UNIVERSITY OF ASIA PACIFIC

Department of Computer Science and Engineering

Course Title :
Artificial Intelligence and Expert Systems Lab
Course Code : CSE 404

Project No: 02

Submitted By:

Ripa Rani Biswas(20201001)

Md. Mominul Huque(20201049)

Anowar Hossen Farvez(20201059)

Submitted to:

Dr. Nasima Begum

Assistant Professor

Department of CSE,

University Of Asia Pacific.

Problem Title:

Implement Tic Tac Toe Game with Minimax and Alpha-Beta Pruning Algorithm.

Problem Description:

In this project, we need to implement the classic game of Tic Tac Toe as an adversarial search problem and develop an AI player using the minimax algorithm with alpha-beta pruning. We have to create a program that allows human vs. computer and computer vs. computer gameplay and should print the game board on the console after every move.

Tools And Languages:

1. Python
2. Google Colab

Tik Tac Toe:

Tic Tac Toe is a well-known game played on a 3x3 grid, where two players take turns marking empty spaces with their respective symbols, typically "X" and "O". The objective is to form a row, column, or diagonal of three of one's own symbols before the opponent does. The game ends in a draw if the grid is full and neither player achieves a winning pattern.

Minimax Algorithm with Alpha-Beta Pruning:

Alpha-Beta pruning is an optimization technique applied to the Minimax algorithm, significantly reducing computation time by eliminating unnecessary search branches in the game tree. The algorithm employs two parameters, alpha and beta, to track the best possible outcomes for the maximizing and minimizing players, respectively. By pruning branches that cannot lead to better outcomes, Alpha-Beta pruning enhances the efficiency of the Minimax algorithm.

Pseudo Code:

```
def minimax(board, depth, alpha, beta, maximizing_player, player):
    """
    Minimax algorithm with Alpha-Beta Pruning
    """
    if terminal(board) or depth == 0:
        return evaluate_board(board, player), None

    if maximizing_player:
        v = float("-inf")
        best_move = None
        for action in actions(board):
            new_board = result(board, action)
            val = minimax(new_board, depth - 1, alpha, beta, False, player)
            if val > v:
                v = val
                best_move = action
            alpha = max(alpha, v)
            if alpha >= beta: #beta cut-off
                break
        return v, best_move
    else:
        v = float("inf")
        best_move = None
        for action in actions(board):
            new_board = result(board, action)
            val = minimax(new_board, depth - 1, alpha, beta, True, player)
            if val < v:
                v = val
                best_move = action
            beta = min(beta, v)
            if beta <= alpha: #alpha cut-off
                break
        return v, best_move
```

Sample Input/output:

Human vs. Computer :

```
Welcome to Tic Tac Toe!
Please select a game mode:
1. Human vs. Computer
2. Computer vs. Computer
0. Quit
Enter your choice: 1
  |   |
---------
  |   |
---------
  |   |
---------

Your turn (X)
Enter your move (row col): 1 1
  |   |
---------
  | X |
---------
  |   |
---------

Computer's turn (O)
O |   |
---------
  | X |
---------
  |   |
---------

Your turn (X)
Enter your move (row col): 0 1
O | X |
---------
  | X |
---------
  |   |
---------

Computer's turn (O)
O | X |
---------
  | X |
---------
  | O |
---------

Your turn (X)
Enter your move (row col): 0 2
O | X | X
---------
  | X |
---------
  | O |
---------

Computer's turn (O)
O | X | X
---------
  | X |
---------
O | O |
---------

Your turn (X)
Enter your move (row col): 1 2
O | X | X
---------
  | X | X
---------
O | O |
---------

Computer's turn (O)
O | X | X
---------
O | X | X
---------
O | O |
---------

O wins!
Game Over
```

Computer vs. Computer :

```
Welcome to Tic Tac Toe!
Please select a game mode:
1. Human vs. Computer
2. Computer vs. Computer
0. Quit
Enter your choice: 2
  |   |
---------
  |   |
---------
  |   |
---------

  | X |
---------
  |   |
---------
  |   |
---------

  | X |
---------
  |   |
---------
  | O |
---------

  | X |
---------
  |   | X
---------
  | O |
---------

O | X |
---------
  |   | X
---------
  | O |
---------

O | X |
---------
  | X | X
---------
  | O |
---------

O | X |
---------
O | X | X
---------
  | O |
---------

O | X |
---------
O | X | X
---------
X | O |
---------

O | X | O
---------
O | X | X
---------
X | O |
---------

O | X | O
---------
O | X | X
---------
X | O | X
---------

It's a draw!
Game Over
```

Challenges

Building a Tic Tac Toe game with Minimax and Alpha-Beta Pruning can be tricky due to the complexity of these algorithms. Designing a user-friendly interface, optimizing the AI, and handling unexpected situations are also challenges. Balancing AI difficulty levels, maintaining clean code, and ensuring an enjoyable user experience are key. Thorough testing and potential integration into a larger project add to the complexity.

Conclusion

Creating a Tic Tac Toe game using Minimax and Alpha-Beta Pruning techniques is a helpful exercise in AI and game development. It teaches us about adversarial search, decision-making, and optimization. Building a smart AI opponent demonstrates the power of these algorithms in strategic games. Alpha-Beta Pruning makes the AI efficient and competitive in a simple game like Tic Tac Toe. This project also helps improve problem-solving and programming skills, which are valuable for tackling more complex AI games and applications. Overall, it's a great way to learn about AI and gaming.