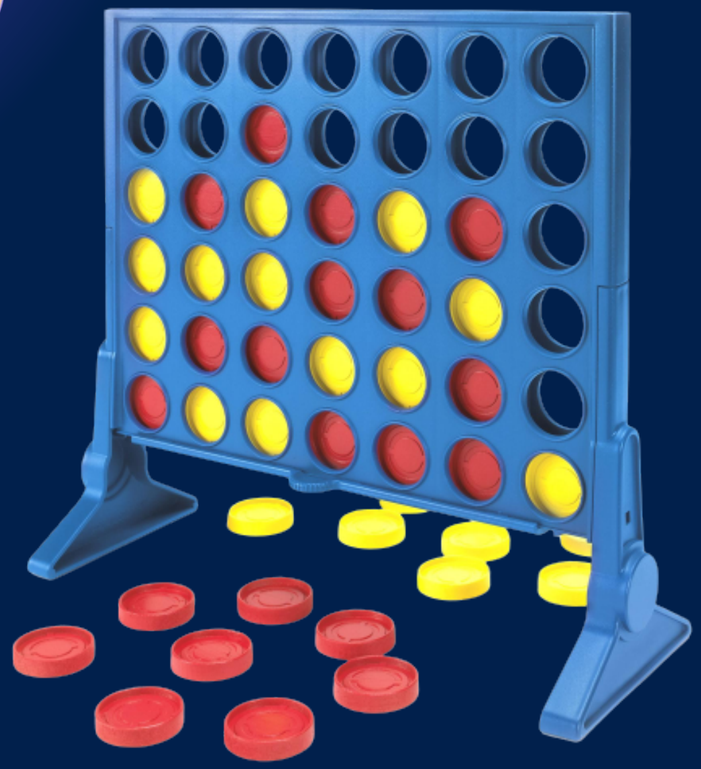


ARTIFICAL INTELLIGENCE

ASSIGNMENT

4Connect

Rohit Kodam
2020B3A71141G

https://github.com/Rohitkk432/Connect4AI

P2 Logic

- I implemented my own logic to make the P2 logic improving it iteratively.
- You can check my working on it on https://github.com/Rohitkk432/Connect4AI
- MoveFinder creates a dummy 4connect object for every action and checks if P2 wins.
- If doesnt then P1 move is played by MyopicPlayerAction and check if P1 wins.
- if doesnt then we recall the MoveFinder function to check the same in lower depth of the graph/tree.
- EvaluateMoves function gives which move is best to play.
- Basically we are recursively finding best moves.
- It is not exactly like Minimax.

Rewards

- At each node i am calculating reward for each move(0-6)
- And then passing it to evaluation function to find the best move in that node.

```
# Rewards keeps track of rewards
on each action at each max node
    win = 1
    draw/no winner yet = 0
    lose = -1
    invalid action = -2
'''
rewards=[]
```

Evaluation Function

- Basically i am giving priorities as -
- 1 - Win now
- 2 - Block P1 Win
- 3 - Win in next moves take lowest depth win action
- 4 - Actions which can win eventually not depth based just reward based.
- 5 - Draw / no win moves
- 6 - Loss Moves

```
#priority1 - win if possible now
        if len(winNow)>0:
            rewardBest=1
            bestMove=random.choice(winNow)
        #priority2 - block p1 if he wins now
        elif len(p1WinCols)>0:
            bestMove=random.choice(p1WinCols)
            rewardBest=rewards[bestMove]
        #priority3 - p2 wins later based on on min win Depth
        elif len(minDepthArr)>0 and minim<5:
            bestMove=random.choice(minDepthArr)
            rewardBest=1
        #priority4 - win if possible in next moves
        elif len(winIndex)>0:
            rewardBest=1
            bestMove=random.choice(winIndex)
        #priority5 - draw/no winner yet moves
        elif len(dkIndex)>0:
            rewardBest=0
            bestMove=random.choice(dkIndex)
        #priority6 - no choice and we lose in next moves/now
        elif len(lossIndex)>0:
            rewardBest=-1
            bestMove=random.choice(lossIndex)
        #no moves possible
        else:
            rewardBest=-2
            bestMove=0
        return bestMove,rewardBest
```

P2 Logic Code

- Just a Snippet of code inside MoveFinder function.
- This part Simulates the game to create GameTree
- Here i would like to address that i am just working on Maximizing player that is P2
- And for P1 i am just running MyopicPlayerAction

```python
for action in range(7):
    #creating game from current state
    fourConnectDummy = FourConnect()
    fourConnectDummy.SetCurrentState(currentState)
    gameWinner=0

    #checking action validity
    if currentState[0][action]==0:
        #p2 playing action
        fourConnectDummy.GameTreePlayerAction(action)
        gameWinner = self.checkWinner(fourConnectDummy)

        #checking is p2 wins
        if gameWinner!=2:
            #now p1 plays
            #try block as if total moves exhausted p1 cant move and assert will throw error
            try:
                #getting state before p1 play
                state1 = fourConnectDummy.GetCurrentState()
                #p1 plays myopic
                fourConnectDummy.MyopicPlayerAction()
                gameWinner = self.checkWinner(fourConnectDummy)

                #checking is p1 wins
                if gameWinner!=1:
                    stateNow = fourConnectDummy.GetCurrentState()
                    bestMove1,rewardBest1,winDepth1 = self.MoveFinder(stateNow,depth-1)
                    minDepthArr1,minim1 = self.minOfArrIndx(winDepth1)
                    winDepth[action]=minim1+1
                    rewards.append(rewardBest1)
                else:
                    #p2 wins so we try to block
                    #getting state after p1 play
                    state2 = fourConnectDummy.GetCurrentState()
                    #finding p1 winning move so we can block it
                    p1Move = self.CheckP1Col(state1,state2)
                    p1WinCols.append(p1Move)
                    rewards.append(-1)
            #moves exhausted so throws and hence draw
            except AssertionError as e:
                rewards.append(0)
        else:
            #p2 wins so reward add 1 and winNow add action
            winDepth[action]=0
            winNow.append(action)
            rewards.append(1)
    else:
        #not valid action
        rewards.append(-2)
```

(a)Evauation Function (comparison)

- Depth 3 (50 PlayGame iterations)
- Evaluation function with just my rewards , win now, P1 blocking.
- Not WinDepth(priority3) which i implemented later.
- Without move order heuristic (added later)

```
#PlayGame:
Wins:  20 , Avg Moves:  24.0
Losses:  29 , Avg Moves:  22.310344827586206
Draws:  1 , Avg Moves:  42.0
(base) rohit@rohit-ubuntu:~/Desktop/Connect4AI$
```

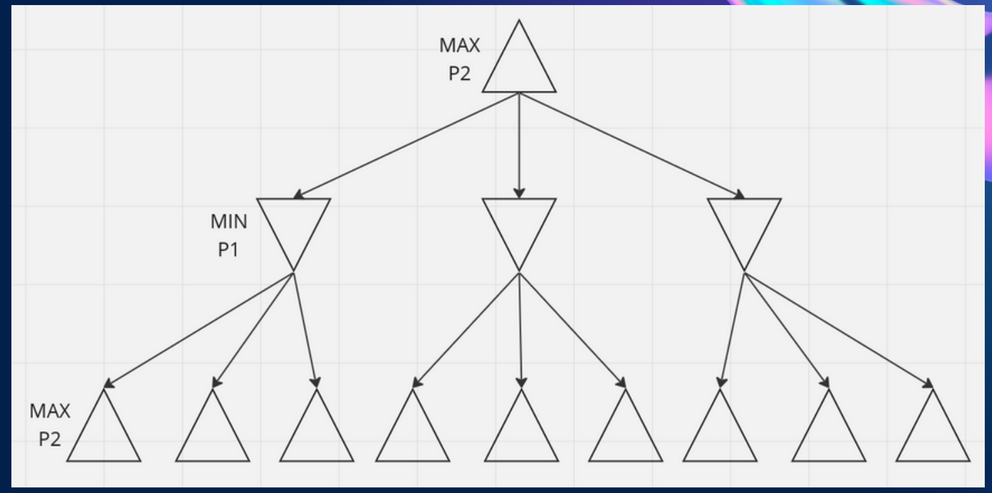
- Depth 3 (50 PlayGame iterations)
- With WinDepth implemented.
- Which i figured out iteratively.
- Without move order heuristic (added later)

```
-----50 Games-----
#PlayGame:
Wins:  34 , Avg Moves:  17.235294117647058
Losses:  16 , Avg Moves:  20.5
(base) rohit@rohit-ubuntu:~/Desktop/Connect4AI$
```

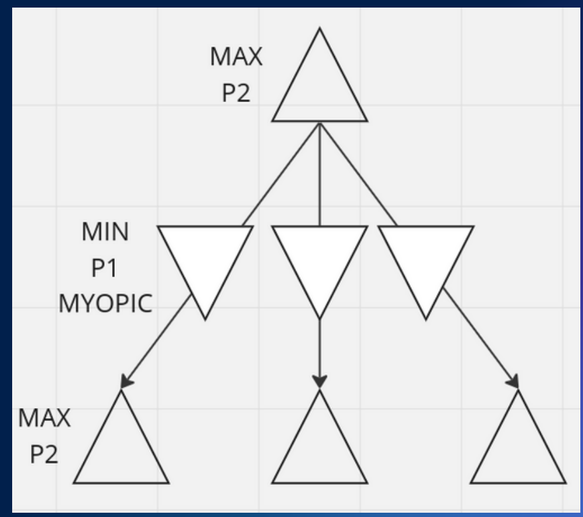
(b)Alpha Beta Pruning (comparison)

- Figure (1) , (2) are just illustrations
- In Minimax both player follow the same strategy and we can use alpha beta pruning.(Figure(1))
- But in my case Min player is just playing Myopic (Figure(2)) so i cant figure out Beta and how to do alpha beta pruning as i created my own Logic and is not exactly Minimax.

(1)



(2)



(c)Move ordering Heuristic (comparison)

- Depth 3 (50 PlayGame iterations)
- Without Move Order Heuristic

```
#PlayGame:
Wins:  35 , Avg Moves:  20.571428571428573 , Avg Time:  0.23505714285714285 sec
Losses:  14 , Avg Moves:  21.714285714285715 , Avg Time:  0.22842857142857143 sec
Draws:  1 , Avg Moves:  42.0 , Avg Time:  0.331 sec
(base) rohit@rohit-ubuntu:~/Desktop/Connect4AI$
```

- Depth 3 (50 PlayGame iterations)
- With Move Order Heuristic

```
-----50 Games-----
#PlayGame:
Wins:  30 , Avg Moves:  19.2 , Avg Time:  0.20726666666666665 sec
Losses:  19 , Avg Moves:  20.68421052631579 , Avg Time:  0.20015789473684212 sec
Draws:  1 , Avg Moves:  42.0 , Avg Time:  0.303 sec
```

(d)Depth Increase (comparison)(Part1)

- Depth 3 (50 PlayGame iterations)
- Without move order heuristic (added later)

```
-----50 Games-----

#PlayGame:

Wins:  34 , Avg Moves:  17.235294117647058

Losses:  16 , Avg Moves:  20.5

(base) rohit@rohit-ubuntu:~/Desktop/Connect4AI$
```

- Depth 5 (50 PlayGame iterations)
- Here we can see Wins Increased and Avg moves needed decreased.
- Without move order heuristic (added later)

```
-----50 Games-----

#PlayGame:

Wins:  37 , Avg Moves:  15.567567567567568

Losses:  13 , Avg Moves:  19.153846153846153

(base) rohit@rohit-ubuntu:~/Desktop/Connect4AI$
```

(d)Depth Increase (comparison) (Part2)

- Depth 3 (50 PlayGame iterations)
- With move order heuristic

```
-----50 Games-----

#PlayGame:

Wins:  30 , Avg Moves:  19.2 , Avg Time:  0.20726666666666665 sec

Losses:  19 , Avg Moves:  20.68421052631579 , Avg Time:  0.20015789473684212 sec

Draws:  1 , Avg Moves:  42.0 , Avg Time:  0.303 sec
```

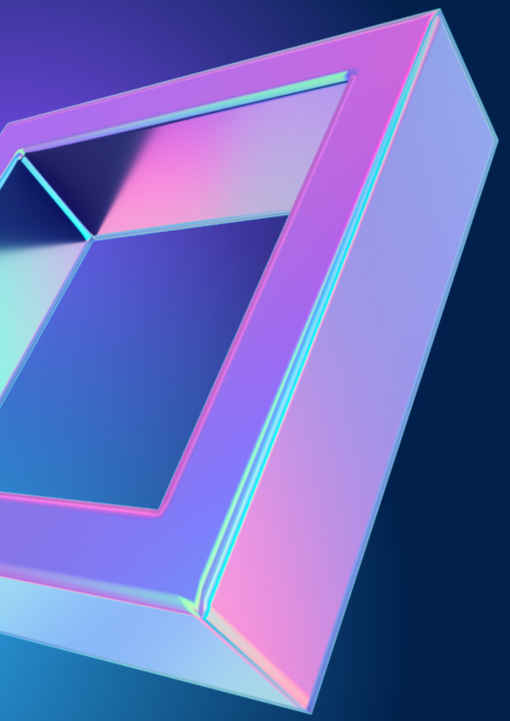
- Depth 5 (50 PlayGame iterations)
- Here we can see Wins Increased and Avg moves needed decreased.
- With move order heuristic
- Ignore time as it is bound to increase with depth

```
-----50 Games-----

#PlayGame:

Wins:  38 , Avg Moves:  17.31578947368421 , Avg Time:  9.605052631578948 sec

Losses:  12 , Avg Moves:  17.166666666666668 , Avg Time:  8.92425 sec

(base) rohit@rohit-ubuntu:~/Desktop/Connect4AI$
```

Findings

- Increasing depth gives better results in terms of moves needed and wins.
- Move ordering Heruristic improves the computation time to find a move for P2
- This algorithm will not always win as Myopic player doesn’t just have one move always in that state it has a set of moves and it uses random choice to choose between them.

Thank You



- I was a fun project to work on and i enjoyed doing it.
- And get to know my coding potential better.
- I gave some time to read the code and implemented everything on my own without anyone’s help.
- Overall i am fulfilled with the results i have achieved in this assignment.

Rohit Kodam
2020B3A71141G