**Project Title: Bishop-Heavy Chess AI**

**Submitted By:** AAWAIZ - 22i-0845

TAHA KHAN - 22i-2335

**Course:** AI

**Instructor:** Sir Fahad Sherwani

1. **Project Overview**

**Project Topic:** Bishop-Heavy Chess. This project explores a Chess variant played on a standard 8x8 board, but with a significantly altered starting piece composition for each player: 1 King, 1 Queen, 1 Rook, 1 Knight ("horse"), and 4 Bishops, alongside the standard 8 Pawns.

**Objective:** The primary objective is to fully define the starting positions and confirm the rules for this variant. Subsequently, the project aims to implement the game logic and develop a competent AI opponent using the Minimax algorithm optimized with Alpha-Beta Pruning. A key focus will be designing a heuristic evaluation function tailored to the unique strategic landscape created by the abundance of Bishops and scarcity of Rooks and Knights.

1. **Game Description**

**Original Game Background:** Standard chess is a two-player strategy board game played on an 8x8 grid (chessboard). Each player begins with 16 pieces: one King, one Queen, two Rooks, two Knights, two Bishops, and eight Pawns. Pieces move in specific ways; the objective is to checkmate the opponent's King, putting it under an inescapable threat of capture. Key rules include piece movement, capturing, check, checkmate, stalemate, castling, en passant, and pawn promotion.

**Innovations Introduced:**

**Modified Piece Set:** The standard back-rank pieces (excluding King and Queen) are replaced. Each player starts with only 1 Rook, 1 Knight, and is given 4 Bishops. The King, Queen, and 8 Pawns remain standard.

**Starting Position:** A defined starting setup is required. A plausible arrangement preserving King/Queen centrality and symmetry could be:

White (Row 1): R(a1), B(b1), B(c1), Q(d1), K(e1), B(f1), B(g1), N(h1). Pawns on Row 2.

Black (Row 8): R(a8), B(b8), B(c8), Q(d8), K(e8), B(f8), B(g8), N(h8). Pawns on Row 7.

**Gameplay Impact:** This setup drastically changes the game's strategic nature.

**Bishop Dominance:** With four bishops (two on light squares, two on dark squares), control of diagonals becomes extremely important. Opening the center and clearing paths for bishops is critical.

**Reduced Rook/Knight Influence:** Fewer open file contests (due to only one Rook) and fewer knight forks/complex maneuvers. The single Rook and Knight become unique, valuable pieces whose trade requires careful consideration.

**Altered Piece Values:** The relative value of pieces shifts. Bishops likely increase in value, and the scarcity might slightly increase the value of the single Rook and Knight compared to standard chess.

**Castling Limitation:** Castling is only possible on one side (Queenside if Rook is on a1/a8, Kingside if Rook is on h1/h8). Assuming R on a1/a8, only Queenside castling is possible.

1. **AI Approach and Methodology**

**AI Techniques to be Used:**

**Minimax Algorithm:** The standard search algorithm for two-player, zero-sum games like chess and its variants.

**Alpha-Beta Pruning:** A crucial optimization technique to prune the search tree, making deeper searches computationally feasible. Essential for chess.

**Heuristic Design:** The evaluation function is critical and must be adapted from standard chess:

**Material Balance:** Re-evaluate standard piece values. Suggestion: P=1, N=3.25-3.75, B=3.5-4.0, R=5.5-6.0, Q=9. Consider bonuses for having active bishop pairs on different color complexes. The value of the unique Rook and Knight might be higher due to their scarcity.

**Positional Factors:** Emphasize control of open diagonals, center control, pawn structure (especially avoiding blocked pawns that restrict bishops), piece mobility (especially for bishops), King safety.

**Piece-Specific Factors:** Bonuses for active bishops, penalties for bishops blocked by own pawns, strategic positioning of the unique Rook (e.g., on open/semi-open files) and Knight (e.g., on outposts).

**Complexity Analysis:**

The game tree complexity is comparable to standard chess. The state space is identical, and the branching factor is similar, though piece distribution changes move possibilities.

The primary challenge is developing and tuning the heuristic evaluation function to accurately reflect the altered strategic values and dynamics of this variant.

Implementation complexity involves correctly handling all chess rules (including castling based on the defined setup) and the AI search algorithm.

**Performance Measurement:**

**Measurement Metrics:**

* Classification Accuracy: Optimal moves by AI vs total moves played.
* Confusion Matrix: Visual representation of correct vs incorrect move predictions.
* Win-rate Analysis: AI performance against baseline chess engines or intermediate-level players.

**Minimum Expected Accuracy:**

* 70% optimal move prediction accuracy.
* At least 50% win-rate against baseline chess engines.

**Risks and Dependencies:**

**Risks:**

* Complexity of heuristic tuning due to altered strategic conditions.
* Computational limitations potentially affecting depth of Alpha-Beta searches.
* Unforeseen edge-cases due to unique rule set.

**Mitigation Strategies:**

* Incremental implementation and thorough testing.
* Begin with simpler heuristic parameters, gradually increasing complexity.
* Comprehensive validation of rules and AI behavior.

**Dependencies:**

* Python programming language environment.
* python-chess library.
* Optional GUI tools such as Pygame.

**Run Performance Checks:**

* Periodic test matches against baseline chess engines.
* Regularly generate confusion matrices to assess and visualize accuracy.
* Monitor classification accuracy continuously to fine-tune heuristic parameters.

1. **Game Rules and Mechanics**

**Modified Rules:**

**Piece Set:** Each player begins with 1 King, 1 Queen, 1 Rook, 1 Knight, 4 Bishops, 8 Pawns.

**Starting Position:** White: R(a1), B(b1), B(c1), Q(d1), K(e1), B(f1), B(g1), N(h1). Black: Mirrored on row 8.

**Castling:** Based on the above setup, only Queenside castling is possible (King moves to c1/c8, Rook moves to d1/d8), provided the standard castling conditions are met (squares are clear and unattacked, King and Rook haven't moved).

**All Other Rules:** All other standard FIDE chess rules apply, including piece movements, captures, check, checkmate, stalemate, pawn promotion (can promote to Q, R, N, or B), and en passant.

**Winning Conditions:** Checkmate the opponent's King. Standard draw conditions (stalemate, threefold repetition, 50-move rule, insufficient material) also apply.

**Turn Sequence:** White moves first, players alternate turns.

1. **Implementation Plan**

**Programming Language:** Python due to readily available libraries like python-chess, facilitating development).

**Libraries and Tools:**

**python-chess library:** It handles board representation, move generation, move validation, and checks for standard rules (checkmate, stalemate, etc.), greatly simplifying implementation. You would primarily focus on the AI search and heuristic function.

Pygame (optional)

1. **References**

[FIDE Laws of Chess - Official standard chess rules]

Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach (4th ed.). (Relevant chapters on adversarial search, Minimax, Alpha-Beta).

[Documentation for the python-chess library: <https://python-chess.readthedocs.io/>]