Richard Nadar
RA1911030010109
12-4-2022

ARTIFICIAL INTELLIGENCE (18CSC305J) LAB

EXPERIMENT 6: Implementation of MiniMax algorithm for an application

Aim: To implement min max algorithm for tic tac toe problem.

Explanation:

The key to the Minimax algorithm is a back and forth between the two players, where the player whose "turn it is" desires to pick the move with the maximum score. In turn, the scores for each of the available moves are determined by the opposing player deciding which of its available moves has the minimum score.

And the scores for the opposing players moves are again determined by the turn-taking player trying to maximize its score and so on all the way down the move tree to an end state.

A description for the algorithm, assuming X is the "turn taking player," would look something like:

If the game is over, return the score from X's perspective.

Otherwise get a list of new game states for every possible move

Create a scores list

For each of these states add the minimax result of that state to the scores list

If it's X's turn, return the maximum score from the scores list

If it's O's turn, return the minimum score from the scores list

You'll notice that this algorithm is recursive, it flips back and forth between the players until a final score is found.

CODE:

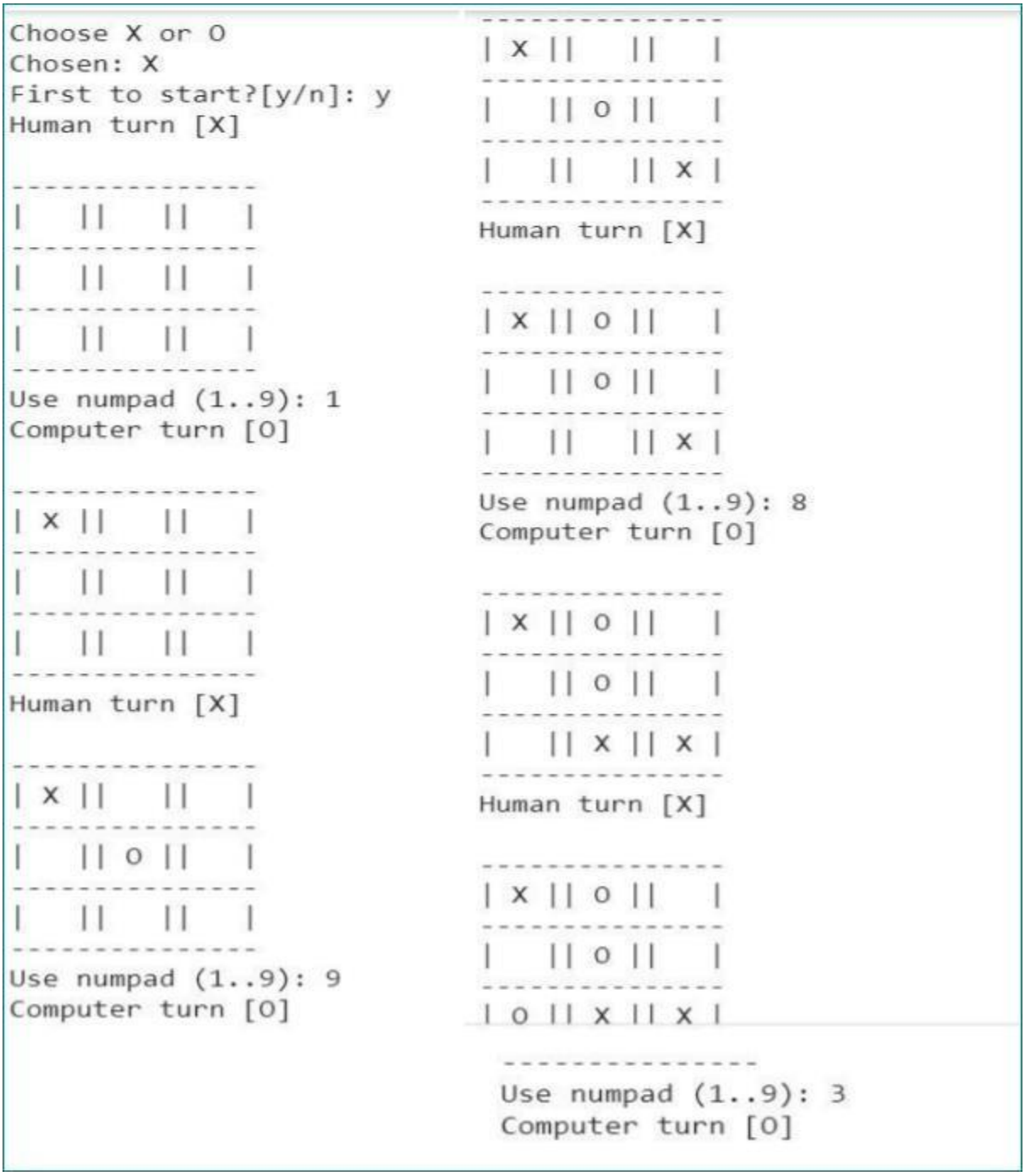
```
def printBoard(board):
        print(board[1] + '|' + board[2] + '|' + board[3]) print('-+-+-')
```

```
        print(board[4] + '|' + board[5] + '|' + board[6]) print('-+-+-')
        print(board[7] + '|' + board[8] + '|' + board[9]) print("\n")

def spaceIsFree(position):
        if board[position] == ' ':
                return True
        else:
                return False
def insertLetter(letter, position):
        if spaceIsFree(position):
                board[position] = letter printBoard(board)
        if (checkDraw()):
                print("Draw!") exit()
        if checkForWin():
                if letter == 'X':
                        print("Bot wins!")
                        exit()
                else:
                        print("Player wins!")
                        exit()
                        return
        else:
                print("Can't insert there!")
        position = int(input("Please enter new position:         ")) insertLetter(letter, position)
        return
def checkForWin():
        if (board[1] == board[2] and board[1] == board[3] and board[1] ! = ' '):
                return True
        elif (board[4] == board[5] and board[4] == board[6] and board[4] ! = ' '):
                return True
        elif (board[7] == board[8] and board[7] == board[9] and board[7] != ' '):
                return True
        elif (board[1] == board[4] and board[1] == board[7] and board[1] != ' '):
                return True
        elif (board[2] == board[5] and board[2] == board[8] and board[2] != ' '):
                return True
        elif (board[3] == board[6] and board[3] == board[9] and board[3] != ' '):
                return True
        elif (board[1] == board[5] and board[1] == board[9] and board[1] != ' '):
                return True
        elif (board[7] == board[5] and board[7] == board[3] and board[7] != ' '):
        return True
        else:
                return False
def checkWhichMarkWon(mark):
        if board[1] == board[2] and board[1] == board[3] and board[1] == mark:
                return True
        elif (board[4] == board[5] and board[4] == board[6] and board[4] == mark):
```

```
                return True
        elif (board[7] == board[8] and board[7] == board[9] and board[7] == mark):
                return True
        elif (board[1] == board[4] and board[1] == board[7] and board[1] == mark):
                return True
        elif (board[2] == board[5] and board[2] == board[8] and board[2] == mark):
                return True
        elif (board[3] == board[6] and board[3] == board[9] and board[3] == mark):
                return True
        elif (board[1] == board[5] and board[1] == board[9] and board[1] == mark):
                return True
        elif (board[7] == board[5] and board[7] == board[3] and board[7] == mark):
                return True
        else:
                return False
def checkDraw():
        for key in board.keys():
                if (board[key] == ' '):
                        return False
                else:
                        return True

def playerMove():
        position = int(input("Enter the position for 'O':         "))
        insertLetter(player, position)
        return

def compMove():
        bestScore = -800
        bestMove = 0
        for key in board.keys():
                if (board[key] == ' '):
                        board[key] = bot
        score = minimax(board, 0, False)
        board[key] = ' '
                if (score > bestScore):
                        bestScore = score
                        bestMove = key
        insertLetter(bot, bestMove)
        return

def minimax(board, depth, isMaximizing):
        if (checkWhichMarkWon(bot)):
                return 1
        elif (checkWhichMarkWon(player)):
                return -1
        elif (checkDraw()):
                return 0
```

```
        if (isMaximizing):
                bestScore = -800
        for key in board.keys():
                if (board[key] == ' '):
                        board[key] = bot
        score = minimax(board, depth + 1, False)
        board[key] = ' '
        if (score > bestScore):
                bestScore = score
                return bestScore
        else:
                bestScore = 800
        for key in board.keys():
                if (board[key] == ' '):
                        board[key] = player
        score = minimax(board, depth + 1, True)
        board[key] = ' '
        if (score < bestScore):
                bestScore = score
                return bestScore

board = {1: ' ', 2: ' ', 3: ' ',
4: ' ', 5: ' ', 6: ' ',
7: ' ', 8: ' ', 9: ' '}
printBoard(board)
print("Computer goes first! Good luck.")

print("Positions are as follow:")
print("1,          2,        3         ")
print("4,          5,        6         ")
print("7,          8,        9         ")
print("\n") player = 'O' bot = 'X'
global firstComputerMove firstComputerMove = True while not checkForWin():
compMove()
playerMove()
```

OUTPUT:



```
Choose X or O
Chosen: X
First to start?[y/n]: y
Human turn [X]

---------------
|   ||   ||   |
---------------
|   ||   ||   |
---------------
|   ||   ||   |
---------------
Use numpad (1..9): 1
Computer turn [O]

---------------
| X ||   ||   |
---------------
|   ||   ||   |
---------------
|   ||   ||   |
---------------
Human turn [X]

---------------
| X ||   ||   |
---------------
|   || O ||   |
---------------
|   ||   ||   |
---------------
Use numpad (1..9): 9
Computer turn [O]

---------------
| X ||   ||   |
---------------
|   || O ||   |
---------------
|   ||   || X |
---------------
Human turn [X]

---------------
| X || O ||   |
---------------
|   || O ||   |
---------------
|   ||   || X |
---------------
Use numpad (1..9): 8
Computer turn [O]

---------------
| X || O ||   |
---------------
|   || O ||   |
---------------
|   || X || X |
---------------
Human turn [X]

---------------
| X || O ||   |
---------------
|   || O ||   |
---------------
| O || X || X |
---------------
Use numpad (1..9): 3
Computer turn [O]
```

RESULT: Min-max algorithm is successfully implemented in python.