Περιεχόμενα:

Τρόπος Χρήσης

Εκτέλεση:
game.py [-s SIZE]

Πηγαίνετε στο φάκελο που βρίσκεται το αρχείο game.py
Εκτελείτε το πρόγραμμα: python game.py
Θα σας ζητηθεί
“Players turn. Enter 1 if you want to go first, 2 to go second: "
και μετά
"Enter the maximum search depth for the MinMax algorithm: "
δίνετε τις τιμές που θέλετε και το παιχνίδι θα ξεκινήσει.

Μπορείτε να εκτελέσετε το πρόγραμμα με την παράμετρο -s από τη γραμμή εντολών,
η οποία καθορίζει το μέγεθος του ταμπλό. Π.χ. python game.py -s 4
Η προεπιλεγμένη τιμή είναι 8.
Για το προεπιλεγμένο μέγεθος ταμπλό, μετά το βάθος 8 το πρόγραμμα γίνεται αρκετά αργό.

Περιγραφή

Αρχική κατάσταση:
Εμφανίζεται ο πίνακας (*board*) του παιχνιδιού
Εκτέλεση κινήσεων:
Αν είναι η σειρά του υπολογιστή, εκτελείται η κίνησή του αυτόματα.
Αν είναι η σειρά του χρήστη, ο χρήστης καλείται να εισάγει την κίνησή του.

> Εάν η κίνηση είναι έγκυρη, πραγματοποιείται στον πίνακα.
> Εάν η κίνηση είναι μη έγκυρη, το πρόγραμμα ενημερώνει τον χρήστη για τη μη εγκυρότητα και ζητά να εισαχθεί μια νέα κίνηση.

Έλλειψη έγκυρων κινήσεων:
Εάν κάποιος παίκτης (χρήστης ή AI) δεν έχει καμία δυνατή έγκυρη κίνηση, το πρόγραμμα το ανακοινώνει και η σειρά έρχεται η σειρά του αντιπάλου.
Τερματισμός Παιχνιδιού:
Όταν το παιχνίδι φτάσει σε τελική κατάσταση, το πρόγραμμα:
Ανακοινώνει το τελικό σκορ.
Ανακοινώνει το νικητή.
Υποδεικνύει ποια μέθοδος απόφασης κινήσεων (π.χ. *human* ή *minimax*) νίκησε.

Ενδεικτική παρτίδα σε 4x4

```
C:\Users\odimi\Desktop\Σχολη\Τεχνητή Νοημοσύνη>py game.py -s 4
Players turn. Enter 1 if you want to go first, 2 to go second: 2
Enter the maximum search depth for the minmax algorithm: 5
  . 0 . 1 . 2 . 3 .
0 |   |   |   |   |
1 |   | X | O |   |
2 |   | O | X |   |
3 |   |   |   |   |

its X turn
  . 0 . 1 . 2 . 3 .
0 |   |   | X |   |
1 |   | X | X |   |
2 |   | O | X |   |
3 |   |   |   |   |

its O turn
Enter your move O (column and row separated by space): 3 2
  . 0 . 1 . 2 . 3 .
0 |   |   | X |   |
1 |   | X | X |   |
2 |   | O | O | O |
3 |   |   |   |   |

its X turn
  . 0 . 1 . 2 . 3 .
0 |   |   | X |   |
1 |   | X | X |   |
2 |   | X | O | O |
3 | X |   |   |   |

its O turn
Enter your move O (column and row separated by space): 0 2
  . 0 . 1 . 2 . 3 .
0 |   |   | X |   |
1 |   | X | X |   |
2 | O | O | O | O |
3 | X |   |   |   |

its X turn
  . 0 . 1 . 2 . 3 .
0 |   |   | X |   |
1 |   | X | X |   |
2 | O | X | O | O |
3 | X | X |   |   |

its O turn
Enter your move O (column and row separated by space): 0 0
  . 0 . 1 . 2 . 3 .
0 | O |   | X |   |
1 |   | O | X |   |
2 | O | X | O | O |
3 | X | X |   |   |

its X turn
  . 0 . 1 . 2 . 3 .
0 | O |   | X |   |
1 | X | X | X |   |
2 | X | X | O | O |
3 | X | X |   |   |

its O turn
3 | X | X |   |   |

its O turn
Enter your move O (column and row separated by space): 1 0
  . 0 . 1 . 2 . 3 .
0 | O | O | X |   |
1 | X | X | O |   |
2 | X | X | O | O |
3 | X | X |   |   |

its X turn
  . 0 . 1 . 2 . 3 .
0 | O | O | X | X |
1 | X | X | X |   |
2 | X | X | O | O |
3 | X | X |   |   |

its O turn
No valid moves for O
  . 0 . 1 . 2 . 3 .
0 | O | O | X | X |
1 | X | X | X |   |
2 | X | X | O | O |
3 | X | X |   |   |

its X turn
  . 0 . 1 . 2 . 3 .
0 | O | O | X | X |
1 | X | X | X | X |
2 | X | X | X | O |
3 | X | X |   |   |

its O turn
Enter your move O (column and row separated by space): 3 3
  . 0 . 1 . 2 . 3 .
0 | O | O | X | X |
1 | X | O | X | X |
2 | X | X | O | O |
3 | X | X |   | O |

its X turn
  . 0 . 1 . 2 . 3 .
0 | O | O | X | X |
1 | X | O | X | X |
2 | X | X | X | O |
3 | X | X | X | O |
Game Over!
 Score  X:11 - O:5
 X min_max Won Over O human

C:\Users\odimi\Desktop\Σχολη\Τεχνητή Νοημοσύνη>
```

.

Αρχιτεκτονική :

Game

Η κύρια λογική του παιχνιδιού υλοποιείται στο αρχείο game.py.
Ορίζεται ένα board και δύο players που πραγματοποιούν κινήσεις πάνω σε αυτό το board.
Οι κινήσεις των players αποφασίζονται από τη μέθοδο που δίνεται ως όρισμα κατά την κατασκευή τους. Η μέθοδος μπορεί να είναι η εισαγωγή από τον χρήστη ή κάποιος αλγόριθμος.
Οι παράμετροι depth (βάθος) και eval (συνάρτηση αξιολόγησης) έχουν νόημα όταν η μέθοδος είναι ο αλγόριθμος minmax.
game(method1, method2, depth1, depth2, size, eval1, eval2, isTest)
methos1, deth1, eval1 Η μέθοδος απόφασης κίνησης, το μέγιστο βάθος αναζήτησης και η ευρετική συνάρτηση του player1. Αντίστοιχα για το 2.
size το μέγεθος του ταμπλό και isTest αν είναι παρτίδα ή test.
Αυτή τη στιγμή είναι προ αποφασισμένος ο αλγόριθμος minımax με πριόνισμα α – β για τον ένα παίκτη και εισαγωγή από το χρήστη για τον άλλο.

Player

getMove: Η συνάρτηση που δίνει την κίνηση του παίκτη, μπορεί να είναι AI ή user input.
minımax: Επιλογή κίνησης με αλγόριθμο minimax με πριόνισμα α – β
humanInput: Επιλογή κίνησης από την είσοδο του χρήστη
randomMove, instaMax, instaMaxRandom, firstone : Άλλες συναρτήσεις επιλογής κινήσεων
(var) evaluate: Μεταβλητή που της δίνεται το όνομα της ευρετικής συνάρτησης. Προσδιορίζεται από το αντίστοιχο όρισμα του κατασκευαστή. Χρησιμοποιείται για την κλήση της εκάστοτε ευρετικής συνάρτησης
(var) method: Μεταβλητή που της δίνεται η συνάρτηση του αλγόριθμου επιλογής κίνησης, προσδιορίζεται από το αντίστοιχο όρισμα του κατασκευαστή.

Board

(Μερικές συναρτήσεις ορίστηκαν έξω από την κλάση για να μπορούν να χρησιμοποιηθούν και με άλλο τρόπο, αλλά νοηματικά ανήκουν σε αυτή την κλάση)

Moves

toTurn(x, y, player, table, size) # βρίσκει ποια πιόνια είναι προς αλλαγή χρώματος αν πραγματοποιηθεί κίνηση στο x, y από τον player
is_valid_move(player, row, col)
has_valid_move(player)
allPossibleMoves(player)
otherSymbol(symbol) # Επιστρέφει το σύμβολο του άλλου παίκτη
placeRaw(x, y) # τοποθετεί πιόνι και αναστρέφει όσα πρέπει να αναστραφούν
nextMove() # επόμενη κίνηση στην παρτίδα

Evaluations
evaluate(from_players_perspective, isKnownTerminal=False) # Απλή ευρετική
evaluate2(from_players_perspective, endNode) # Σύνθετη ευρετική
is_game_over()
isTerminal()
calculateScore()
getResult()

Children
generateChildFromMove(x, y) # Παράγεται η κατάσταση παιδί του board που αντιστοιχεί στην πραγματοποίηση της κίνησης στο x, y
generateEmptyMove() # Οταν δεν υπάρχει έγκυρη κίνηση για τον παίκτη παράγεται αυτή η ψευδο- κίνηση, δηλαδή απλά αλλάζει η σειρά του παίκτη, για να μην επηρεαστεί η λειτουργία του mini max
generateChildrenFromMoves(possibleMoves) # Παράγονται οι καταστάσεις παιδιά του board από όλες τις δυνατές κινήσεις

Presentation
print_possible_moves(moves, table)
printResult()
print_table(table, size)

Μέθοδος τεχνητής νοημοσύνης:
Η μέθοδος Τεχνητής Νοημοσύνης που χρησιμοποιείται είναι ο αλγόριθμος **minimax** με **Alpha-Beta pruning**:

minimax: Ο αλγόριθμος αυτός αναζητά τη βέλτιστη κίνηση για έναν παίκτη, υποθέτοντας ότι ο αντίπαλος θα παίξει επίσης με τον τρόπο που ο αλγόριθμος υπολογίζει ως βέλτιστο για εκείνον. Η αναζήτηση πραγματοποιείται αναδρομικά, εξετάζοντας όλες τις πιθανές κινήσεις (εκτός όσων κόβονται από το α – β πριόνισμα αν αυτό υπάρχει) μέχρι να φτάσει σε τελικό αποτέλεσμα ή στο μέγιστο επιθυμητό βάθος αναζήτησης. Η αξιολόγηση κάθε κατάστασης γίνεται με βάση την επιλεγμένη ευρετική συνάρτηση.
Ο αλγόριθμος βασίζεται στην παραγωγή νέων στιγμιότυπων παιδιών της κλάσης Board και την αξιολόγησή τους.

Alpha - Beta pruning: Εφαρμόζεται στον minimax με στόχο τη βελτίωση της χρονικής απόδοσης του αλγορίθμου, παραλείποντας την εξερεύνηση των κλαδιών που δεν επηρεάζουν την τελική απόφαση.

Υλοποιήθηκαν δύο ευρετικές συναρτήσεις.
Η **απλή ευρετική** υπολογίζει το συνολικό πλήθος των πιονιών του παίκτη μείον το συνολικό πλήθος των πιονιών του αντιπάλου. Αν είναι τελική κατάσταση επιστρέφει 1000 για νίκη, -1000 για ήτα, 0 για ισοπαλία.

Η **σύνθετη ευρετική** $h(n) = f1(n) + 3 \times f2(n) + 2 \times f3(n)$
όπου
f1(n) είναι το συνολικό πλήθος των πιονιών του παίκτη μείον το συνολικό πλήθος των πιονιών του αντιπάλου.

f2(n) είναι το συνολικό πλήθος των πιονιών του παίκτη που βρίσκονται σε γωνίες μείον το συνολικό πλήθος των αντίπαλων πιονιών που βρίσκονται σε γωνίες.
f3(n) είναι το συνολικό πλήθος των πιονιών του παίκτη που βρίσκονται σε μη γωνιακές ακραίες θέσεις μείον το συνολικό πλήθος των αντίπαλων πιονιών που βρίσκονται σε μη γωνιακές ακραίες θέσεις.
Αν είναι τελική κατάσταση επιστρέφει 1000 για νίκη, -1000 για ήτα, 0 για ισοπαλία.

Πειραματικά Δεδομένα

testing.py
Με τη βοήθεια της μεθόδου test ο minimax ανταγωνίστηκε διαφορετικούς αλγόριθμους και με τις δύο διαφορετικές ευρετικές.
Αλγόριθμος random: Επιλέγει μια έγκυρη κίνηση στην τύχη.
Αλγόριθμος instaMaxRandom: Βρίσκει τις αμέσως επόμενες κινήσεις με την βέλτιστη αξιολόγηση βάση της ευρετικής και επιλέγει μια από αυτές τυχαία.
Στους παρακάτω πίνακες:
depth είναι το βάθος του minimax.
turn το αν ο minmax παίζει πρώτος
time ο κατά μέσο όρο χρόνος ολοκλήρωσης ενός παιχνιδιού
games ο αριθμός των παιχνιδιών που πραγματοποιήθηκαν
Το % αντιπροσωπεύει την πιθανότητα θετικού αποτελέσματος για τον minmax αλγόριθμο.
Υπολογίστηκε ως (wins + ties/2)/games

random

simple heuristic

depth	turn	wins	loses	ties	games	time (m/o)	%
1	1	334	153	13	500	0.1	0.681
1	2	262	218	20	500	0.1	0.544
2	1	189	54	7	250	0.3	0.77
2	2	175	68	7	250	0.4	0.714
3	1	43	7	0	50	1.1	0.86
3	2	43	5	2	50	1.6	0.88
4	1	21	4	0	25	4.8	0.84
4	2	19	5	1	25	5.9	0.78
5	1	5	0	0	5	19.9	1
5	2	5	0	0	5	24.5	1

complex heuristic

depth	turn	wins	loses	ties	games	time (m/o)	%
1	1	369	125	6	500	0.1	0.744
1	2	335	148	17	500	0.1	0.687
2	1	212	35	3	250	0.4	0.854

2	2	200	45	5	250	0.5	0.81
3	1	45	5	0	50	1.5	0.9
3	2	43	5	2	50	2	0.88
4	1	25	0	0	25	5.7	1
4	2	23	2	0	25	8.8	0.92
5	1	5	0	0	5	35.7	1
5	2	5	0	0	5	37.7	1

instaMaxRandom

simple heuristic

depth	turn	wins	loses	ties	games	time (m/o)	%
1	1	284	198	18	500	0.1	0.586
1	2	204	284	12	500	0.1	0.42
2	1	182	65	3	250	0.5	0.734
2	2	174	72	4	250	0.5	0.704
3	1	35	12	3	50	1.9	0.73
3	2	39	11	0	50	2.1	0.78
4	1	21	4	0	25	6.8	0.84
4	2	21	3	1	25	6	0.86
5	1	4	1	0	5	25.2	0.8
5	2	5	0	0	5	34.8	1

complex heuristic

depth	turn	wins	loses	ties	games	time (m/o)	%
1	1	170	316	14	500	0.2	0.354
1	2	298	190	12	500	0.2	0.608
2	1	194	53	3	250	0.5	0.782
2	2	199	43	8	250	0.6	0.812
3	1	49	0	1	50	2.1	0.99
3	2	43	7	0	50	2.6	0.86
4	1	25	0	0	25	6.7	1
4	2	25	0	0	25	9.6	1
5	1	5	5	0	5	30.9	1
5	2	5	5	0	5	56.6	1

Συμπεράσματα

Ο minimax αλγόριθμος είναι προτιμότερος από την τυχαία επιλογή κινήσεων
Ο minimax αλγόριθμος είναι προτιμότερος από το να επιλέγουμε την κίνηση που αξιολογούμε αμέσως ως καλύτερη (τουλάχιστον με βάση τις δύο ευρετικές μας).
Ο minimax αλγόριθμος είναι καλύτερος από έναν αρχάριο παίκτη (εμένα).
Ο minimax αλγόριθμος γίνεται καλύτερος όσο αυξάνεται το βάθος, αλλά επίσης αυξάνεται σημαντικά και ο χρόνος που χρειάζεται για να επιλέξει μια κίνηση.
Η σύνθετη ευρετική είναι καλύτερη από την απλή ευρετική.
Όλα τα συμπεράσματα ήταν αναμενόμενα και επιβεβαιώνουν τη θεωρία.

Το τελικό συμπέρασμα που απορρέει από τα πειραματικά δεδομένα είναι ότι η συγκεκριμένη υλοποίηση είναι ικανοποιητική επιλογή για ένα απλό παιχνίδι reversi.