Authors:
Dimosthenis-Marios Karagiannis p3140069
Vasilis Loizidis p3200223

Othello(Reversi) Game

Player vs AI

(Using Mini-max algorithm)

Οικονομικό Πανεπιστήμιο Αθηνών 2022-2023

Περιγραφή

Η υλοποίηση της εφαρμογής μας πραγματοποιήθηκε με την χρήση της Java. Δημιουργήθηκαν 4 αρχεία που περιέχουν τις αντίστοιχες κλάσεις

1) Main.java
2) Board.java
3) Computer.java
4) Move.java

Περιγραφή Κλάσεων:

1) Main.java:

Αρχικα στην κλάση αυτή δίνουμε την επιλογή στον χρήστη να επιλέγει εάν θα παίξει πρώτος καθώς επίσης να επιλέξει το βάθος της αναζήτησης του αλγορίθμου MinMax (1 εως 10)

Βάσει της επιλογής του χρήστη δημιουργούμε ένα αντικείμενο τύπου Computer το οποίο θα κάνει τις κινήσεις του AI Player στο παιχνίδι μας και αρχικοποιούμε το game board με όνομα boarding.(Αντικείμενο τύπου Board)

Όλο το παιχνίδι περιλαμβάνεται σε ένα while loop το οποίο τερματίζει όταν φτάνουμε σε τελική κατάσταση βάσει των κανόνων του παιχνιδιού.

Σε κάθε κίνηση του παίχτη ελέγχεται το εάν υπάρχει διαθέσιμη κίνηση(αν δεν υπάρχουν, παίζει ο επόμενος παίκτης) καθώς επίσης καλείται και η μέθοδος isValidMove() της κλάσης Board για να γίνει ο έλεγχος ορθότητας της κίνησης.

Όταν παίζει ο Υπολογιστής καλείται η μέθοδος MiniMax() από το αντικείμενο computer όπου αποφασίζεται η επόμενη κίνηση του υπολογιστή με βάση τον υλοποιημένο αλγόριθμο Minimax ,στην κλάση Computer.

2) Board.java:

Αποτελείται από έναν πίνακα 8x8 .

Η κλάση αυτή δημιουργεί αντικείμενα τύπου Board. Κάθε αντικείμενο αντιστοιχεί σε ένα Game State και χρησιμοποιείται τόσο στην Main σαν ο κύριος πίνακας παιχνιδιού , όσο και στον αλγόριθμο minimax όπου δημιουργούνται διαφορετικά «Game States» - κινήσεων ένα εκ των οποίων επιλέγεται από τον αλγόριθμο για την επόμενη κίνηση του Υπολογιστή.

Επίσης με την μέθοδο isValidMove ελέγχεται η εγκυρότητα μίας κίνησης βάσει των κανόνων του παιχνιδιού

Αρχικά η μέθοδος αυτή ελέγχει εάν η κίνηση είναι εκτός ορίων του πίνακα

Κατά 2^{ον} ελέγχει εάν υπάρχει κάποιο πιόνι αντιπάλου στα γειτονικά 8 cells από την κίνηση που θέλουμε να ελέγξουμε με την checkNeighbor()η οποία παίρνει σαν ορίσματα row, col και τον παίκτη που κάνει την κίνηση (X η O).

Πρακτικά μέσω 2 εμφωλευμένων for ελέγχουμε όλες τις συντεταγμένες που γειτνιάζουν με τις row,col προσθέτοντας ή αφαιρώντας τους αριθμούς από -1 έως και 1 :

```
for(int i = -1 ; i <= 1 ; i++ ){
    for(int j = -1 ; j<=1 ; j++){
        if((row + i > 7) || (col + j> 7) || (row + i < 0) || (col + j < 0)){
             continue;}

        if(isEnemy(row+i,col+j,currentValue) ){
            if(checkExpandedCells(row+i,col+j,i,j,currentValue)){
                canPlace = true;
            }
        }
    }
}
```

Εάν βρεθεί αντίπαλο πιόνι καλείται η μέθοδος checkExpandedCells() με ορίσματα τις συντεταγμένες όπου βρέθηκε το εχθρικό πιόνι (row+ i , col +j) , και σαν rowOffset και colOffset παίρνει τα I, j έτσι ώστε να γνωρίζει την κατεύθυνση που θα ακολουθήσει για να βρει εάν υπάρχει κάποιο friendly cell :

```
while(isEnemy(checkRow,checkCol,currentValue)==true){

  if((checkCol+colOffset)>7||(checkRow+rowOffset)>7 ||(checkCol+colOffset)<0
||(checkRow+rowOffset)<0  ){break;}

  if(isFriendly(checkRow+rowOffset,checkCol+colOffset,currentValue)==true){
    canPut = true;
    flip(row,col,checkRow+rowOffset,checkCol+colOffset,rowOffset,colOffset,
    currentValue);
    }
    checkRow = checkRow + rowOffset;
    checkCol = checkCol + colOffset;

}
```

Εάν βρεθεί friendly cell τότε καλείται η μέθοδος flip για να αντιστρέψουμε τα πιόνια μεταξύ της επιλεγμένης κίνησης και του friendly cell που βρήκαμε. Η flip παίρνει σαν ορίσματα το row και το col που βρέθηκε το πρώτο εχθρικό cell , τα checkRow και τα checkColumn τα οποία αντιστοιχούν στο σημείο που θα σταματήσει να γίνεται η αντιμεταστροφη των πιονιών καθώς και την κατεύθυνση που ακολουθεί για να αντιστρέψει τα πιόνια (ως rowOffset και colOffset) :

```
void flip(int rowStart, int colStart , int rowFinish , int colFinish ,int
rowOffset,int colOffset,int currentValue){

    int currentRow = rowStart;
    int currentCol = colStart;

    while(true){

        this.gameBoard[currentRow][currentCol] = currentValue;//
        currentRow += rowOffset;
        currentCol += colOffset;
        if(currentRow == rowFinish && currentCol == colFinish)break;
    }
}
```

Επίσης χρησιμοποιήσαμε την ευρετική συνάρτηση $h(n) = f1(n) + 3 \times f2(n) + 2 \times f3(n)$, όπου n ο κόμβος του δέντρου αναζήτησης στον οποίο εφαρμόζεται η ευρετική οπου:

- f 1 (n) είναι το συνολικό πλήθος των μαύρων πιονιών μείον το συνολικό πλήθος των άσπρων πιονιών στον κόμβο n.
- f 2 (n) είναι το συνολικό πλήθος των μαύρων πιονιών που βρίσκονται σε γωνίες μείον το συνολικό πλήθος των άσπρων που βρίσκονται σε γωνίες στον κόμβο n.
- f 3 (n) είναι το συνολικό πλήθος των μαύρων πιονιών που βρίσκονται σε μη γωνιακές ακραίες θέσεις μείον το συνολικό πλήθος των άσπρων που βρίσκονται σε μη γωνιακές ακραίες θέσεις στον n.

Την ευρετική συνάρτηση την υλοποιήσαμε μέσα στην μέθοδο evaluate()

Τέλος δημιουργήσαμε μια μέθοδο availableMoves() η οποία επιστρέφει true αν υπάρχει διαθέσιμη κίνηση για τον παίκτη που παίζει χωρίς να παραβιάσει τους κανόνες

3) Computer.java

Στην class Computer υλοποιούμε τον αλγόριθμο MiniMax . Για την υλοποίηση του αλγορίθμου αυτού χρησιμοποιήσαμε αναδρομική αναζήτηση Depth-First (DFS) , και πριόνισμα A-B καθώς ο αλγόριθμος αναζήτησης γινόταν πολύ αργός σε μεγάλο βάθος αναζήτησης.

Κάθε αντικείμενο Computer δέχεται σαν όρισμα στον Default Consturctor το μέγιστο βάθος αναζήτησης, καθώς επίσης το compLetter που το πληροφορεί για το "γράμμα" με το οποίο παίζει ο υπολογιστής.

Η μέθοδος MiniMax παίρνει σαν όρισμα ένα αντικείμενο τύπου board και καλεί την max() εάν ο Υπολογιστής παίζει σαν X ή την min() εάν ο Υπολογιστής παίζει σαν O. (Έχουμε υπόψιν ότι ο X παίζει πρώτος και ο O 2ος) .

```
Move MiniMax(Board board)
{
    if(compLetter == Board.X)
    {
        return max(new Board(board), 0 , Integer.MIN_VALUE ,
Integer.MAX_VALUE);
    }
    else
    {
        return min(new Board(board), 0,Integer.MIN_VALUE , Integer.MAX_VALUE
);
    }
}
```

Η μέθοδος max όπως και η min αρχικά ελέγχει εάν ένα state είναι terminal node, δηλαδή εάν κάποιος παίκτης έχει νικήσει , είτε εάν έχουμε φτάσει το μέγιστο βάθος αναζήτησης. Εάν ένα από τα δύο παραπάνω statements είναι αληθές τότε επιστρέφει ένα νέο αντικείμενο move στο οποίο καλείται και η μέθοδος evaluate.

Πρακτικά αυτό σημαίνει ότι όταν το υλοποιημένο δέντρο από το backtracked DFS φτάσει στην δημιουργία ενός φύλλου του , το αξιολογεί με βάση την ευρετική και το επιστρέφει αναδρομικά.

```
if(board.isTerminal() || (depth == this.maxDepth))
{
    return new Move(board.getLastMove().getRow(),
board.getLastMove().getCol(), board.evaluate());
}
```

Εάν δεν είμαστε στο τέλος της αναδρομής, δημιουργείται ένα ArrayList τύπου board με τα παιδιά του τωρινού node.

Για κάθε παιδί(εάν αντίστοιχα δεν είναι φύλλο) καλείται η μέθοδος min , σε παραπάνω βάθος ,η οποία είναι καθ' ομοίωσίν της μεθόδου max και δημιουργούνται τα παιδιά του οπότε είτε τερματίζουμε στα φύλλα του δέντρου είτε καλείτε εκ νέου η μέθοδος max για κάθε παιδί του min. Με αυτό τον τρόπο δημιουργείται το δέντρο μας μέσω της ανάδρομης .

Έπειτα επιλέγουμε και επιστρέφουμε το (Max/Min)-move με την μεγαλύτερη αξία(στην max) ή με την μικρότερή αξία(στην min).

Εάν υπάρχουν ίσα values επιλέγουμε τυχαία μια από τις δυο κινήσεις με το random generator

Στην συνέχεια ελέγχουμε εάν μπορούμε να πριονίσουμε το δέντρο μας. Πχ:

Στην max αν η αξία του maxMove είναι μεγαλύτερη ή ίση από την beta τότε δεν χρειάζεται να αναπτύξουμε τα υπόλοιπα παιδιά του κόμβου αφού δεν θα αλλάξει η αξία του κόμβου οπότε "σπάει" η επανάληψη και πριονίζεται το δέντρο, εάν όχι τότε κάνουμε update τον alpha και συνεχίζουμε

```
if(maxMove.getValue()>=beta){
    break;
}
alpha = Integer.max(alpha , maxMove.getValue());
```

Με παρόμοιο τρόπο δουλεύει η μέθοδος min , σε αντιστοιχία με την max, και το πριόνισμα γίνεται με alpha

Παράδειγμα εκτέλεσης παιχνιδιού

```
Type 1 if you want to play first and be X or 2 if you want to play second and be O
2
Type max depth for the MiniMax algorithm(1-10)
10
|*|1|2|3|4|5|6|7|8|||
|1|- - - - - - - - ||
|2|- - - - - - - - ||
|3|- - - - - - - - ||
|4|- - - O X - - - ||
|5|- - - X O - - - ||
|6|- - - - - - - - ||
|7|- - - - - - - - ||
|8|- - - - - - - - ||
**********
computer Plays as X
|*|1|2|3|4|5|6|7|8|||
|1|- - - - - - - - ||
|2|- - - - - - - - ||
|3|- - - - - - - - ||
|4|- - - O X - - - ||
|5|- - - X X X - - ||
|6|- - - - - - - - ||
|7|- - - - - - - - ||
|8|- - - - - - - - ||
**********
Computer play as X: (5,6)
Score: Computer = 4|| Player = 1
Player O please
enter row:
4
enter column:
6
|*|1|2|3|4|5|6|7|8|||
|1|- - - - - - - - ||
|2|- - - - - - - - ||
|3|- - - - - - - - ||
|4|- - - O O O - - ||
|5|- - - X X X - - ||
|6|- - - - - - - - ||
|7|- - - - - - - - ||
|8|- - - - - - - - ||
**********
Score: Computer = 3 || Player = 3
computer Plays as X
|*|1|2|3|4|5|6|7|8|||
|1|- - - - - - - - ||
|2|- - - - - - - - ||
|3|- - - - - - X - ||
|4|- - - O O X - - ||
|5|- - - X X X - - ||
|6|- - - - - - - - ||
|7|- - - - - - - - ||
|8|- - - - - - - - ||
**********
Computer play as X: (3,7)
Score: Computer = 5|| Player = 2
Player O please
enter row:
6
enter column:
3
|*|1|2|3|4|5|6|7|8|||
|1|- - - - - - - - ||
|2|- - - - - - - - ||
|3|- - - - - - X - ||
|4|- - - O O X - - ||
|5|- - - O X X - - ||
|6|- - O - - - - - ||
|7|- - - - - - - - ||
|8|- - - - - - - - ||
**********
Score: Computer = 4 || Player = 4
computer Plays as X
|*|1|2|3|4|5|6|7|8|||
|1|- - - - - - - - ||
|2|- - - - - - - - ||
|3|- - - - - - X - ||
|4|- - X X X X - - ||
|5|- - - O X X - - ||
|6|- - O - - - - - ||
|7|- - - - - - - - ||
|8|- - - - - - - - ||
**********
Computer play as X: (4,3)
Score: Computer = 7|| Player = 2
Player O please
enter row:
5
enter column:
7
|*|1|2|3|4|5|6|7|8|||
|1|- - - - - - - - ||
|2|- - - - - - - - ||
|3|- - - - - - X - ||
|4|- - X X X X - - ||
|5|- - - O O O O - ||
|6|- - O - - - - - ||
|7|- - - - - - - - ||
|8|- - - - - - - - ||
**********
Score: Computer = 5 || Player = 5
computer Plays as X
|*|1|2|3|4|5|6|7|8|||
|1|- - - - - - - - ||
|2|- - - - - - - - ||
|3|- - - - - - X - ||
|4|- - X X X X - - ||
|5|- - - O O X O - ||
|6|- - O - - - X - ||
|7|- - - - - - - - ||
|8|- - - - - - - - ||
**********
Computer play as X: (6,7)
Score: Computer = 7|| Player = 4
```

.....

```
Player O please
enter row:
2
enter column:
5
|*|1|2|3|4|5|6|7|8|||
|1|- X X X X X X X ||
|2|- - X O O O - X ||
|3|- - O O O O X X ||
|4|- O O O X X O X ||
|5|- - O O O X X X ||
|6|- - O O O O X X ||
|7|- - - - - - O X ||
|8|- - - - - - - X ||
**********************
Score: Computer = 21 || Player = 19
computer Plays as X
|*|1|2|3|4|5|6|7|8|||
|1|- X X X X X X X ||
|2|- - X X X X X X ||
|3|- - O O O X X X ||
|4|- O O O X X O X ||
|5|- - O O O X X X ||
|6|- - O O O O X X ||
|7|- - - - - - O X ||
|8|- - - - - - - X ||
**********************
Computer play as X: (2,7)
Score: Computer = 26|| Player = 15
No available moves for O. Turn skipped
computer Plays as X
|*|1|2|3|4|5|6|7|8|||
|1|- X X X X X X X ||
|2|- - X X X X X X ||
|3|- X X X X X X X ||
|4|- O O O X X O X ||
|5|- - O O O X X X ||
|6|- - O O O O X X ||
|7|- - - - - - O X ||
|8|- - - - - - - X ||
**********************
Computer play as X: (3,2)
Score: Computer = 30|| Player = 12
Player O please
enter row:
2
enter column:
1
|*|1|2|3|4|5|6|7|8|||
|1|- X X X X X X X ||
|2|O - X X X X X X ||
|3|- O X X X X X X ||
|4|- O O O X X O X ||
|5|- - O O O X X X ||
|6|- - O O O O X X ||
|7|- - - - - - O X ||
|8|- - - - - - - X ||
**********************
Score: Computer = 29 || Player = 14
```

.....

```
Score: Computer = 54 || Player = 7
computer Plays as X
|*|1|2|3|4|5|6|7|8|||
|1|X X X X X X X X ||
|2|X X X X X X X X ||
|3|X X X X X X X X ||
|4|X X X X X X O X ||
|5|X X X X X O X X ||
|6|X X X X O O X X ||
|7|X - X O X X X X ||
|8|- X X X X X X X ||
**********************
Computer play as X: (8,2)
Score: Computer = 57|| Player = 5
Player O please
enter row:
7
enter column:
2
|*|1|2|3|4|5|6|7|8|||
|1|X X X X X X X X ||
|2|X X X X X X X X ||
|3|X X X X X X X X ||
|4|X X X X X X O X ||
|5|X X X X X O X X ||
|6|X X X X O O X X ||
|7|X O O O X X X X ||
|8|- X X X X X X X ||
**********************
Score: Computer = 56 || Player = 7
computer Plays as X
|*|1|2|3|4|5|6|7|8|||
|1|X X X X X X X X ||
|2|X X X X X X X X ||
|3|X X X X X X X X ||
|4|X X X X X X O X ||
|5|X X X X X O X X ||
|6|X X X X O O X X ||
|7|X X O O X X X X ||
|8|X X X X X X X X ||
**********************
Computer play as X: (8,1)
Score: Computer = 58|| Player = 6

Game Over
Computer Wins :(
```