CSC 215-01 Artificial Intelligence (Fall 2019)

Mini-Project 4: Solving Tic-Tac-Toe and Wild Tic-Tac-Toe using Minimax Search

Team Members:

Ayushi Vadwala (220234041)

Jeet Shah (220267750)

I. Problem Statement:

In this project, we aim to build two variations of tic tac toe.

The first one is tic tac toe where the first player plays x and the second player plays o. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. If all cells are used without someone getting a three in a row, the game is a draw.

The Second one is wild tic tac toe where on each turn a player can play either x or o. The player who gets the first three (x's or o's) in a row wins the game. If all cells are used without someone getting a three in a row, the game is a draw.

II. Methodology:

Tic Tac Toe

```
def availableMoves(board):
    #Return empty spaces on the board
    moves = []
    for i in range(0, len(board)):
        if board[i] != "x" and board[i] != "o":
            moves.append(i)
    return moves

def makeMove(board,position, player):
    #Make a move on the board
    board[position] = player
    return board

def changePlayer(player):
    #Returns the opposite player given any player
    if player == "x":
        return "o"
    else:
        return "x"
```

```
def minimax(board, player):    # return the minimax score of a node
    global current
    current_score = check_game_over(board)

    if current_score is not False:
        return current_score

    # if the game is not over, do the following
    scores = []
    moves = []
    x_win = False
    o_win = False

    # check all possible moves.
    availablemoves = availableMoves(board)

    for i in availablemoves:
        new_board = board.copy()
        node = {}
        new_board = makeMove(new_board , i , player)
        node['move'] = new_board

        score = minimax(new_board,changePlayer(player))
        node['score'] = score

        moves.append(node)
```

```
  if player == "x":
      best_score = -100
      best_node = node
      # Find the move with the highest score.   Add that move to current and return that score.
      for node in moves:
          if node['score'] > best_score:
              best_score = node['score']
              best_node = node
      current.append(best_node['move'])

  if player == "o":
      best_score = 100
      best_node = node
      # Find the move with the lowest score.   Add that move to current and return that score.

      for node in moves:
          if node['score'] < best_score:
              best_score = node['score']
              best_node = node
      current.append(best_node['move'])

  return best_score
```

Wild Tic Tac Toe

```
def availableMoves(board):
    #Return empty spaces on the board
    moves = []
    for i in range(0, len(board)):
        if board[i] != "x" and board[i] != "o":
            moves.append(i)
    return moves

def makeMove(board,board1,position, player):
    #Make a move on the board
    board1[position] = 'o'
    board[position] = 'x'
    return board,board1

def minimax(board, player):
    global current
    current_score = check_game_over(board, player)

    if current_score is not False:
        return current_score

    scores = []
    moves = []
    p1_win = False
    p2_win = False
    flag = None
    # check all possible moves.
    availablemoves = availableMoves(board)


    for i in availablemoves:
        new_board1 = board.copy()
        new_board2 = board.copy()
        node = {}
        new_board1,new_board2 = makeMove(new_board1, new_board2 , i , player)

        if player == "P1":
            score = minimax(new_board1, "P2")
            score1 = minimax(new_board2, "P2")

        if player == "P2":
            score = minimax(new_board1, "P1")
            score1 = minimax(new_board2, "P1")

        node['move1'] = new_board1
        node['score'] = score
        node['move2'] = new_board2
        node['score1'] = score1

        moves.append(node)


    if player == "P1":
    # Find the move with the highest score.   Add that move to current and return that score.
```

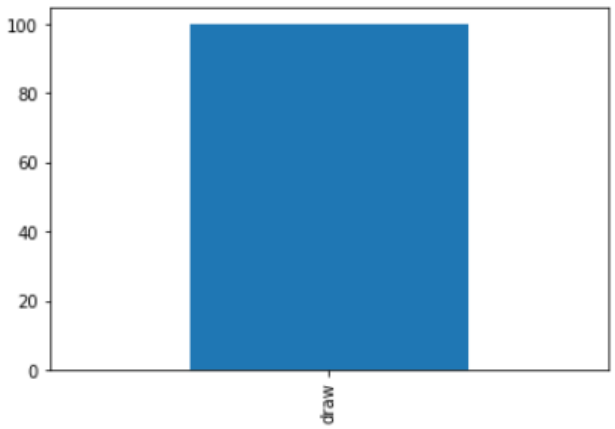
```
    best_score = -100
    best_node = {}
    for node in moves:
        if node['score'] > node['score1']:
            if node['score'] > best_score:
                best_score = node['score']
                best_node = node
                flag = 1
        else:
            if node['score1'] > best_score:
                best_score = node['score1']
                best_node = node
                flag = 0
    if flag == 1:
        current.append(best_node['move1'])
    else:
        current.append(best_node['move2'])


elif player == "P2":
# Find the move with the lowest score.   Add that move to current and return that score.

    best_score = 100
    best_node = {}
    for node in moves:
        if node['score'] < node['score1']:

            if node['score'] < best_score:
                best_score = node['score']
                best_node = node
                flag = 1
        else:
            if node['score1'] < best_score:
                best_score = node['score1']
                best_node = node
                flag = 0
    if flag == 1:
        current.append(best_node['move1'])
    else:
        current.append(best_node['move2'])

return best_score
```

III. Experimental Results and Analysis

Tic Tac Toe

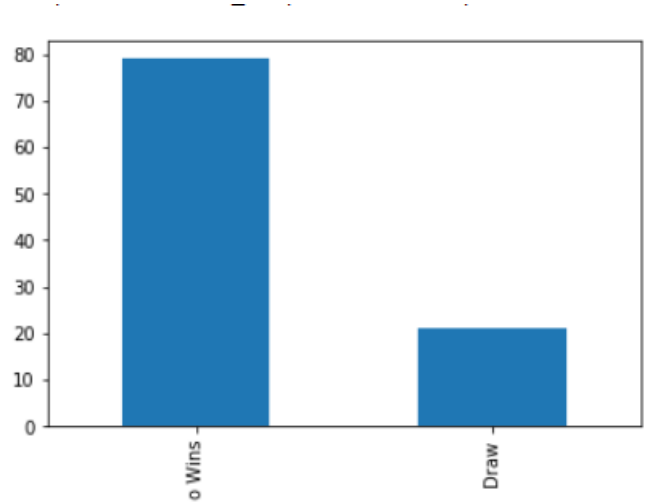
Optimal Vs Optimal

For experimenting we tried the Optimal Vs Optimal function 100 times and the result is given as below:



Random Vs Optimal

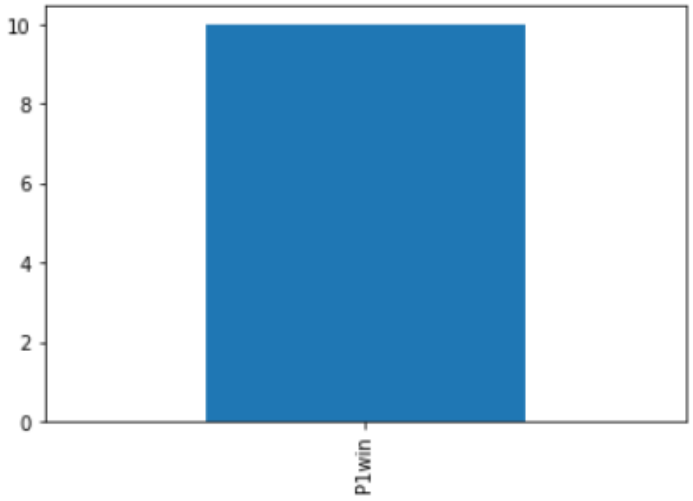
For experimenting we tried the Random Vs Optimal function 100 times and the result is given as below:



Wild Tic Tac Toe

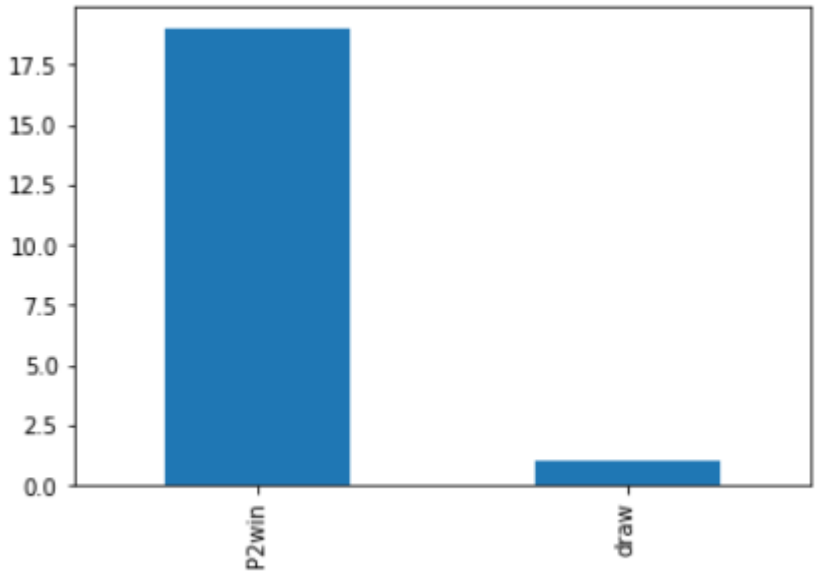
Optimal Vs Optimal

For experimenting we tried the Optimal Vs Optimal function 10 times and the result is given as below:



Random Vs Optimal

For experimenting we tried the Random Vs Optimal function 20 times and the result is given as below:



IV. Task Division and Project Reflection

Name: Ayushi Vadwala

Tasks performed:

Worked on minimax for tic tac toe
Worked on graph for the multiple outputs
Testing using you_vs_optimal

Name: Jeet Shah

Tasks performed:

Worked on minimax for wild tic tac toe
Worked on graph for the multiple outputs
Testing using you_vs_optimal

V. Learnings

- How to use minimax as a decision rule for AI
- How to use recursion to decide the move in minimax