Name - Vasu Kalariya
Roll - PE29
Sub - AI

Lab Assignment - 2

Title: Implementation of MinMax algorithm for Tic-Toc-Toe game.

Aim: Solve Tic-Toc-Toe using MinMax algorithm

Objective: To study & implement minmax algorithm for Tic Toc Toe

Theory

→ **Adversarial search:**
Adversarial search is a search when there is an enemy or opponent changing the state of the problem every step in a direction you do not want.
Eg: Chess, business, trading, war.
You change state, but then you don't control next state opponent will change next state in a way.
a) Unpredictable
b) hostile to you
You can get to change every alternate state

→ **Tic-Toc-Toe solving steps**
Consider two opponents, 1st represent by X & the other by 'O' where we aim on maximizing the chance of 'X' winning. Rules are as follow.
a) If X wins, take it
b) If opponent wins, block it
c) If possible create a fork (2 wining ways)
d) Do not let opponent block 'X' winning move
e) If neither 'X' or 'O' wins call it a tie.

→ Data Structure & other details about MinMax algo.
MinMax is a backtracking algo. that is used in decision making & game theory to find optimal move for a player. A Binary tree is used for this algo. It has 2 players ~~maxmin~~ maximizer who tries to get highest ~~so~~ score possible & minimizer who tries to get highest score possible. It is widly used in 2 player turn-based games such as tic-toc-toe Backgammon, mancala, Chess, etc. Performs depth-first search algorithm

Input : Initial State

Output: Solution/goal state with optimal path

Algorithm: MinMax

FAQ

1 Compare Informed search & adversarial search.

Informed search	Adversarial search
→ Uses knowledge for search process	→ Doesn't use knowledge for searching process.
→ Finds solution quickly	→ Finds solution slowly
→ Cost is low	→ Cost is height
→ It takes less time	→ It takes moderate time
→ less lengthy while implementation	→ more lengthy while implementation.

2 Explain minimax algorithm with example.

Every board state has value associated with it. In a given state if maximizer has upper hand then score of board will tend to be some positive value if minimizer has upper hand in that board state then it will tend to be some negative value. Values of board are calculated by some heuristics which are unique for every type of game.

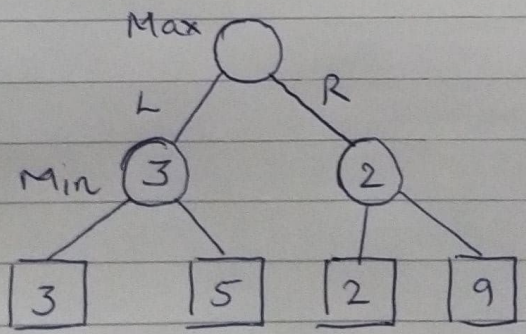
Eg: Consider game with 4 final states & maximizing player starting first. The game tries all possible moves since its a backtracking algo.

Maximizer goes left - it is minimizer turn now, that has choice betw 3 & 5

Being minimizer it will choose least among both that is 3

Maximizer goes right - It is minimizer turn. It has now a choice b/w 2 & 9 He will choose 2

Maximizer will choose largest value = 3. Hence optimal move for maximizer is to go left



3 Explain Alpha beta pruning.

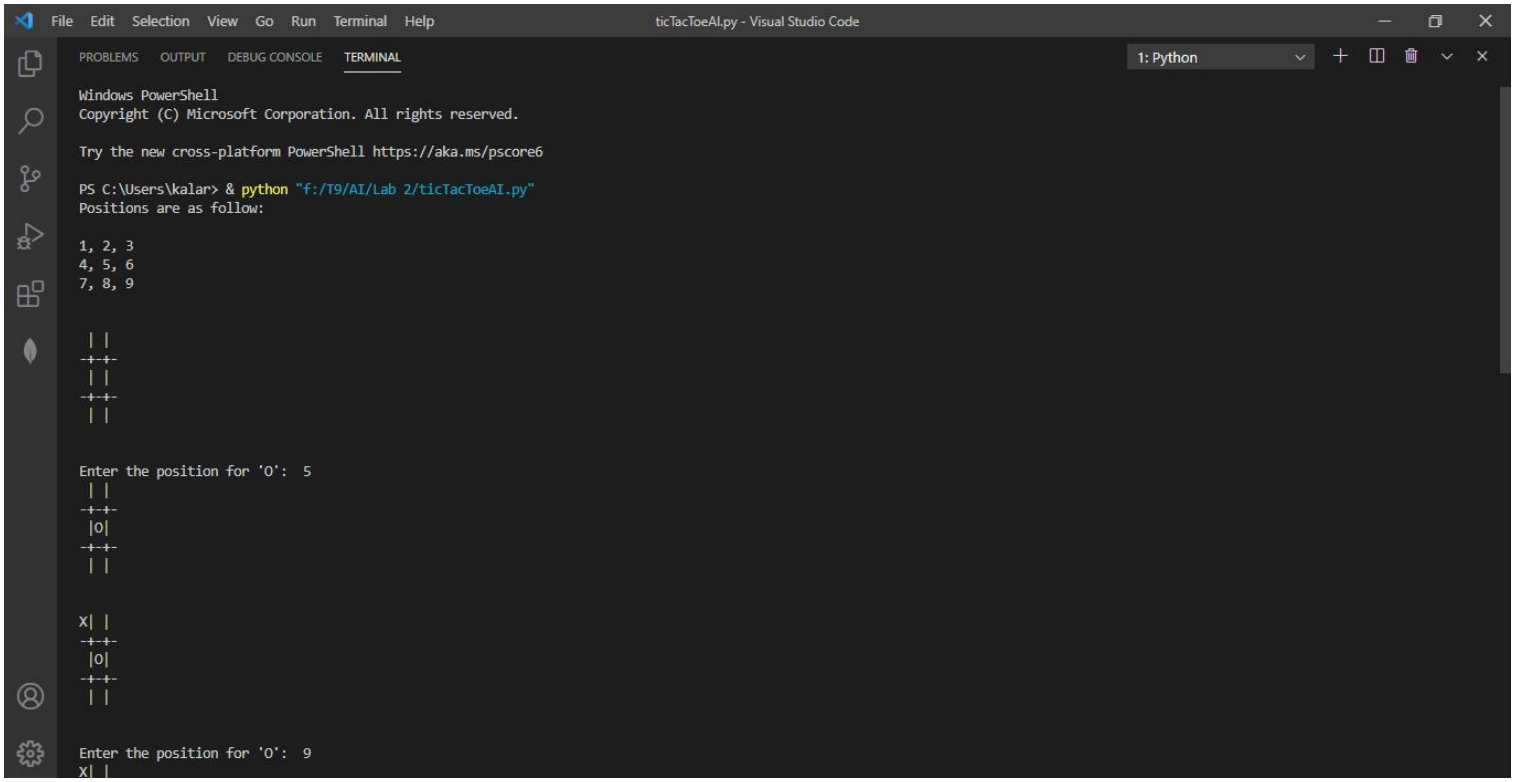
It is a optimization technique for minimax algorithm. It reduces computation time & allows us to search much faster & even go into deeper levels in game tree. It cuts off branches in game ~~tree~~ tree which need not be searched because there already exists a better move available. It passes 2 extra parameters in minimax function

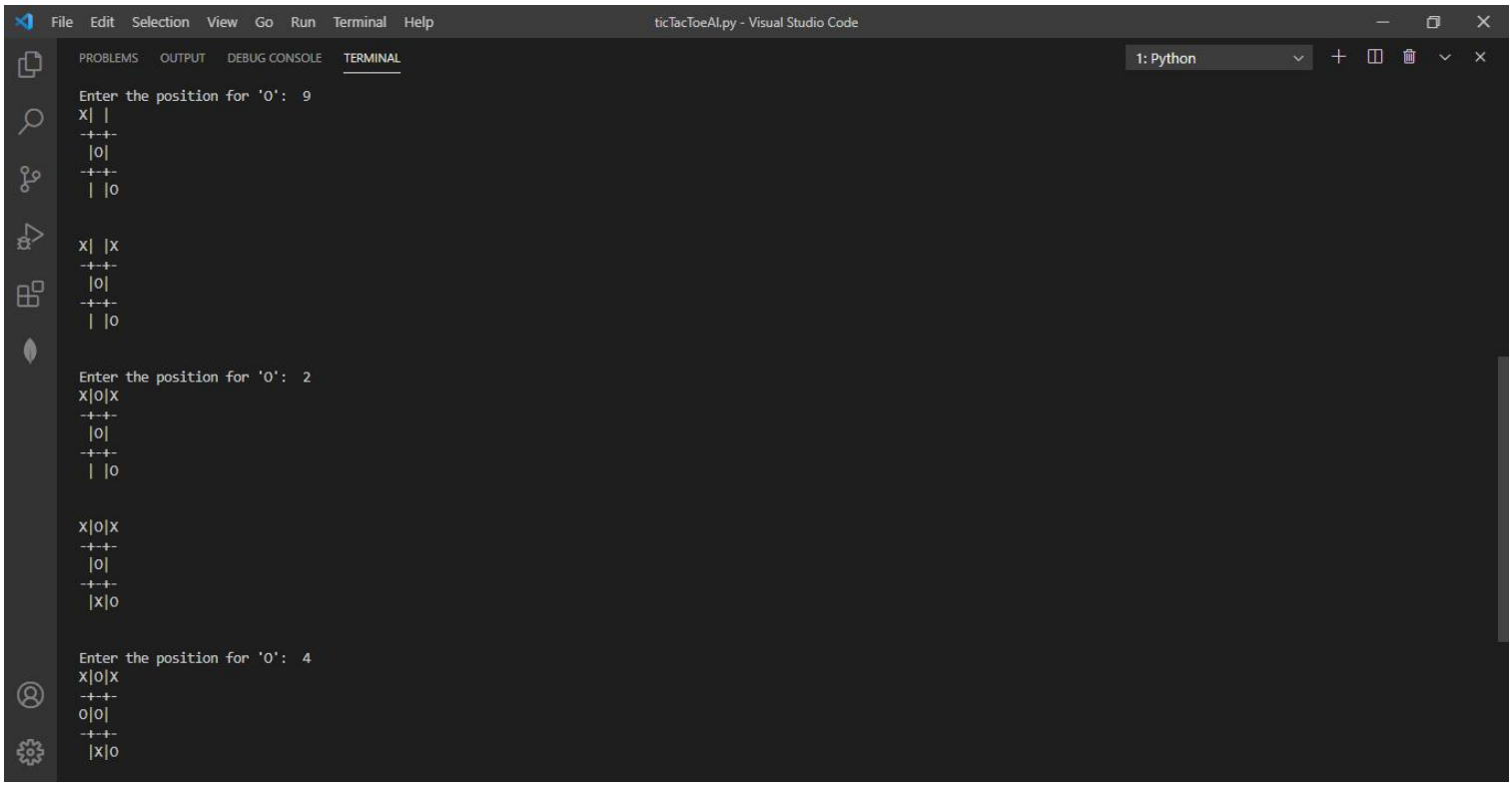
Alpha - Best value that maximizer currently can guarantee at that ~~to~~ level or above
Beta - Best value that minimizer currently can guarantee at that level or above.

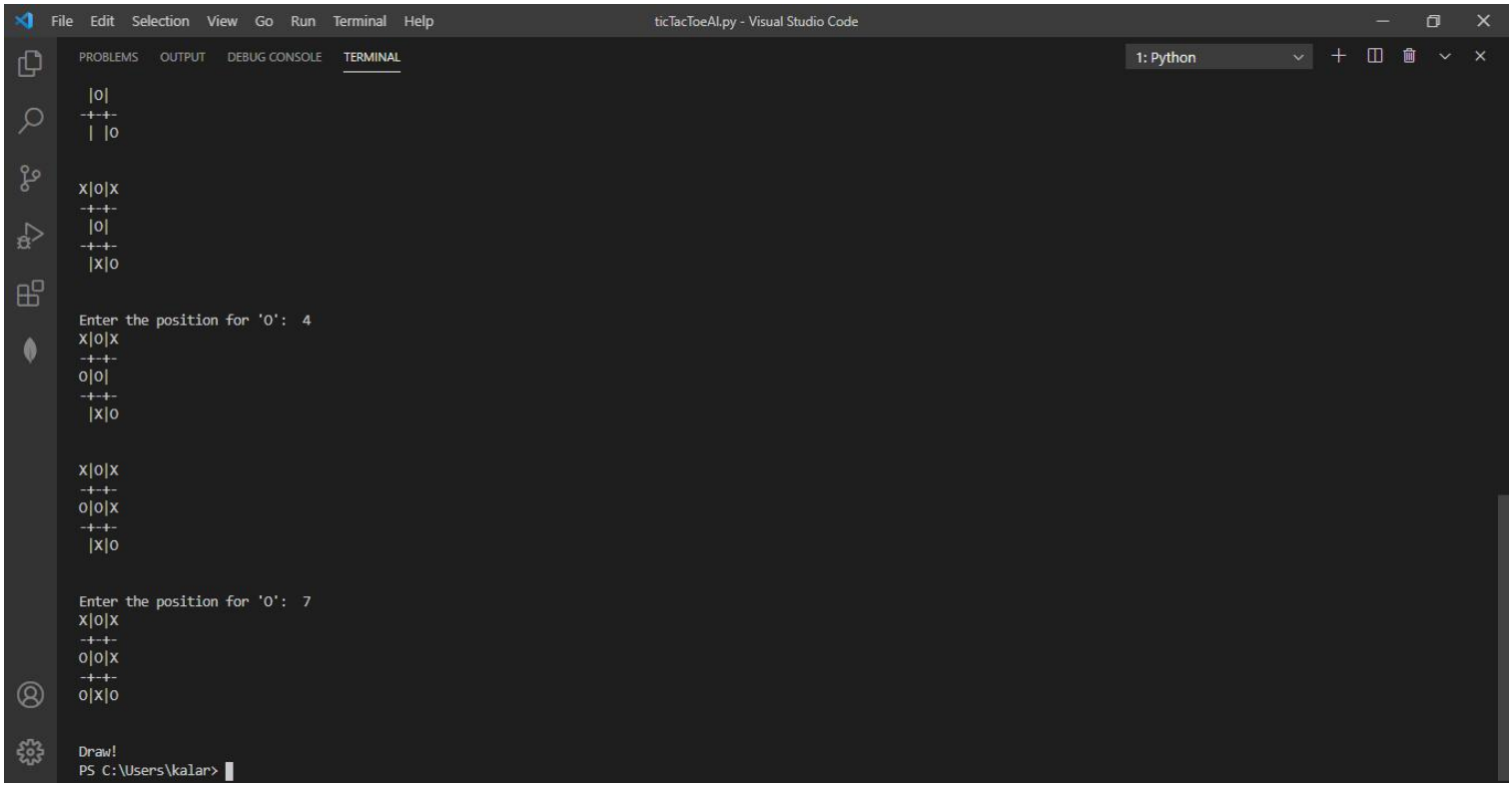
```python
#   Name : Vasu Kalariya
#   Roll : PE29
#   AI lab Assi 2 (MinMax)

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):                  # checking for the space is free or not
    if board[position] == ' ':
        return True
    else:
        return False


def insertSymbol(letter, position):             # insert symbol at given number
    if spaceIsFree(position):                   # check for free space
        board[position] = letter
        printBoard(board)
        if (checkDraw()):                       # check for Draw
            print("Draw!")
            exit()
        if checkForWin():                       # check for WIN
            if letter == 'X':
                print("AI wins!")
                exit()
            else:
                print("Player wins!")
                exit()

        return


    else:                                       # space is already filled
        print("Can't insert there!")
        position = int(input("Please enter new position:  "))
        insertSymbol(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 checkWhichSymbolWon(symbol):
    if board[1] == board[2] and board[1] == board[3] and board[1] == symbol:
        return True
    elif (board[4] == board[5] and board[4] == board[6] and board[4] == symbol):
        return True
    elif (board[7] == board[8] and board[7] == board[9] and board[7] == symbol):
        return True
    elif (board[1] == board[4] and board[1] == board[7] and board[1] == symbol):
        return True
    elif (board[2] == board[5] and board[2] == board[8] and board[2] == symbol):
        return True
    elif (board[3] == board[6] and board[3] == board[9] and board[3] == symbol):
        return True
    elif (board[1] == board[5] and board[1] == board[9] and board[1] == symbol):
        return True
    elif (board[7] == board[5] and board[7] == board[3] and board[7] == symbol):
        return True
    else:
        return False


def checkDraw():
    for key in board.keys():
        if (board[key] == ' '):                    # if there is empty space
            return False
    return True                                    # if no empty space left


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


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

    insertSymbol(AI, bestMove)
    return


def minimax(board, isMaximizing):
    if (checkWhichSymbolWon(AI)):
        return 1
    elif (checkWhichSymbolWon(player)):
```

```
        return -1
    elif (checkDraw()):
        return 0

    if (isMaximizing):                                  # trying to Maximize score
        bestScore = -1000
        for key in board.keys():
            if (board[key] == ' '):
                board[key] = AI
                score = minimax(board, False)
                board[key] = ' '
                if (score > bestScore):
                    bestScore = score
        return bestScore

    else:                                               # trying to Minimize score at
next depth
        bestScore = 1000
        for key in board.keys():
            if (board[key] == ' '):
                board[key] = player
                score = minimax(board, True)
                board[key] = ' '
                if (score < bestScore):
                    bestScore = score
        return bestScore


board = {1: ' ', 2: ' ', 3: ' ',
         4: ' ', 5: ' ', 6: ' ',
         7: ' ', 8: ' ', 9: ' '}



print("Positions are as follow:")
print("")
print("1, 2, 3 ")
print("4, 5, 6 ")
print("7, 8, 9 ")
print("\n")
player = 'O'
AI = 'X'
printBoard(board)

while not checkForWin():
    playerTurn()
    compTurn()
```


File Edit Selection View Go Run Terminal Help
ticTacToeAI.py - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\kalar> & python "f:/T9/AI/Lab 2/ticTacToeAI.py"
Positions are as follow:
1, 2, 3
4, 5, 6
7, 8, 9
Enter the position for 'O': 5
Enter the position for 'O': 9


File Edit Selection View Go Run Terminal Help
ticTacToeAI.py - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python
Enter the position for 'O': 9
X| |
-+-+-
|O|
-+-+-
| |O
X| |X
-+-+-
|O|
-+-+-
| |O
Enter the position for 'O': 2
X|O|X
-+-+-
|O|
-+-+-
| |O
X|O|X
-+-+-
|O|
-+-+-
|X|O
Enter the position for 'O': 4
X|O|X
-+-+-
O|O|
-+-+-
|X|O


File Edit Selection View Go Run Terminal Help
ticTacToeAI.py - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Python
|O|
-+-+-
| |O
X|O|X
-+-+-
|O|
-+-+-
|X|O
Enter the position for 'O': 4
X|O|X
-+-+-
O|O|
-+-+-
|X|O
X|O|X
-+-+-
O|O|X
-+-+-
|X|O
Enter the position for 'O': 7
X|O|X
-+-+-
O|O|X
-+-+-
O|X|O
Draw!
PS C:\Users\kalar>