Muhammad Junaid Shaikh 22k4300

Izaan Mujeeb 22k4635

Hassan Shaikh 22k4440

AI Project: Three-Player Connect 4 with Minimax and Alpha-Beta Pruning

Complete Implementation

1. Overview

This project implements a multiplayer version of the classic Connect 4 game, adapted to support **three players**. The AI players use the **Minimax algorithm** enhanced with **Alpha-Beta pruning** to make strategic decisions. The game is built using Python and runs on the command line interface (CLI).

2. Game Logic

a. Game Board

A 6x7 grid is initialized using a matrix to represent the game board. Each cell can be empty (represented by 0) or filled by a player's token (represented by integers 1, 2, or 3).

b. Turn Management

The game supports three players. Turn order is managed by cycling through player numbers (1 → 2 → 3 → 1 → ...). The board is displayed after every move.

c. Move Validation

Before placing a token, the program checks:

- Whether the selected column is within range.
- Whether the column is not full.

If valid, the token is placed in the next available row in the chosen column.

d. Win Detection

The win condition is based on forming a line of four consecutive tokens horizontally, vertically, or diagonally. The system scans the board in all directions to detect such a pattern for each player.

3. AI Decision-Making (Minimax + Alpha-Beta Pruning)

a. Game Tree Exploration

For AI-controlled players, moves are chosen using the **Minimax algorithm**, which simulates possible future game states up to a certain depth. This recursive algorithm alternates between maximizing and minimizing the score for each simulated move, depending on the player's role.

b. Alpha-Beta Pruning

To optimize the Minimax process, **Alpha-Beta pruning** is implemented. It reduces unnecessary calculations by skipping branches that won't affect the final decision, which significantly improves performance.

c. Heuristic Evaluation

Since it's impractical to simulate the entire game tree for every move, a **heuristic function** evaluates the desirability of board states at leaf nodes. The evaluation considers:

- Number of tokens in a row for each player.
- Opportunities to form winning lines.
- Blocking potential winning lines of opponents.

d. Valid Move Selection

The AI collects a list of all columns where moves are possible. From the evaluated options, the column with the best heuristic score is selected and played.

4. Game End Conditions

The game ends when:

- A player wins by connecting four tokens.
- The board is completely filled (declared as a draw).

After every turn, the system checks for these conditions.

5. User Interaction

- **Human Input**: Player 1 is controlled manually. The player inputs a column number to place a token.
- **AI Input**: Players 2 and 3 are AI-controlled and compute their moves based on Minimax decisions.

The board is updated in the terminal after every move for real-time visualization.

6. Features and Strengths

- **Support for 3 Players**: Unlike traditional 2-player games, the turn logic and win conditions are extended for a third player.
- **Strategic AI**: Each AI move is calculated with look-ahead ability and competitive foresight, balancing offense and defense.
- **Optimized Search**: Alpha-Beta pruning ensures efficient performance even with large branching factors.

7. Execution & Testing

The game is run from the command line. After starting the script:

- Player 1 enters column numbers manually.
- The AI agents (Players 2 and 3) automatically compute and play their turns.

The system handles invalid inputs gracefully and continues until a win or draw is detected.