Tic-Tac-Toe with Minimax AI

TEAM

1. *Aryaman Shukla (102303622)*
2. *Armaan Gogoi (102303623)*

GROUP: 2C44

Synopsis

This project is an advanced Tic-Tac-Toe game featuring an AI opponent powered by the Minimax algorithm. The game will be built using python and can be played in the terminal where the program will be run.

Key Features:

- **Play against AI**: Play against an AI opponent.
- **Minimax AI**: The AI uses the Minimax algorithm to evaluate all possible moves and select the optimal strategy, making it an unbeatable opponent.
- **Optimized Gameplay**: The algorithm ensures efficient decision-making, with pruning techniques (like Alpha-Beta pruning) to speed up calculations.
- **Scalability**: The logic can be extended to larger grid-based games (e.g., 4x4 or 5x5 Tic-Tac-Toe).

How It Works:

1. The player chooses their symbol (X or O).
2. The game starts with either the player or AI making the first move.
3. The AI evaluates the board using the Minimax algorithm, simulating all possible moves to select the optimal one.
4. The game continues until a player(human or AI) wins or the board is full.
5. This project serves as a great introduction to AI-based game development and decision-tree algorithms.

MINI-MAX ALGORITHM

Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.

Min-Max algorithm is mostly used for game playing in AI such as Chess, Checkers, tic-tac-toe, go, and various other games. This algorithm computes the minimax decision for the current state.

Both the players fight it as the opponent player gets the minimum benefit while it should get the maximum benefit.

Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.

The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.

The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.