Domineering - A Web-Based AI-driven Game

Group members:

- Ruchika Sonagote - 21110212
- Harshal Sonawane - 21110213
- Nidhi Kumari - 21110139
- Kanamarlapudi Hema - 21110092

Under the guidance of **Prof. Neeldhara Misra** and **Prof. Manisha Padala**

Domineering is a two-player, combinatorial, strategy board game played on a grid. The players alternate placing dominos, which occupy two adjacent cells on the grid. Each player has a specific orientation for their dominos:

- **Player 1 (Vertical): Places domino's vertically.**
- **Player 2 (Horizontal): Places domino's horizontally.**

The primary goal of the project is to **create an AI agent** to play Domineering using **Minimax Algorithm with Alpha-Beta pruning**

Domineering is an example of an impartial game (like Nim) when viewed through the lens of combinatorial game theory. However, the roles (vertical vs. horizontal) make it asymmetric, providing unique challenges for AI development.

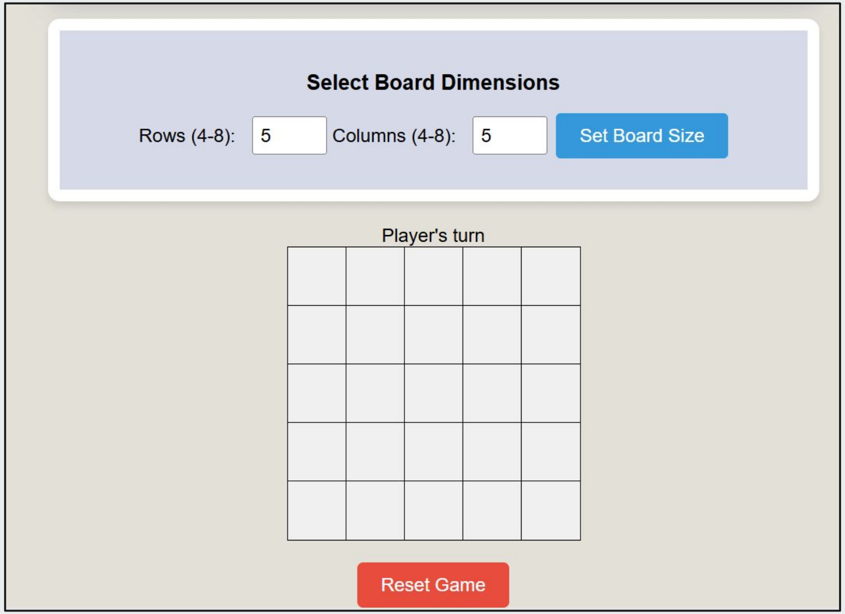
Rules

1. Players take turns placing dominos in their respective orientation.
2. A domino cannot overlap another domino or extend beyond the grid's boundaries.
3. The game ends when no legal moves are available for either player.
4. The player who cannot make a move loses the game.

Development Framework

Frontend Technologies

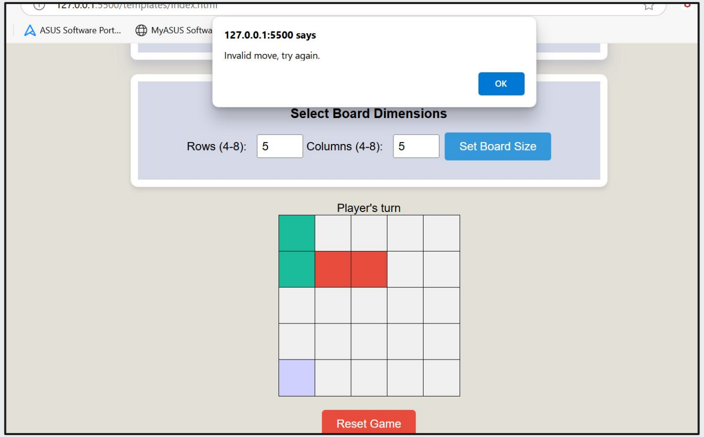
- HTML: Structure and layout of the game interface.
- CSS: Styling for an intuitive and visually appealing design.
- JavaScript: Adds interactivity and ensures responsive gameplay across devices.



Development Framework

Game Interface

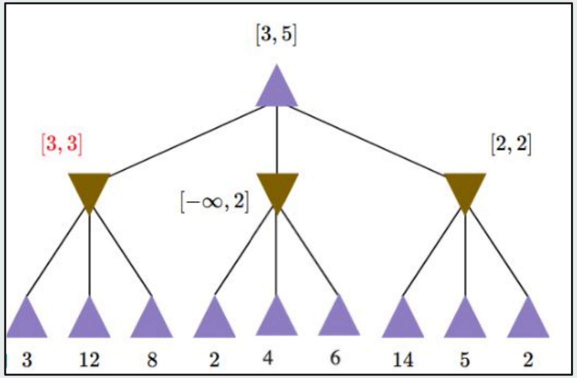
- Designed for a dynamic board dimension and engaging user experience.
- JavaScript handles player moves, AI turns, and rule enforcement.



Development Framework

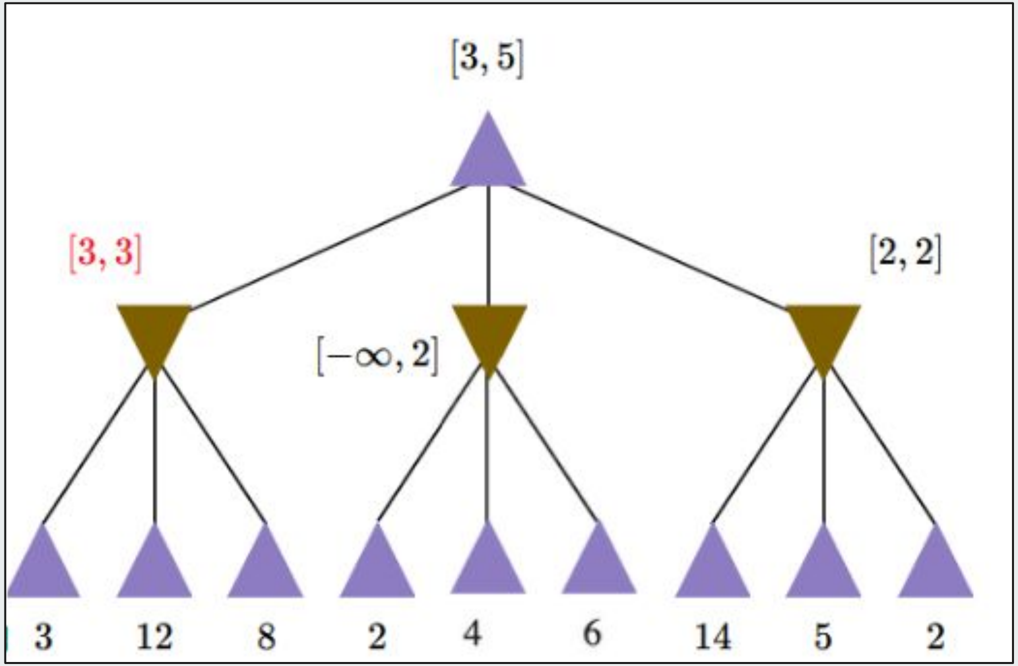
AI Logic

- Implemented with JavaScript.
- Used the Minimax algorithm with Alpha-Beta Pruning for efficient and strategic AI decision-making.

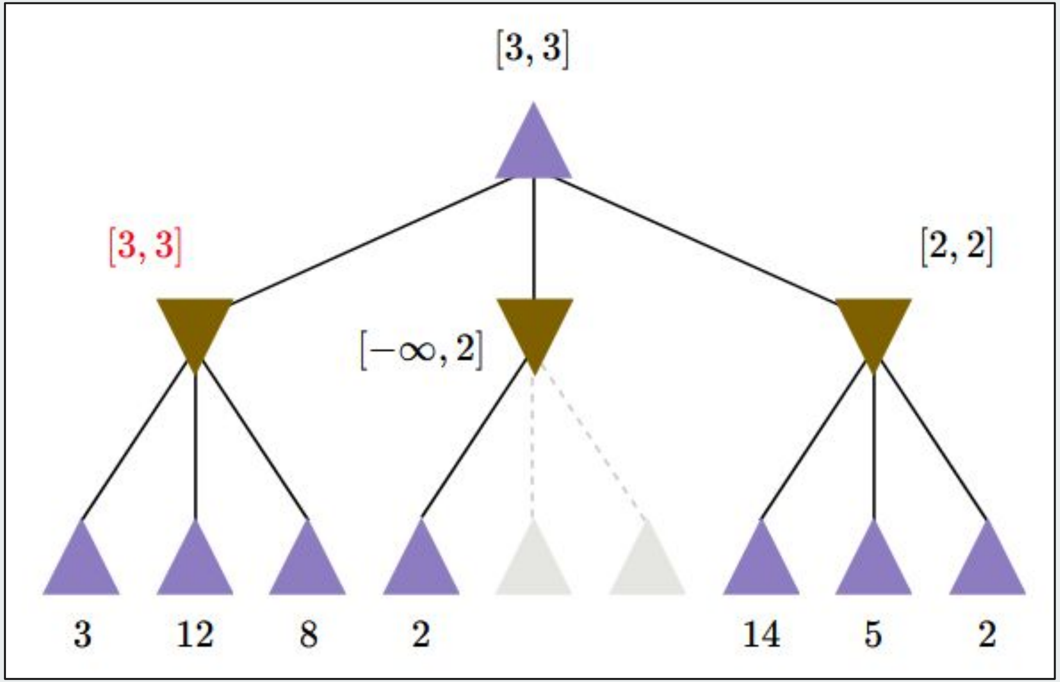


Minimax Algorithm

Minimax Algorithm



MiniMax with Alpha Beta Pruning



Implementation

- 0 indicates an empty cell.
- V and H indicate player and computer moves, respectively.

```
const getPossibilities = (player) => {
    let count = 0;
    for (let row = 0; row < boardlen; row++) {
        for (let col = 0; col < boardwid; col++) {
            if (player === PLAYER_VERTICAL) {
                if (row + 1 < boardlen && board[row][col] === '0' && board[row + 1][col] === '0') {
                    count++;
                }
            } else if (player === COMPUTER_HORIZONTAL) {
                if (col + 1 < boardwid && board[row][col] === '0' && board[row][col + 1] === '0') {
                    count++;
                }
            }
        }
    }
    return count;
};
```

```javascript
const alphabeta = (depth, player, alpha, beta) => {
    // End condition: no more depth or no more possible moves
    if (depth === 0 || getPossibilities(PLAYER_VERTICAL) === 0 || getPossibilities(COMPUTER_HORIZONTAL) === 0) {
        return { score: getPossibilities(COMPUTER_HORIZONTAL) - getPossibilities(PLAYER_VERTICAL), row: null, col: null };
    }
    const maximizingPlayer = (player === COMPUTER_HORIZONTAL);
    let bestScore = maximizingPlayer ? -Infinity : Infinity;
    let bestRow = null;
    let bestCol = null;
    for (let row = 0; row < boardlen; row++) {
        for (let col = 0; col < boardwid; col++) {
            if (placeItem(row, col, player)) {
                const result = alphabeta(depth - 1, player === COMPUTER_HORIZONTAL ? PLAYER_VERTICAL : COMPUTER_HORIZONTAL, alpha, beta);
                removeItem(row, col, player); // Undo move after evaluating
                if (maximizingPlayer) { // Update scores based on maximizing/minimizing player
                    if (result.score > bestScore) {
                        bestScore = result.score;
                        bestRow = row;
                        bestCol = col;
                    }
                    alpha = Math.max(alpha, bestScore);
                } else {
                    if (result.score < bestScore) {
                        bestScore = result.score;
                        bestRow = row;
                        bestCol = col;
                    }
                    beta = Math.min(beta, bestScore);
                }
                if (alpha >= beta) { // Alpha-Beta Pruning
                    return { score: bestScore, row: bestRow, col: bestCol };
                }
            }
        }
    }
    return { score: bestScore, row: bestRow, col: bestCol };
};
```

Demonstration of the game features

Quick Demo of the Website

- **Interactive GUI:** User-friendly Interface for seamless gameplay.
- **Customizable Grid Size:** Allows users to adjust the matrix size as needed.
- **Strategic AI opponent:** Implements a Smart AI for competitive Gameplay.
- **Invalid move alerts:** Displays pop-ups for invalid user moves.
- **Turn Notifications:** Shows messages indicating the current turn and declares the winner at the end.
- **Game Reset Option:** Provides an option to reset the game for the fresh start.

Challenges Faced

- Ensuring moves were valid and did not overlap existing pieces.
  - Solution: The ‘placeItem’ function checks adjacency rules before placing a piece on the board.
- Recursive exploration of moves required undoing each placed piece after evaluation.
  - Solution: Implemented the ‘removeItem’ function to clean up after evaluating each branch.
- The computer's depth setting affected its performance and difficulty level.
  - Solution: Depth was set to 5 for a balance between challenging gameplay and reasonable response time.

Results of testing and user feedback

- Tested with different depths in min-max algorithm with alpha-beta pruning- AI is taking strategic moves
- Smooth gameplay
- Positive user feedback on AI responsiveness and Interactive GUI.

Improvements in Future:

- Expanding AI difficulty levels to cater to a broader range of players.
- Can include more efficient AI algorithms for more competitive play.
- Integrate machine learning to allow the AI to adapt and improve based on past games.
- Add 2 human player option for increased engagement.
- Optimize for mobile devices for broader accessibility.
- Deploy the website.
- Implement user authentication and authorization to scale the website effectively.

Thank You!