Project Report

Project Title:

AI-Based Sequence Game

Submitted By:

Muhammad Hasnain (22K-5099)

Ali Masood (22K-5127)

Syed Shees Ali (22K-5047)

Course:

Artificial Intelligence

Instructor:

Ms. Alishba Subhani

Submission Date:

11 May 2025

1. Introduction

This report documents the implementation of an AI opponent for the Sequence board game using Flutter and the minimax algorithm with alpha-beta pruning. The AI is designed to make strategic decisions by evaluating possible moves, predicting player responses, and optimizing its gameplay to win the game.

1.1 Objectives

- Develop a functional Sequence game board with a 10x10 grid.
- Implement a two-player mode (Human vs. AI).
- Design an AI opponent using minimax with alpha-beta pruning.
- Ensure the AI makes strategically optimal moves by evaluating board states.
- Provide visual feedback for player and AI moves.

2. Game Mechanics

2.1 Board Setup

- The game uses a 10×10 grid with playing cards in each cell.
- Wild spots (corners) act as free spaces for sequences.
- Players place chips on matching card positions.

2.2 Winning Conditions

- A player wins by forming two sequences of five chips (horizontally, vertically, or diagonally).
- Sequences can include wild spots (corners).

2.3 Card Mechanics

- Standard Cards: Must match the board position (e.g., 'Ace of Spades' must be placed on the corresponding cell).
- Jack Cards:
  - Single-Eyed Jacks (Spades, Hearts): Remove an opponent's chip.
  - Double-Eyed Jacks (Clubs, Diamonds): Place a chip on any empty space.

3. AI Implementation

3.1 Minimax Algorithm

- The AI uses minimax with alpha-beta pruning to evaluate moves:
  - Maximizing Player (AI): Chooses moves that maximize its score.
  - Minimizing Player (Human): Simulates the opponent's best responses.
- Depth-Limited Search: Evaluates moves 3 steps ahead (depth=3).

3.2 Alpha-Beta Pruning

- Optimizes minimax by cutting off irrelevant branches.
- Reduces computation time while maintaining optimal decisions.

3.3 Move Evaluation

- The AI scores moves based on:
  - Completed Sequences (+1000 for AI, -1000 for player)
  - Potential Sequences (+100 for AI, -100 for player)
  - Center Control (+10 per controlled center cell)
  - Jack Cards in Hand (+50 for AI, -50 for player)

3.4 AI Decision-Making Process

- Generate Possible Moves: For each card in hand, find all valid placements.
- Simulate Moves: Temporarily apply each move.
- Evaluate Board State: Score the board using _evaluateBoard().
- Undo Moves: Revert the board to its original state.
- Select Best Move: Choose the move with the highest minimax score.

4. Key Features

4.1 Player Interaction

- Highlight Valid Moves: Shows possible placements for the selected card.
- Card Dropping: Automatically discards unplayable cards.
- Move History: Tracks and displays recent moves.

4.2 AI Feedback

- Visual Indication: Shows AI's last move.
- Card Played Popup: Displays which card the AI used.
- Sequence Detection: Marks completed sequences.

4.3 Game State Management

- Turn-Based System: Alternates between player and AI.
- Deck & Hand Management: Draws cards when needed.
- Win Detection: Checks for two sequences to declare a winner.

5. Technical Implementation

5.1 Data Structures

- boardState: 2D list tracking chip placements ('player', 'ai', 'empty', 'wild').
- sequenceMarkers: 2D list marking positions in completed sequences.
- deck: List of remaining cards.
- playerHand & aiHand: Lists of current cards.

5.2 Key Functions

Function	Purpose
_findBestMove()	Selects the best move using minimax.
_minimax()	Recursively evaluates moves.
_evaluateBoard()	Scores the current board state.
checkForSequence()	Detects completed sequences.
placeChip()	Handles player moves.
aiMakeMove()	Executes AI's turn.

5.3 UI Components

- GridView: Displays the 10×10 board.
- PlayingCard Widget: Renders card visuals.
- AlertDialog: Shows game status (win/loss, AI moves).
- AnimatedSwitcher: Smooth transitions for move history.

6. Challenges & Solutions

6.1 Performance Optimization

- Problem: Minimax can be computationally expensive.
- Solution: Alpha-beta pruning reduces unnecessary evaluations.

6.2 Sequence Detection

- Problem: Accurately detecting sequences with wild spots.
- Solution: Special checks for corner wild spots in checkForSequence().

6.3 AI Strategy

- Problem: AI sometimes makes suboptimal defensive moves.
- Solution: Added evaluation for potential sequences and center control.

7. Future Improvements

- Adaptive Difficulty: Adjust AI depth based on game phase.
- Better Heuristics: Improve sequence potential detection.
- Multiplayer Mode: Online PvP support.
- Animations: Smoother chip placement effects.
- Machine Learning: Train AI on real gameplay data.

8. Conclusion

This implementation provides a strong AI opponent for the Sequence game using minimax with alpha-beta pruning. The AI effectively evaluates board states, predicts player moves, and makes strategic decisions. Future enhancements could further optimize performance and gameplay experience.

GitHub Repository

https://github.com/muhammadhasnain115/Sequence-game-using-flutter-