σειρά του παραλείπεται και ξαναπαίζει ο άλλος παίκτης. Το παιχνίδι δεν τερματίζει (εκτός και αν οι 2 δεν μπορούν να παίξουν).

```
Black's Turn
Give row: 7
Give col: 8
Black's move: 7 8
    1 2 3 4 5 6 7 8
    - - - - - - - -
1 | W W W W W W . .
2 | W W W W W W W .
3 | B W B B B W W B
4 | B B W B B W W B
5 | B W B B B B W B
6 | B W B W B B B B
7 | B W W B B W B B
8 | B W W W W W W B

White cannot play! Skipping turn...
```

```
White cannot play! Skipping turn...
    1 2 3 4 5 6 7 8
    - - - - - - - -
1 | W W W W W W . .
2 | W W W W W W W .
3 | B W B B B W W B
4 | B B W B B W W B
5 | B W B B B B W B
6 | B W B W B B B B
7 | B W W B B W B B
8 | B W W W W W W B

Black's Turn
```

Λήξη Παιχνιδιού

Το παιχνίδι τερματίζει είτε όταν το ταμπλό έχει γεμίσει η κανένας από τους δύο παίκτες δεν μπορούν να κάνουν κάποια κίνηση. Εμφανίζεται το τελικό σκορ και ανακοινώνεται ο νικητής (ή ισοπαλία.)

```
    1 2 3 4 5 6 7 8
    - - - - - - - -
1 | W B B B B B B B
2 | W B B B B W W W
3 | W B W W W B W W
4 | W B W W B B W B
5 | W W B B W W B B
6 | W W B W B B W B
7 | W W W W B W W B
8 | W W B B B B B B

Black wins!

The final score is:
Black: 33
White: 31
```

```
    1 2 3 4 5 6 7 8
    - - - - - - - -
1 | W W B B B B B W
2 | W W W W W B W W
3 | W W B W B W W W
4 | W B B B W W W W
5 | B B B B B B W W
6 | B B W B W B W W
7 | B B B W B W W W
8 | W B B B B B B B

Draw!

The final score is:
Black: 32
White: 32
```

Main Class

Η κύρια κλάση όπου αλληλεπιδρά ο χρήστης με το πρόγραμμα.

main: Καλεί τις παρακάτω συναρτήσεις εισόδου έτσι ώστε να γίνουν οι απαραίτητες αρχικοποιήσεις στο πρόγραμμα. Στη συνέχεια ξεκινάει το loop του παιχνιδιού, το οποίο τερματίζει ανν το board γεμίσει η δεν μπορεί να παίξει κανένας από τους δύο παίκτες. Διαχειρίζεται το παιχνίδι, εναλλάσοντας τις κινήσεις ανάλογα με το ποιος πάικτης έχει σειρά (και εάν μπορεί να παίξει). Όταν τελειώνει το παιχνίδι, εμφανίζει το σκορ και τον νικητή.

intro: Εκτυπώνει το εισαγωγικό μήνυμα του παιχνιδιού.

getGameMode: Επιτρέπει στον χρήστη να διαλέξει μεταξύ Human vs AI, AI vs AI (simulation) και Human vs Human. Κάνει τους απαραίτητους ελέγχους εγκυρότητας εισόδου.

playFirst: Ρωτάει τον χρήστη εάν επιθυμεί να παίξει πρώτος (τα μάυρα δισκία) η όχι.

getDepth: Ρωτάει τον χρήστη για το επιθυμητό του depth αναζήτησης του minimax.

Board Class

Η κλάση αυτή αντιπροσωπεύει το ταμπλό και είναι υπεύθυνη για την διαχείρισή του.

Variables: **int** W, B, E, lastPlayer, whiteDiscs, blackDiscs || **int[][]** gameboard ||

Move lastMove

Board (constructors): είναι οι κατασκευαστές της κλάσης. Αρχικοποίησή τις τιμές του παιχνιδιού και του ταμπλό σύμφωνα με τους κανόνες του παιχνιδιού. Συμπεριλαμβάνουμε copy constructor για την παραγωγή παιδιών καθώς και constructor που επιτρέπει την εισαγωγή ήδη-τρέχων παιχνιδιού για debugging και flexibility.

isValidMove: Ελέγχει αν οι συντεταγμένες που δόθηκαν σαν arguments αποτελούν μια έγκυρη κίνηση. Μια κίνηση θεωρείται έγκυρη αν :

Πρώτον, οι συντεταγμένες που δόθηκαν βρίσκονται εντός ορίων του board. Αυτό το ελέγχει καλώντας την inBounds.

Δεύτερον, στο σημείο με αυτές τις συντεταγμένες δεν υπάρχει ήδη τοποθετημένος δίσκος.

Τρίτον, ένας από του 8 γείτονες του σημείου που δόθηκε (με βάση τις συντεταγμένες) είναι του αντίθετου χρώματος. Αν βρεθεί γείτονας αντίθετου χρώματος, ψάχνει να βρει

ένα φιλικό δισκίο προς την κατεύθυνση του γείτονα ελέγχοντας ότι δεν μεσολαβούν κενά κελιά. Αν βρεθεί φιλικός δίσκος τότε η κίνηση είναι έγκυρη και επιστρέφει true αλλιώς false.

canPlay: Ελέγχει όλο τον πίνακα και επιστρέφει true εάν υπάρχει έστω και 1 δυνατή κίνηση για το χρώμα του argument. Χρησιμοποιεί την isValidMove, και επιστρέφει False εάν δεν βρει καμία κίνηση.

inBounds: Ελέγχει τις συντεταγμένες που δόθηκαν σαν arguments και ελέγχει εάν βρίσκεται εντός ορίων του board. Επιστρέφει True η False.

evaluate: Η ευρετική μας συνάρτηση κάνει διάφορους ελέγχους που αυξάνουν η μειώνουν την αξία για το κάθε χρώμα respectively. Μειώνει σε κινήσεις που συμφέρουν τον White και αυξάνει σε κινήσεις που συμφέρουν τον black. Όταν προσθέτουμε από τον πίνακα board, ανεξάρτητα ποιος παίζει, κάνει σωστή πράξη καθώς το βάρος πολλαπλασιάζεται με τον αριθμό που αντιστοιχεί στον παίχτη (1 για Black, -1 για White).

Αρχικά, ελέγχουμε ότι το board δεν είναι τελική κατάσταση. Εάν είναι, επιστρέφουμε ένα υπερβολικά υψηλό η χαμηλό σκόρ, ανάλογα ποιος νίκησε, η 0 εάν πρόκειται για ισοπαλία.

Στη συνέχεια, ελέγχεται εάν δεν μπορεί να παίξει κάποιος παίκτης από τους δύο και προσθαφαρεί ανάλογα σκόρ. Αυτό έχει αξία 1000 καθώς δίνει σημαντικό προβάδισμα.

Μετά, δίνουμε αξία 50 στις γωνίες, την μεγαλύτερη αξία από όλες τις θέσεις, καθώς αποτελούν στρατηγικά σημεία που μπορούν να παγιδεύσουν τον αντίπαλο.

Προσεχώς, δίνουμε βαρύτητα -20 (η +20 για τον Άσπρο) στα C και X squares, τα οποία είναι γειτονικά από τις γωνίες και μπορεί να δώσουν πάτημα στον αντίπαλο να τις κερδίσει.

Οι άκρες του πίνακα κερδίζουν λίγους πόντους (3 το καθένα) καθώς ενώ μπορεί να κάνουν την διαφορά, μερικές φορές δίνουν πάτημα στον αντίπαλο να πιάσουν πολλά δισκία μας μαζί.

Τέλος, εάν ο αντίπαλος έχει 1.5 φορες τα δισκία μας, αφαιρούμε 25 μονάδες από την αξία καθώς δεν θα κάνει την διαφορά, αλλα μερικές φορές μπορεί να αποτελεί ένδειξη ότι το παιχνίδι δεν πάει πολύ καλά.

placeDisk: Τοποθετεί ένα δισκίο του επιλεγμένου χρώματος στις συντεταγμένες που δόθηκαν. Στη συνέχεια αυξάνει το count για το ανάλογο χρώμα του δισκίου και γυρνάει όλα τα δισκία που κυριαρχήθηκαν από την κίνηση αυτή με την χρήση της flipDisks. Δεν κάνει έλεγχο εγκυρότητας στην κίνηση, υποθέτει πως έχει γίνει.

flipDisks: Κάνει traverse προς όλες τις κατευθύνσεις όπου υπάρχουν γειτονικά δισκία του αντιπάλου και ψάχνει να βρει ένα φιλικό δισκίο προς αυτή την κατεύθυνση. Εάν βρει, γυρνάει όλα τα δισκία. Εάν όχι, δεν κάνει τίποτα.

getChildren: Ελέγχει όλο το ταμπλό για έγκυρες (valid) κινήσεις του παίκτη με το τρέχον χρώμα. Για κάθε τέτοια έγκυρη κίνηση, την εκτελεί (καλώντας την placeDisk) σε αντίγραφο του τρέχοντος board. Επιστρέφει ArrayList που περιλαμβάνει όλα τα παραγόμενα αντικείμενα board.

print: Εκτυπώνει το current ταμπλό διαμορφωμένο έτσι ώστε να είναι κατανοητό από τον χρήστη.

printScore: Εκτυπώνει το current score του παιχνιδιού (αριθμό δισκίων ανα χρώμα).

printScore: Εκτυπώνει τον νικητή του παιχνιδιού (υπολογίζει τη διαφορά δισκίων).

Self-Explanatory Getters: getLastMove, getLastPlayer, getGameBoard

Self-Explanatory Setters: setLastMove, setLastPlayer

PlayerAB Class

Η κλάση αυτή αντιπροσωπεύει τον παίκτη και υπολογίζει τις κινήσεις του. Εάν είναι AI, χρησιμοποιεί τον minimax με a-b pruning (Max = Μάυρο, Min = Άσπρο), αλλιώς δέχεται input από το command line.

Variables: **int** color, maxDepth || Boolean isAi

PlayerAB (constructors): Ο default constructor δημιουργεί έναν AI player, άσπρου χρώματος με depth 2. Ιδανικά ο χρήστης θέτει αυτές τις τιμές βάσει input.

Move: Ανάλογα εάν ο παίκτης είναι AI, διαλέγει τον αλγόριθμο Minimax (isAi == True) ή την είσοδο από command line.

Minimax: Καλεί την max (μάυρο) η την min (άσπρο) ανάλογα το χρώμα του παίκτη.

Pickmove: Επιτρέπει στον χρήστη να εισάγει input μέσω του command line, κάνοντας τους απαραίτητους ελέγχους εγκυρότητας.

Max, min: Ανάλογα με το χρώμα του παίκτη, μία από τις δύο καλείται από τη Minimax, με ορίσματα το τρέχον ταμπλό στο παιχνίδι, βάθος (αρχικά 0), τιμή alpha (αρχικά -inf) και beta (αρχικά +inf).

Για κάθε παιδί του τρέχοντος ταμπλό, καλεί αναδρομικά την αντίθετή συνάρτηση με ορίσματα το παιδί ταμπλό, βάθος ίσο με το προηγούμενο +1, και τα προηγούμενα alpha και beta.

Αν το ταμπλό είναι γεμάτο ή το μέγιστο βάθος έχει φταστεί, επιστρέφει την τελευταία κίνηση του ταμπλό και την ευρετική της αξία.

Αν ένας παίκτης δεν μπορεί να παίξει στο τρέχον ταμπλό, καλείται η αντίθετη συνάρτηση με ίδια ταμπλό και alpha,beta και με βάθος ίσο με το προηγούμενο +1.

Ορίζει μία κίνηση bestMove (αρχικά αξίας -inf για τη max και +inf για την min). Για κάθε κίνηση που επιστρέφεται από την αναδρομική κλήση ελέγχει αν είναι καλύτερη από την bestMove (μεγαλύτερης αξίας για τη max και μικρότερης αξίας για τη min), και αν ναι, αντικαθιστά τη bestMove με την κίνηση.

Υλοποιεί a-b pruning:

Αν η αξία της bestMove στη max είναι μεγαλύτερη ή ίση με το beta, τότε η συνάρτηση σταματά και επιστέφει τη bestMove. Αν όχι, τότε στο alpha της max ανατίθεται το μέγιστο της τρέχουσας τιμής του και της αξίας του bestMove.

Αν η αξία της bestMove στη min είναι μικρότερη ή ίση με το alpha, τότε συνάρτηση σταματά και επιστέφει τη bestMove. Αν όχι, τότε στο beta της min ανατίθεται το μέγιστο της τρέχουσας τιμής του και της αξίας του bestMove.

isAi: Επιστρέφει εάν πρόκειται για AI παίκτη.

Move Class

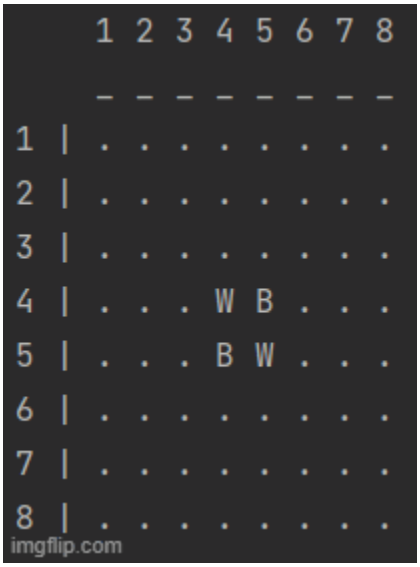
Η κλάση αυτή αντιπροσωπεύει είναι αυτούσια από το εργαστήριο και συνεπώς δεν χρειάζεται περαιτέρω εξήγηση πέρα από το γεγονός ότι διαχειρίζεται ένα αντικείμενο Move, το οποίο περιέχει συντεταγμένες και αξία (για χρήση στον minimax) και μία συνάρτηση ανάθεσης τιμών.

setMove: Θέτει όλες τις τιμές που μπορεί να έχει μια κίνηση (row, col, value) ταυτόχρονα.

Παραδείγματα Παιχνιδιών

ΣΕ ΠΕΡΙΠΤΩΣΗ ΠΟΥ ΔΕΝ ΛΕΙΤΟΥΡΓΟΥΝ ΤΑ GIF, ΤΑ ΕΧΟΥΜΕ ΒΑΛΕΙ ΣΕ ΦΑΚΕΛΟ ΜΕ ΤΟ ΑΝΑΛΟΓΟ ΟΝΟΜΑ ΤΟΥΣ

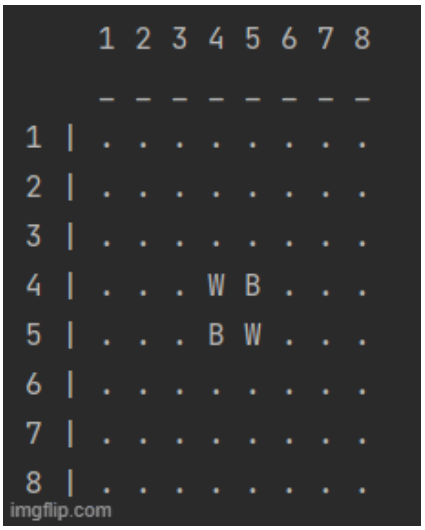
GAME 1 (D8): **Black** Ομάδα(3200283, 3200276, 3180269) *vs* **White** Εμείς




```
White wins!

The score is:
Black: 16
White: 48
```

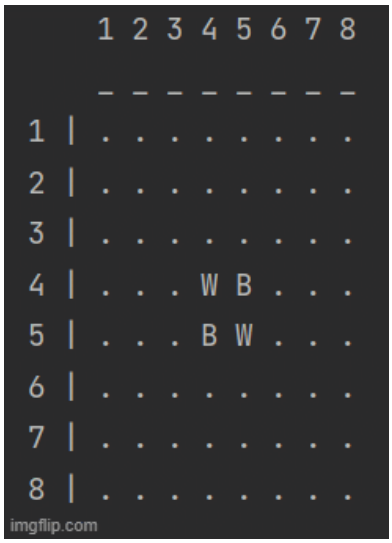
GAME 2 (D10): **Black** Εμείς *vs* **White** (3200283, 3200276, 3180269)

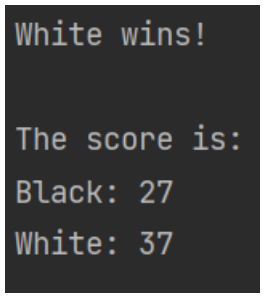



```
Black wins!

The score is:
Black: 60
White: 4
```

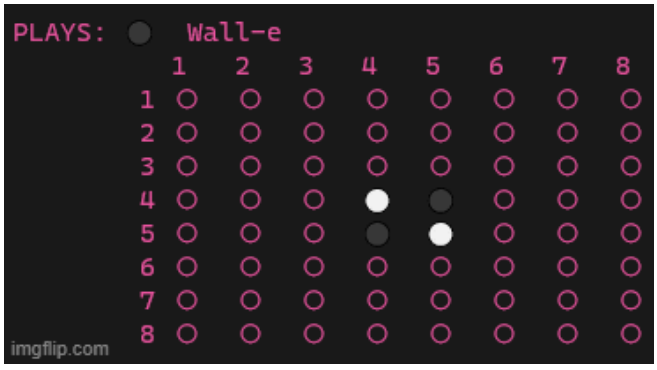
GAME 3 (D5): **Black** **Άνθρωπος** *vs* **White** AI





GAME 4 (D8): **Black** Ομάδα(3200150, 3200098) *vs* **White** Εμείς

NOTES: Το παιχνίδι δεν πήγαινε καθόλου καλά για εμάς, όμως λόγω των επιλογών της ευρετικής, ο παίκτης με τα μαύρα δισκία έχασε 3 φορές τη σειρά του στις 3 τελευταίες κινήσεις, και φτάσαμε σε ισοπαλία.



GAME 5 (D5): **Black** *vs* **White** Ομάδα(3200150, 3200098)