# Project Title: Mufti Chess: A Four-Queen Chess Variant

Submitted By: Munnazzar Shahzad, Asfand Khanzada, Hanzala Umer

Course: AI

Instructor: Sir Abdullah Yaqoob

Submission Date: March 10, 2025

## 1. Executive Summary

Mufti Chess is an innovative twist on the classical game of chess, where each player begins with four queens instead of one. This modification introduces new strategic dynamics, demanding both human players and AI agents to adapt to a more offensive, fast-paced gameplay environment. The project involved implementing this variant and designing an AI opponent using the Negamax algorithm with Alpha-Beta pruning. A custom heuristic evaluation function was developed to assess the board state effectively in the context of increased queen power, leading to a challenging and intelligent AI opponent.

## 2. Introduction

Background:  
Chess has long served as a benchmark for AI research due to its complexity and depth. This project explores a unique variation, Mufti Chess, which alters the standard piece configuration by providing each player with four queens. This not only significantly boosts the tactical options available but also increases the game's difficulty, making it an ideal candidate for implementing and testing AI techniques.  
  
Objectives of the Project:  
- Design and implement Mufti Chess with custom rule modifications.  
- Develop a game-playing AI using the Negamax algorithm with Alpha-Beta pruning.  
- Create a heuristic evaluation function suitable for multi-queen dynamics.  
- Test the AI's effectiveness against human players.  
- Build an interactive, GUI-based application for gameplay.

## 3. Game Description

Original Game Rules:  
Standard chess is played on an 8×8 board, with each player controlling 16 pieces including a king, queen, rooks, knights, bishops, and pawns. The objective is to checkmate the opponent’s king using strategic piece movement and positioning.  
  
Innovations and Modifications:  
- Each player starts with four queens, replacing two bishops and one knight.  
- The board remains standard, but the starting piece positions are updated.  
- The presence of additional queens increases complexity, encourages faster attacks, and changes traditional opening and endgame strategies.

## 4. AI Approach and Methodology

AI Techniques Used:  
- Negamax Algorithm: A variant of Minimax optimized for zero-sum games like chess. This simplifies the logic by treating the opponent's utility as the negative of the current player's.  
- Alpha-Beta Pruning: Integrated into Negamax to eliminate branches that won't affect the final decision, greatly improving computational efficiency.  
  
Algorithm and Heuristic Design:  
The heuristic evaluation function used for move scoring includes:  
- Material Score: Evaluates material difference based on customized piece values considering multiple queens.  
- Mobility: Rewards positions offering more legal moves, especially for queens.  
- King Safety: Penalizes exposed kings, factoring in enemy queen proximity.  
- Positional Bonuses: Encourages control of the center, open files, and coordinated attacks.  
- Pawn Structure Penalties: Considers isolated, doubled, or backward pawns to reflect positional weaknesses.  
  
AI Performance Evaluation:  
- Performance was evaluated through win rates, response times, and strategic quality of moves.  
- The Negamax AI averaged 1.5–2 seconds per move at search depth 3–4.  
- Against human players, the AI demonstrated strong offensive play, maintaining a 30% win rate on average.

## 5. Game Mechanics and Rules

Modified Game Rules:  
- Each player starts with 4 queens.  
- Two bishops and one knight are replaced to accommodate additional queens.  
- All standard movement, check/checkmate, and draw conditions remain valid.  
  
Turn-based Mechanics:  
- White moves first, followed by alternating turns.  
- Legal move generation is strictly enforced by the engine.  
- The game ends in checkmate, draw, or resignation.  
  
Winning Conditions:  
- Checkmate: The standard win condition.  
- Draws: Can occur through stalemate, repetition, or the 50-move rule.  
- Resignation: Players may resign at any point.

## 6. Implementation and Development

Development Process:  
- Week 1–2: Game rule modification and board setup logic.  
- Week 3: Implementation of move generation and validation using python-chess.  
- Week 4–5: Design and implementation of the Negamax algorithm with Alpha-Beta pruning.  
- Week 6: Development of the heuristic evaluation function.  
- Week 7: GUI integration using Cute Chess.  
- Week 8: Testing, optimization, and documentation.  
  
Programming Languages and Tools:  
- Language: Python  
- Libraries:  
 - python-chess: Handling board logic, move validation  
 - NumPy: For numerical evaluation and board scoring  
 -Cute Chess: For GUI design and interaction  
- Tools:  
 - GitHub: Version control and collaboration  
  
Challenges Encountered:  
- Evaluating board positions accurately with multiple queens required custom weighting.  
- Search space grew quickly due to increased piece activity, needing efficient pruning.  
- Maintaining GUI responsiveness while executing AI logic in the background.

## 7. Team Contributions

- Munnazzar Shahzad:  
 - Implemented the Negamax algorithm with Alpha-Beta pruning.  
 - Designed and optimized the evaluation heuristic.  
- Asfand Khanzada:  
 - Developed the modified game logic and board setup.  
 - Designed and implemented the GUI using Cute Chess and integrated it with the backend.  
- Hanzala Umer:  
 - Assisted in GUI testing and refining the evaluation function.  
 - Helped with code refactoring and documentation.

## 8. Results and Discussion

- The AI showed consistent performance with a 30% win rate over 50+ games against casual players.  
- Decision-making remained within 1.8 seconds per move at 3-ply depth.  
- The evaluation function successfully prioritized mobility and king safety, which are critical in a multi-queen environment.  
- Game outcomes were more aggressive and shorter than standard chess due to the offensive nature of extra queens.

## 9. References

- Russell, S. J., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). Pearson.  
- Shannon, C. E. (1950). Programming a Computer for Playing Chess. Philosophical Magazine, 41(314).  
- Python Chess Documentation: https://python-chess.readthedocs.io/  
- Pygame Documentation: https://www.pygame.org/docs/  
- MuftiChess GitHub Repository: https://github.com/Munnazzar/MuftiChess