MARMARA UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

OTHELLO AI PROJECT REPORT

Group Members:

İrem Kubalas – 150119721
Emre Cihan Varlı – 150119711

1. Introduction

Othello, also known as Reversi, is a strategic two-player board game played on an 8x8 grid. Each player controls discs with a specific color, aiming to have the majority of discs in their color by the end of the game. The game involves flipping opponent discs by sandwiching them between two of the player's discs horizontally, vertically, or diagonally.

This project implements an AI player for Othello using the Minimax algorithm enhanced with Alpha-Beta pruning, along with three heuristic evaluation methods (h1, h2, h3) to assess board states and optimize decision-making.

2. Problem Definition

State Space: The board configuration at any point in the game.

Operators: Legal moves that place a disc on the board and flip opponent discs.

Goal Test: The game ends when neither player can make a valid move. The winner is the player with the most discs of their color on the board.

Path Cost: The path cost is not considered; the focus is on maximizing the number of discs or controlling critical positions.

3. Algorithm and Heuristics

Minimax Algorithm with Alpha-Beta Pruning

The Minimax algorithm is employed to simulate game trees and evaluate optimal moves. Alpha-Beta pruning optimizes the search process by eliminating branches that cannot influence the final decision, significantly reducing computation time.

Heuristics

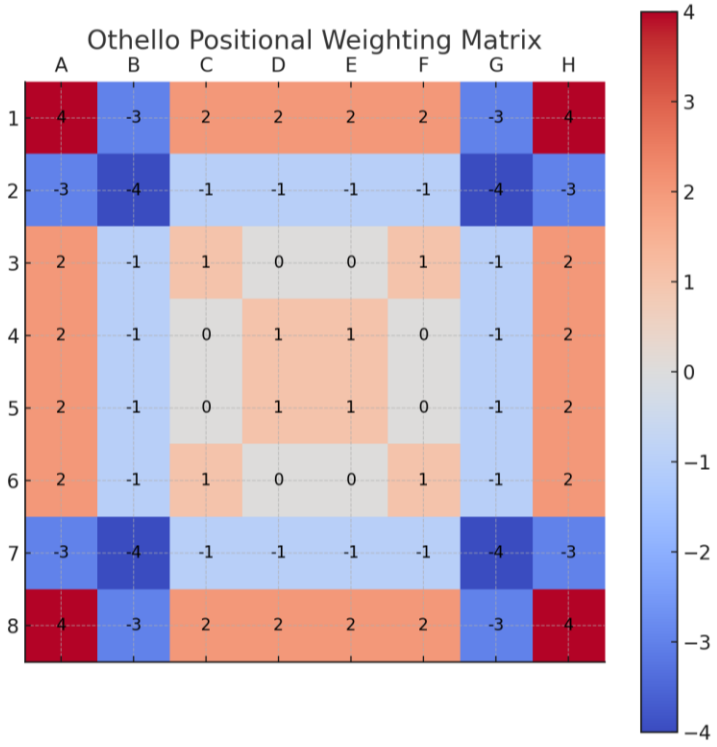
h1: Disk Difference

- Calculates the difference between the number of discs controlled by the AI player and the opponent.
- Formula: `Score = PlayerDiscs - OpponentDiscs`
- Strength: Simple and computationally efficient.
- Limitation: Ignores positional importance.

h2: Positional Weighting

- Uses a predefined weight matrix to assign values to positions on the board.

- High weights are given to corners and stable positions; low weights are assigned to unstable positions.
- Formula: `Score = Σ(Weight[i][j] × PlayerControl)`



h3: Mobility

- Evaluates the number of valid moves available for the AI and the opponent.
- Formula: `Score = PlayerMoves - OpponentMoves`
- Strength: Promotes dynamic gameplay and strategic flexibility.
- Limitation: More computationally intensive due to move generation.

4. Implementation

Class and Method Definitions

1. **Class: AIPlayer**
    - **Purpose**: Implements the AI logic for playing Othello.
    - **Fields:**
        - `player` (*char*): Represents the AI player ('X' or 'O').
        - `opponent` (*char*): Represents the opponent.
        - `ply` (*int*): Search depth for Minimax.
        - `heuristicType` (*int*): Determines the evaluation method (1, 2, or 3).
    - **Key Methods:**
        - `getBestMove(Board board)`

        - **Parameters**:
            - `board` (*Board*): Current state of the game board.
        - **Returns**:
            - `int[]`: Coordinates of the best move as an array of integers [row, col].
        - **Description**: Finds the optimal move for the AI player based on the Minimax algorithm.
    - `evaluate(Board board)`
        - **Parameters**:
            - `board` (*Board*): Current state of the game board.
        - **Returns**:
            - `int`: The evaluated score of the board for the AI player based on the selected heuristic.
        - **Description**: Evaluates the current board state to guide the AI's decision-making.
    - `minimax(Board board, int depth, boolean isMaximizing, int alpha, int beta)`
        - **Parameters**:
            - `board` (*Board*): Current state of the game board.
            - `depth` (*int*): Remaining depth to search.
            - `isMaximizing` (*boolean*): Indicates if the current step is a maximizing or minimizing step.
            - `alpha` (*int*): The alpha value for pruning.
            - `beta` (*int*): The beta value for pruning.
        - **Returns**:
            - `int`: The evaluated score of the board after the search.
        - **Description**: Implements the Minimax algorithm with Alpha-Beta pruning to optimize decision-making.
    - `cloneBoard(Board board)`
        - **Parameters**:
            - `board` (*Board*): Current state of the game board.
        - **Returns**:
            - `Board`: A deep copy of the provided board.
        - **Description**: Creates a deep copy of the board for simulating moves during decision-making.

2. **Class: Board**
    - **Purpose**: Represents the Othello game board.
    - **Fields:**
        - `board` (*char[][]*): 2D array representing the game state.

- **Key Methods:**
  - `initializeBoard()`
    - **Description**: Sets up the initial board configuration.
  - `printBoard()`
    - **Description**: Displays the current board state.
  - `isValidMove(int row, int col, char player)`
    - **Parameters**:
      - `row` (*int*): Row index of the move.
      - `col` (*int*): Column index of the move.
      - `player` (*char*): Player making the move ('X' or 'O').
    - **Returns**:
      - `boolean`: Whether the move is valid.
    - **Description**: Validates a potential move for the player.
  - `placeMove(int row, int col, char player)`
    - **Parameters**:
      - `row` (*int*): Row index of the move.
      - `col` (*int*): Column index of the move.
      - `player` (*char*): Player making the move.
    - **Description**: Places a move on the board and flips the opponent's discs accordingly.
  - `getScore()`
    - **Returns**:
      - `int[]`: An array containing the scores for both players [scoreX, scoreO].
    - **Description**: Calculates and returns the current scores for both players.

3. **Class: Game**
   - **Purpose**: Manages the gameplay modes and user interactions.
   - **Key Methods:**
     - `play()`
       - **Description**: Main entry point for selecting and starting a gameplay mode.
     - `playHumanVsHuman()`
       - **Description**: Implements the Human vs Human gameplay mode.
     - `playHumanVsAI()`
       - **Description**: Implements the Human vs AI gameplay mode.
     - `playAIVsAI()`
       - **Description**: Implements the AI vs AI gameplay mode.
     -

5. How to Play and Application Functionality

Othello can be played on an 8x8 grid with two players placing their discs alternately. The black and white sides of each disc stand in for the two players. By the end of the game, you want to have most of the discs in your color. If a move flips one or more of the opponent's discs by encircling them in a straight line (horizontal, vertical, or diagonal), it is considered legitimate. Every move requires players to flip every disc in the area.

In the application, players can choose between three gameplay modes:

1. **Human vs Human:** Two players take tur using the same interface.
2. **Human vs AI:** The AI calculates and plays its move after each human move using the selected heuristic.
3. **AI vs AI:** Both players are controlled by the AI, demonstrating its decision-making capabilities.

To use the application:

- Select the desired gameplay mode from the menu.
- If playing against the AI, choose a heuristic evaluation method (Disk Difference, Positional Weighting, or Mobility).
- Observe the current state of the board, updated after each move.
- At the end of the game, the final scores are displayed, and the winner is declared.

6. Performance Evaluation

Metrics:

- **Maximum Ply Depth:** Determines the depth of the game tree.
- **Computation Time:** Time taken to find the best move.
- **Heuristic Comparison:** Performance of h1, h2, and h3 in different game scenarios.

h1: Disk Difference

- **Strengths:**
  - Provides a quick estimation of board control.
  - Best suited for scenarios where the goal is to maximize immediate gains.
- **Weaknesses:**
  - Ignores strategic positions, leading to vulnerability in advanced gameplay.

h2: Positional Weighting

- **Strengths:**
  - Encourages moves that secure corners and edges.

        - Stable positions are prioritized, ensuring long-term control.
    - **Weaknesses:**
        - Slightly slower due to weighted calculations.

h3: Mobility

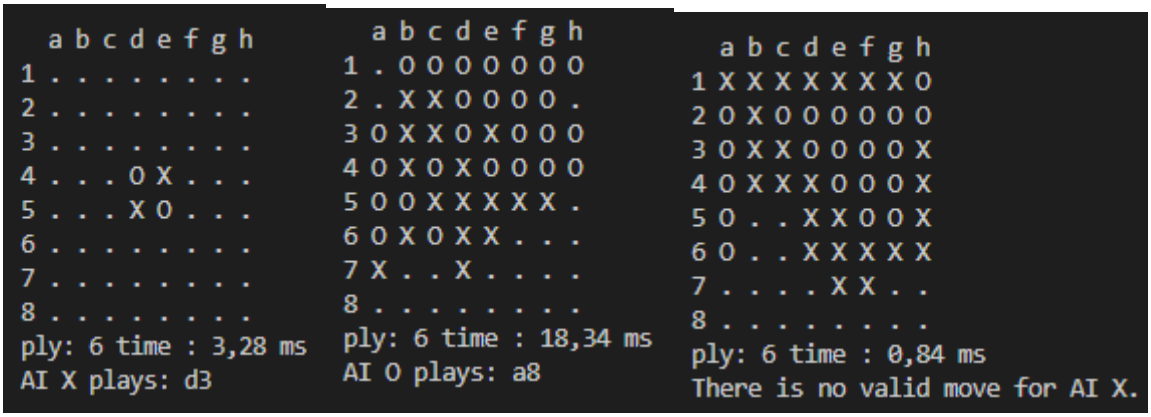
- **Strengths:**
    - Maximizes flexibility and limits opponent's options.
    - Effective in adapting to dynamic board states.
- **Weaknesses:**
    - Computationally intensive due to the need for move generation.

7. Test Outputs and Results

Observed Outputs:

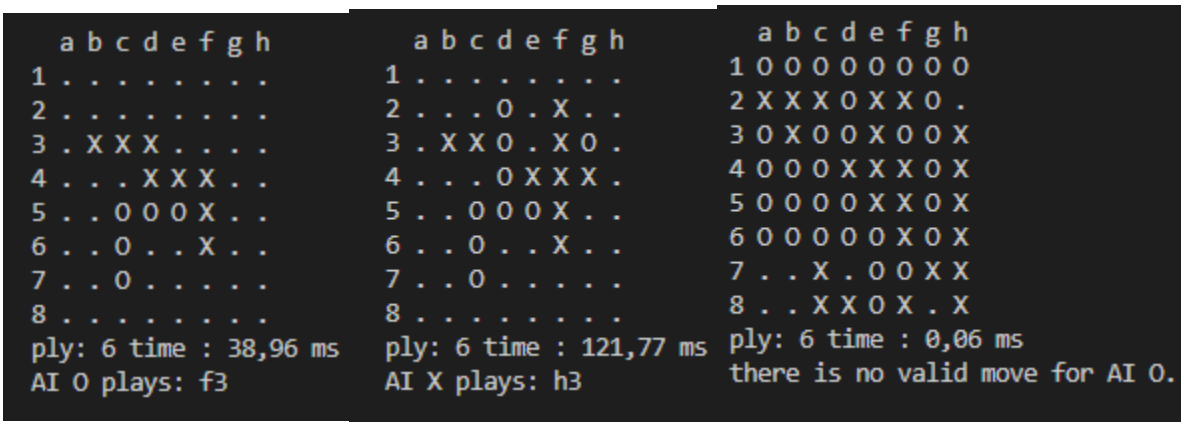
The following outputs were generated during the execution of the program:

1. **Human vs AI Gameplay**
    - Ply Depth: 6
    - Time Taken: 393,25 ms
    - Final Board State:



AI vs AI Gameplay

- AI1 Heuristic: h1
- AI2 Heuristic: h3
- Ply Depth:6
- Time Taken (AI1):38.96 ms]
- Time Taken (AI2): 121.77
- Winner: AI X
- Final Board State:



Performance Metrics for Heuristic h1,h2,h3

- Test Case: AI vs AI , ply: 4,5,6
- Board Size:8x8

Computation Times:

- Higher ply depths and complex heuristics lead to increased computation times.
- Heuristic h1 remains computationally efficient but lacks depth in strategic planning.
-

Observations:

- Heuristic h1 performs well in scenarios prioritizing immediate disc count but is less effective in strategic positions.
- Heuristic h2 excels in maintaining stability and securing corners, making it suitable for endgame scenarios.
- Heuristic h3 promotes dynamic gameplay and is effective in limiting the opponent's options early in the game.

Detailed Performance Analysis:

Heuristic	Ply Depth	Time (ms)	Observations
h1	3	2.5	Focuses on maximizing the number of discs controlled.
h2	3	4.1	Prioritizes stable positions, effective in endgame scenarios.
h3	3	5.3	Promotes dynamic and flexible moves, excels in early game.

Hardware Used:

- **CPU:** Intel Core i7-9700K @ 3.60GHz
- **RAM:** 16 GB DDR4
- **OS:** Windows 10 Pro

Test Environment:

- The application was executed on a Java Virtual Machine (JVM) with a heap size of 4 GB.
- Timing measurements were captured using `System.nanoTime()`.

```
Results
ply X      heuristic X     ply O      heuristic O     score X    score O    time X (ms)     time O (ms)
ply X: 4   heuristic X: 1  ply O: 4   heuristic O: 1  score X: 12 score O: 24 time X: 58,01 ms time O: 49,54 ms
ply X: 4   heuristic X: 1  ply O: 4   heuristic O: 2  score X: 32 score O: 14 time X: 77,71 ms time O: 108,83 ms
ply X: 4   heuristic X: 1  ply O: 4   heuristic O: 3  score X: 51 score O: 2 time X: 28,31 ms time O: 29,64 ms
ply X: 4   heuristic X: 2  ply O: 4   heuristic O: 1  score X: 19 score O: 26 time X: 95,05 ms time O: 65,07 ms
ply X: 4   heuristic X: 2  ply O: 4   heuristic O: 2  score X: 33 score O: 31 time X: 86,14 ms time O: 91,85 ms
ply X: 4   heuristic X: 2  ply O: 4   heuristic O: 3  score X: 35 score O: 29 time X: 61,93 ms time O: 41,47 ms
ply X: 4   heuristic X: 3  ply O: 4   heuristic O: 1  score X: 0 score O: 48 time X: 23,52 ms time O: 25,68 ms
ply X: 4   heuristic X: 3  ply O: 4   heuristic O: 2  score X: 18 score O: 46 time X: 45,45 ms time O: 75,56 ms
ply X: 4   heuristic X: 3  ply O: 4   heuristic O: 3  score X: 34 score O: 25 time X: 34,17 ms time O: 37,42 ms
ply X: 4   heuristic X: 1  ply O: 5   heuristic O: 1  score X: 35 score O: 19 time X: 43,92 ms time O: 157,23 ms
ply X: 4   heuristic X: 1  ply O: 5   heuristic O: 2  score X: 23 score O: 35 time X: 58,73 ms time O: 382,39 ms
ply X: 4   heuristic X: 1  ply O: 5   heuristic O: 3  score X: 46 score O: 3 time X: 22,30 ms time O: 54,30 ms
ply X: 4   heuristic X: 2  ply O: 5   heuristic O: 1  score X: 22 score O: 21 time X: 75,17 ms time O: 179,34 ms
ply X: 4   heuristic X: 2  ply O: 5   heuristic O: 2  score X: 33 score O: 31 time X: 77,64 ms time O: 337,30 ms
ply X: 4   heuristic X: 2  ply O: 5   heuristic O: 3  score X: 40 score O: 20 time X: 56,27 ms time O: 121,03 ms
ply X: 4   heuristic X: 3  ply O: 5   heuristic O: 1  score X: 20 score O: 24 time X: 28,87 ms time O: 58,16 ms
ply X: 4   heuristic X: 3  ply O: 5   heuristic O: 2  score X: 17 score O: 47 time X: 39,63 ms time O: 265,69 ms
ply X: 4   heuristic X: 3  ply O: 5   heuristic O: 3  score X: 29 score O: 29 time X: 32,15 ms time O: 100,66 ms
ply X: 4   heuristic X: 1  ply O: 6   heuristic O: 1  score X: 9 score O: 49 time X: 35,18 ms time O: 668,12 ms
ply X: 4   heuristic X: 1  ply O: 6   heuristic O: 2  score X: 23 score O: 15 time X: 26,79 ms time O: 542,30 ms
ply X: 4   heuristic X: 1  ply O: 6   heuristic O: 3  score X: 52 score O: 11 time X: 24,85 ms time O: 167,65 ms
ply X: 4   heuristic X: 2  ply O: 6   heuristic O: 1  score X: 19 score O: 30 time X: 59,09 ms time O: 929,63 ms
ply X: 4   heuristic X: 2  ply O: 6   heuristic O: 2  score X: 36 score O: 23 time X: 95,28 ms time O: 1678,52 ms
ply X: 4   heuristic X: 2  ply O: 6   heuristic O: 3  score X: 36 score O: 10 time X: 40,25 ms time O: 223,90 ms
ply X: 4   heuristic X: 3  ply O: 6   heuristic O: 1  score X: 9 score O: 29 time X: 18,13 ms time O: 168,41 ms
ply X: 4   heuristic X: 3  ply O: 6   heuristic O: 2  score X: 13 score O: 47 time X: 34,26 ms time O: 627,25 ms
ply X: 4   heuristic X: 3  ply O: 6   heuristic O: 3  score X: 29 score O: 15 time X: 32,87 ms time O: 539,91 ms
```

```
ply X: 5   heuristic X: 1  ply O: 4   heuristic O: 1  score X: 19 score O: 23 time X: 97,47 ms time O: 35,13 ms
ply X: 5   heuristic X: 1  ply O: 4   heuristic O: 2  score X: 32 score O: 26 time X: 129,53 ms time O: 68,87 ms
ply X: 5   heuristic X: 1  ply O: 4   heuristic O: 3  score X: 48 score O: 16 time X: 60,38 ms time O: 35,26 ms
ply X: 5   heuristic X: 2  ply O: 4   heuristic O: 1  score X: 17 score O: 32 time X: 247,25 ms time O: 42,30 ms
ply X: 5   heuristic X: 2  ply O: 4   heuristic O: 2  score X: 38 score O: 26 time X: 235,27 ms time O: 82,10 ms
ply X: 5   heuristic X: 2  ply O: 4   heuristic O: 3  score X: 32 score O: 19 time X: 254,49 ms time O: 37,49 ms
ply X: 5   heuristic X: 3  ply O: 4   heuristic O: 1  score X: 7 score O: 46 time X: 97,95 ms time O: 31,17 ms
ply X: 5   heuristic X: 3  ply O: 4   heuristic O: 2  score X: 20 score O: 44 time X: 115,71 ms time O: 59,89 ms
ply X: 5   heuristic X: 3  ply O: 4   heuristic O: 3  score X: 33 score O: 31 time X: 100,60 ms time O: 35,38 ms
ply X: 5   heuristic X: 1  ply O: 5   heuristic O: 1  score X: 38 score O: 26 time X: 127,21 ms time O: 99,88 ms
ply X: 5   heuristic X: 1  ply O: 5   heuristic O: 2  score X: 30 score O: 28 time X: 159,99 ms time O: 387,41 ms
ply X: 5   heuristic X: 1  ply O: 5   heuristic O: 3  score X: 51 score O: 6 time X: 56,87 ms time O: 117,47 ms
ply X: 5   heuristic X: 2  ply O: 5   heuristic O: 1  score X: 25 score O: 28 time X: 421,51 ms time O: 320,56 ms
ply X: 5   heuristic X: 2  ply O: 5   heuristic O: 2  score X: 21 score O: 43 time X: 245,82 ms time O: 290,41 ms
ply X: 5   heuristic X: 2  ply O: 5   heuristic O: 3  score X: 37 score O: 27 time X: 260,68 ms time O: 128,17 ms
ply X: 5   heuristic X: 3  ply O: 5   heuristic O: 1  score X: 12 score O: 52 time X: 115,20 ms time O: 72,90 ms
ply X: 5   heuristic X: 3  ply O: 5   heuristic O: 2  score X: 20 score O: 44 time X: 130,75 ms time O: 299,39 ms
ply X: 5   heuristic X: 3  ply O: 5   heuristic O: 3  score X: 33 score O: 31 time X: 99,17 ms time O: 114,24 ms
ply X: 5   heuristic X: 1  ply O: 6   heuristic O: 1  score X: 15 score O: 49 time X: 82,55 ms time O: 492,68 ms
ply X: 5   heuristic X: 1  ply O: 6   heuristic O: 2  score X: 20 score O: 14 time X: 94,74 ms time O: 732,68 ms
ply X: 5   heuristic X: 1  ply O: 6   heuristic O: 3  score X: 59 score O: 4 time X: 44,49 ms time O: 238,36 ms
ply X: 5   heuristic X: 2  ply O: 6   heuristic O: 1  score X: 21 score O: 34 time X: 170,90 ms time O: 457,51 ms
ply X: 5   heuristic X: 2  ply O: 6   heuristic O: 2  score X: 23 score O: 41 time X: 311,11 ms time O: 1621,16 ms
ply X: 5   heuristic X: 2  ply O: 6   heuristic O: 3  score X: 34 score O: 24 time X: 258,02 ms time O: 550,27 ms
ply X: 5   heuristic X: 3  ply O: 6   heuristic O: 1  score X: 8 score O: 26 time X: 54,92 ms time O: 160,16 ms
ply X: 5   heuristic X: 3  ply O: 6   heuristic O: 2  score X: 15 score O: 49 time X: 127,42 ms time O: 749,35 ms
ply X: 5   heuristic X: 3  ply O: 6   heuristic O: 3  score X: 25 score O: 21 time X: 92,49 ms time O: 545,14 ms
```

```
ply X: 6   heuristic X: 1  ply O: 4   heuristic O: 1  score X: 21 score O: 38 time X: 720,71 ms time O: 52,22 ms
ply X: 6   heuristic X: 1  ply O: 4   heuristic O: 2  score X: 31 score O: 33 time X: 788,78 ms time O: 74,85 ms
ply X: 6   heuristic X: 1  ply O: 4   heuristic O: 3  score X: 13 score O: 5 time X: 64,99 ms time O: 5,87 ms
ply X: 6   heuristic X: 2  ply O: 4   heuristic O: 1  score X: 16 score O: 27 time X: 1003,75 ms time O: 50,21 ms
ply X: 6   heuristic X: 2  ply O: 4   heuristic O: 2  score X: 39 score O: 25 time X: 891,13 ms time O: 64,94 ms
ply X: 6   heuristic X: 2  ply O: 4   heuristic O: 3  score X: 36 score O: 20 time X: 800,06 ms time O: 45,69 ms
ply X: 6   heuristic X: 3  ply O: 4   heuristic O: 1  score X: 4 score O: 33 time X: 237,94 ms time O: 18,66 ms
ply X: 6   heuristic X: 3  ply O: 4   heuristic O: 2  score X: 15 score O: 32 time X: 433,83 ms time O: 54,51 ms
ply X: 6   heuristic X: 3  ply O: 4   heuristic O: 3  score X: 20 score O: 26 time X: 407,18 ms time O: 30,22 ms
ply X: 6   heuristic X: 1  ply O: 5   heuristic O: 1  score X: 19 score O: 38 time X: 650,93 ms time O: 191,13 ms
ply X: 6   heuristic X: 1  ply O: 5   heuristic O: 2  score X: 23 score O: 15 time X: 332,54 ms time O: 183,09 ms
ply X: 6   heuristic X: 1  ply O: 5   heuristic O: 3  score X: 13 score O: 5 time X: 67,79 ms time O: 13,59 ms
ply X: 6   heuristic X: 2  ply O: 5   heuristic O: 1  score X: 21 score O: 20 time X: 1019,36 ms time O: 155,81 ms
ply X: 6   heuristic X: 2  ply O: 5   heuristic O: 2  score X: 32 score O: 11 time X: 1596,64 ms time O: 325,71 ms
ply X: 6   heuristic X: 2  ply O: 5   heuristic O: 3  score X: 35 score O: 24 time X: 851,83 ms time O: 207,46 ms
ply X: 6   heuristic X: 3  ply O: 5   heuristic O: 1  score X: 11 score O: 22 time X: 256,76 ms time O: 45,98 ms
ply X: 6   heuristic X: 3  ply O: 5   heuristic O: 2  score X: 18 score O: 46 time X: 531,31 ms time O: 288,43 ms
ply X: 6   heuristic X: 3  ply O: 5   heuristic O: 3  score X: 32 score O: 27 time X: 391,44 ms time O: 108,41 ms
ply X: 6   heuristic X: 1  ply O: 6   heuristic O: 1  score X: 12 score O: 52 time X: 327,33 ms time O: 534,95 ms
ply X: 6   heuristic X: 1  ply O: 6   heuristic O: 2  score X: 17 score O: 0 time X: 62,70 ms time O: 79,99 ms
ply X: 6   heuristic X: 1  ply O: 6   heuristic O: 3  score X: 11 score O: 5 time X: 43,42 ms time O: 25,05 ms
ply X: 6   heuristic X: 2  ply O: 6   heuristic O: 1  score X: 25 score O: 22 time X: 882,36 ms time O: 594,58 ms
ply X: 6   heuristic X: 2  ply O: 6   heuristic O: 2  score X: 24 score O: 33 time X: 1784,55 ms time O: 1807,99 ms
ply X: 6   heuristic X: 2  ply O: 6   heuristic O: 3  score X: 45 score O: 19 time X: 852,89 ms time O: 768,67 ms
ply X: 6   heuristic X: 3  ply O: 6   heuristic O: 1  score X: 8 score O: 27 time X: 187,37 ms time O: 142,33 ms
ply X: 6   heuristic X: 3  ply O: 6   heuristic O: 2  score X: 15 score O: 49 time X: 551,51 ms time O: 738,21 ms
ply X: 6   heuristic X: 3  ply O: 6   heuristic O: 3  score X: 25 score O: 21 time X: 393,25 ms time O: 558,68 ms
```

8. Conclusion

This project effectively uses the Minimax algorithm with Alpha-Beta pruning to create an intelligent AI player for Othello. Three heuristic techniques—Disk Difference, Positional Weighting, and Mobility—are included to show how adaptable and strong the AI is to different game situations. Every heuristic has distinct benefits that strike a balance between strategic depth and computing efficiency. The project is a perfect foundation for testing AI techniques in competitive settings because of its design, which allows for numerous gameplay styles.

Performance study showed each heuristic's strengths at various game phases, with Mobility performing best in dynamic situations and Positional Weighting guaranteeing stability at the end of the game. The AI is able to compete effectively against both human and AI opponents by utilizing adaptive strategies and effective computing approaches.

Future developments can concentrate on strengthening the AI's decision-making abilities and integrating machine learning methods for adaptive heuristic optimization. The project might also become a useful teaching resource for fans of AI and game theory if it is expanded to incorporate more sophisticated visuals and interactive tutorials.

9. Resources and References

- Wikipedia, "Abstract Strategy Game," Link.
- Wikipedia, "Turn-Based Strategy," Link.
- Wikipedia, "Reversi," Link.
- Reversi (Othello) - Game Guide and Hints Link
- MSRavan's Implementation.
- Archilk's Repository.
- MmahdiM79's Othello AI.
- Pradhuman Goyal's Othello Game.
- LAustin's Othello AI.