

**Project Title**: Strategic Mancala AI Game

**Made by**: Aaminah Binte Farooq

**Department of Cyber Security**

**Roll no**: 22k-4671

**Submitted to**: Ms. Mehak Mazhar

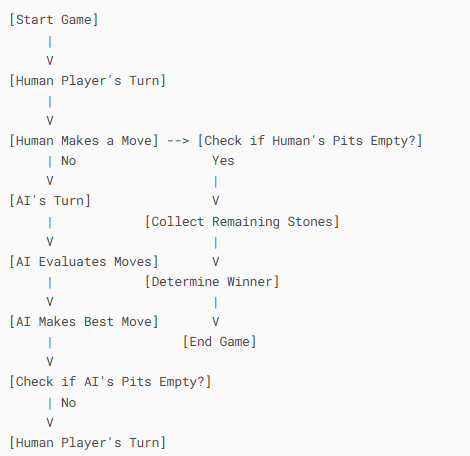
**Dated:** 11 May 2025

**GITHUB Link:** https://github.com/aaminahfarooq/AI-Strategic-MANCALA-Game

**STRATEGIC MANGALA AI GAME WITH MIN/MAX ALGORITHM AND ALPHA-BETA PRUNING**

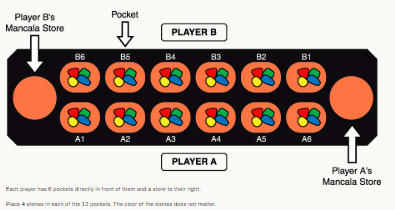
# **1. Introduction**

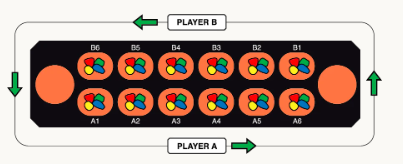
# This project aims to create a strategic AI opponent for the classic board game Mancala using artificial intelligence techniques. The AI uses the Min/max algorithm with Alpha-Beta pruning to simulate optimal decision-making, providing a challenging experience for human players. **2. Game Description** Mancala or African chess is a two-player, turn-based board game in which players take turns sowing stones across the board. The goal is to collect the most stones in your store by the end of the game. The game ends when all of a player's pits are empty. Mancala consists of 14 plates, two of which are scoring plates and the other 12 are non-scoring plates allocated to two players.  In this project, an AI opponent was introduced, which uses the Min/max algorithm to evaluate potential game states and select the optimal move. Alpha-Beta pruning was implemented to enhance the Min/max algorithm's efficiency by eliminating unnecessary branches in the game tree. **3. Workflow Overview**

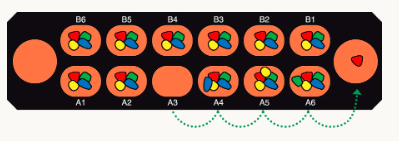


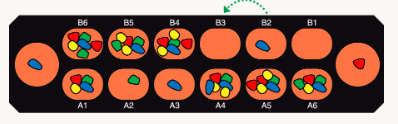
1. The game starts with the human player's turn
2. After each move, the game checks if the current player's pits are empty
3. If pits are empty, remaining stones go to stores and winner is determined
4. If not, turn alternates between players
5. AI always uses Min/max to select the best move

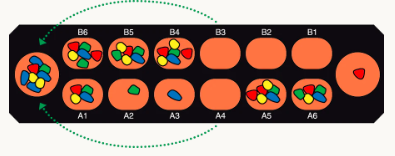
# **4. Model Diagram**









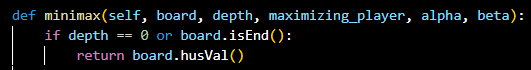


# **5. Tools and Technologies used**

|  |  |
| --- | --- |
| **Tool/Technology** | **Purpose** |
| Python 3.x | Programming Language |
| Tkinter | GUI Development |
| Minimax Algorithm | AI decision-making |
| Alpha-Beta Pruning | Optimization of AI |

# **6. AI Implementation**

* The AI opponent in this project is powered by the ***Minimax algorithm****,* a classic recursive decision-making technique used in two-player games. The AI explores potential moves and simulates how the human player might respond in order to choose the most favorable outcome for itself.
* To optimize performance, ***Alpha-Beta pruning*** is used. This technique eliminates branches in the game tree that do not need to be explored, significantly improving the efficiency of the algorithm.
* Due to the complexity and branching factor of the game, it is not feasible to simulate every possible move until the end of the game. Instead, the AI uses a ***heuristic*** function to estimate the value of non-terminal game states.

In this project, the heuristic function is defined as:

 This line is executed when: The recursion reaches the depth limit (e.g., depth == 0), Or The game reaches an end state. It calls board.husVal(), which is your heuristic evaluation function:

# **7. Results and Observations** The game was tested with both single-player and AI opponent modes. The AI was able to make intelligent decisions based on the board state, resulting in challenging gameplay for the human player. The use of Alpha-Beta pruning significantly reduced the time taken for the AI to evaluate the game tree, allowing for a smoother gaming experience.

# **8. Conclusion**

This project successfully implemented an AI opponent for Mancala using the Minimax algorithm with Alpha-Beta pruning. Future improvements could involve integrating reinforcement learning to allow the AI to learn and adapt over time, as well as adding multiplayer functionality.

# **9. References** - Mancala game rules: https://en.wikipedia.org/wiki/Mancala - Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig