Project - 2: Tic Tac Toe Game with Minimax and Alpha-Beta Pruning Algorithm

Course title: Artificial Intelligence Lab

Course code: CSE 404

Department of CSE

University of Asia Pacific

Submitted By:	**Submitted To:**
Md.Meherub Hossain Yemon (20201037) Md.Raihan kabir (20201048) Md.Sohel Rahman (20201054) Nooruddin Ahammed(19201011) Section: A2	Dr. Nasima Begum Assistant Professor, CSE Department University of Asia Pacific

Problem Title:

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

Problem Description:

In this assignment, 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 Used:

- PyCharm IDE
- Python Language
- MS Word

Tik Tac Toe:

Tic Tac Toe is a game for two players on a 3x3 grid. They take turns putting their symbol (often "X" and "O") in an empty space. The aim is to have three of your symbols in a row, either horizontally, vertically, or diagonally, before your opponent. The first player to do this win. If the grid is full and no one wins, the game ends in a draw.

Minimax Algorithm in Game Theory (Alpha-Beta Pruning):

Alpha-Beta pruning is not actually a new algorithm, but rather an optimization technique for the minimax algorithm. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

Let's define the parameters alpha and beta.

Alpha is the best value that the **maximizer** currently can guarantee at that level or above.
Beta is the best value that the **minimizer** currently can guarantee at that level or below.

Pseudocode:

```
function minimax (node, depth, isMaximizingPlayer, alpha, beta):

    if node is a leaf node:
        return value of the node

    if isMaximizingPlayer:
        bestVal = -INFINITY
        for each child node :
            value = minimax(node, depth+1, false, alpha, beta)
            bestVal = max( bestVal, value)
            alpha = max( alpha, bestVal)
            if beta <= alpha:
                break
        return bestVal

    else:
        bestVal = +INFINITY
        for each child node:
            value = minimax (node, depth+1, true, alpha, beta)
            bestVal = min (bestVal, value)
            beta = min( beta, bestVal)
            if beta <= alpha:
                break
        return bestVal
```

Challenges:

Building a Tic Tac Toe game with Minimax and Alpha-Beta Pruning can be quite challenging due to the intricacy of these algorithms. Designing an intuitive user interface, optimizing the AI for efficient decision-making, and gracefully handling unexpected scenarios are all considerable tasks. Balancing the difficulty levels of the AI, maintaining clean and organized code, and ensuring an engaging user experience are pivotal aspects of the development process. Thorough testing and potential integration into a broader project further add to the complexity, requiring careful attention to detail and robust implementation strategies.

Conclusion:

Creating a Tic Tac Toe game with Minimax and Alpha-Beta Pruning is not only a fun challenge but also an insightful journey into AI's potential in strategic decision-making. It showcases how algorithms can efficiently simulate human-like thinking in games, providing a foundation for more advanced AI applications. This project is a testament to the power of computer science in enhancing our problem-solving capabilities and offers a glimpse into the future of AI in gaming and beyond.