R.vishalini

231701062

Implement Minimax Algorithm

## AIM:

To implement the Minimax algorithm that helps an AI player choose the best move by simulating all possible game states and assuming the opponent plays optimally.

## PROGRAM:

def print\_board(board):

for row in board:

print(' '.join(row))

print()

def is\_moves\_left(board):

return any(cell == '\_' for row in board for cell in row)

def evaluate(board):

for i in range(3):

if board[i][0] == board[i][1] == board[i][2] != '\_':

return 10 if board[i][0] == 'X' else -10

if board[0][i] == board[1][i] == board[2][i] != '\_':

return 10 if board[0][i] == 'X' else -10

if board[0][0] == board[1][1] == board[2][2] != '\_':

return 10 if board[0][0] == 'X' else -10

if board[0][2] == board[1][1] == board[2][0] != '\_':

return 10 if board[0][2] == 'X' else -10

return 0

def minimax(board, is\_max):

score = evaluate(board)

if score != 0 or not is\_moves\_left(board):

return score

if is\_max:

best = -float('inf')

for i in range(3):

for j in range(3):

if board[i][j] == '\_':

board[i][j] = 'X'

best = max(best, minimax(board, False))

board[i][j] = '\_'

return best

else:

best = float('inf')

for i in range(3):

for j in range(3):

if board[i][j] == '\_':

board[i][j] = 'O'

best = min(best, minimax(board, True))

board[i][j] = '\_'

return best

def find\_best\_move(board):

best\_val = -float('inf')

best\_move = (-1, -1)

for i in range(3):

for j in range(3):

if board[i][j] == '\_':

board[i][j] = 'X'

move\_val = minimax(board, False)

board[i][j] = '\_'

if move\_val > best\_val:

best\_val = move\_val

best\_move = (i, j)

return best\_move

# Example use

board = [

['X', 'O', 'X'],

['\_', 'O', '\_'],

['\_', '\_', 'X']

]

print("Board:")

print\_board(board)

move = find\_best\_move(board)

print(f"Best move for X: {move}")

## OUTPUT:

Board:

X O X

\_ O \_

\_ \_ X

Best move for X: (1, 2)

## RESULT:

The Minimax algorithm correctly analyzes all possible future moves and selects the best move for the AI player (X) by assuming the opponent (O) plays optimally. This ensures the AI either **wins or draws** in a perfect game.