Intro to AI: Othello Project

Parker Cummings, Lexi Franklin

Dr. Debasis Mitra

17 October 2024

*Note: Partner was unable to be contacted, and project was completed alone.

Algorithm/Pseudocode:

```
Initialize the game:
    Initialize board as an 8x8 grid
    Set up initial pieces for black (1) and white (2)
    Initialize current_player as 1 (Black starts)
    Initialize AI with player color and max depth

Define the main game loop:
    While game is running:
        If the game is over:
            Check for valid moves for both players
            If no valid moves for both, count the pieces for each player and declare winner
            Break the loop to end the game

        If current_player is human:
            Wait for player input (mouse click)
            Get row and column from player's mouse position
            If the move is valid:
                Place the piece on the board
                Flip opponent's pieces accordingly
                Switch current_player to the opponent (1 to 2 or 2 to 1)

        If current_player is AI:
            Use AI to find the best move with minimax and alpha-beta pruning
            Get row and column from AI's best move
            If the move is valid:
                Place the piece on the board
                Flip opponent's pieces accordingly
                Switch current_player to the opponent (1 to 2 or 2 to 1)

Helper functions:
    is_valid_move(board, row, col, player):
        For each direction:
            while in range of board:
                if current space is opponent
                    found_opponent = True
                else if current space is player
                    valid = True
                    end loop

                increment row and column in direction
```

```
place_piece(board, row, col, player):
    check if move is valid
    set current space to current player color

    For each direction:
        while in range of board:
            if current space belongs to opponent
                add to list of tiles to Flip
            if current space belongs to player
                flip all tiles in list


simulate_place_piece(board, row, col, player):
    check if move is valid
    set move to empty

    For each direction:
        while in range of board:
            if current space belongs to opponent
                add to list of tiles to Flip
            if current space belongs to player
                return current board position

gen_move_tree(board):
    for each row and column in board:
        if is a valid move
            new_board = simulate placing a piece
            if not None
                append to move list
    return move list

eval_board(board):
    Evaluate the current board by counting pieces for AI and opponent
    Return the difference in piece count
```

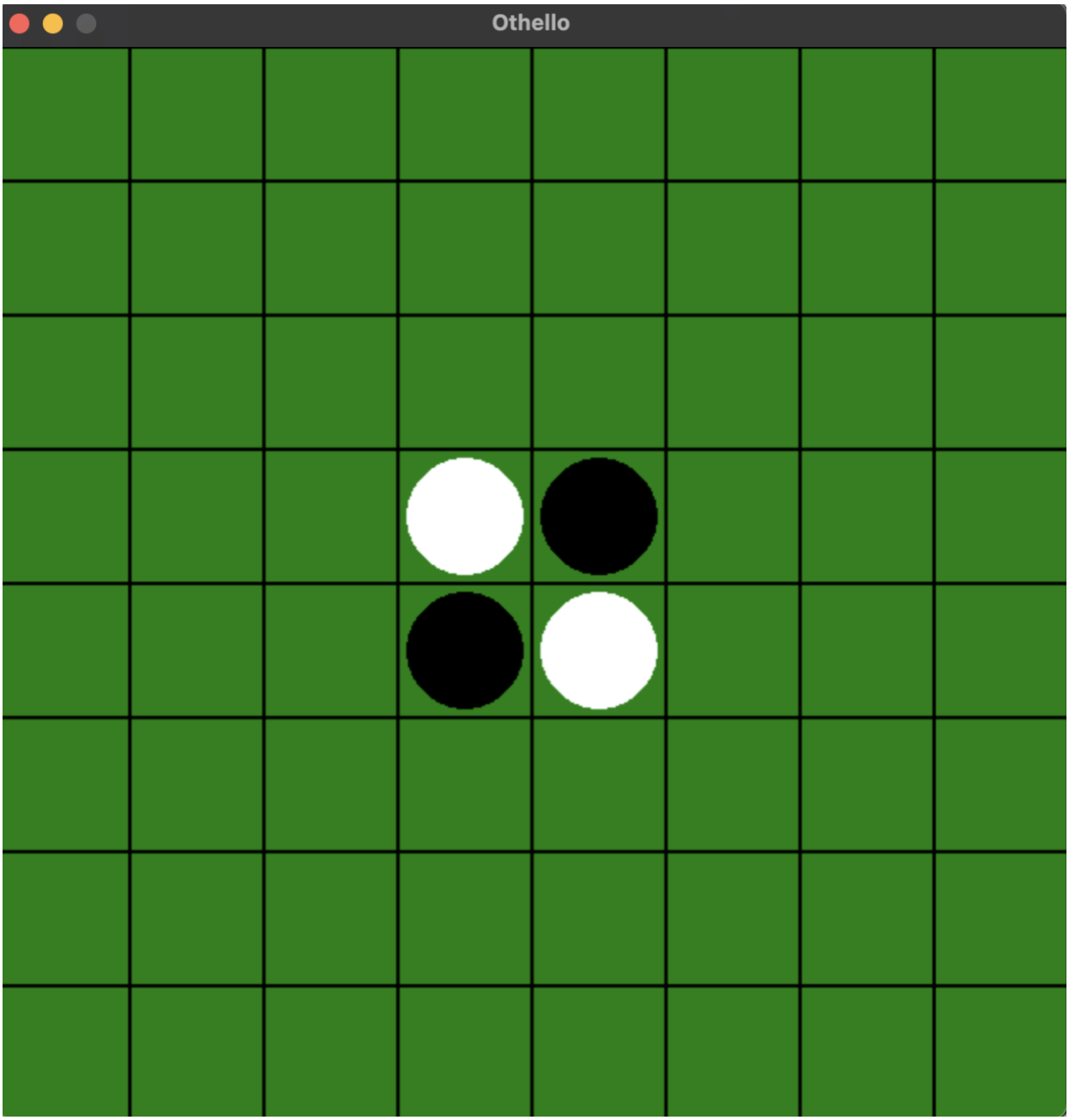
```
minimax_ab(node, depth, alpha, beta, max_player):
    if depth is 0 or terminal state:
        return eval_board(board)

    best_move = None
    if maximizingPlayer
        value = −∞
        for all children of current state
            child_value and best move = minimax(child, depth−1, alpha, beta)

            if child_value > value:
                set new value to beat
                best_move = move

            if value >= beta:
                end loop  // beta pruning

            alpha = max(alpha, value)

    else if minimizingPlayer
        value = ∞
        for all children of current state
            child_value and best move = minimax(child, depth−1, alpha, beta)

            if child_value < value:
                set new value to beat
                best_move = move

            if value <= alpha:
                end loop  // alpha pruning

            beta = min(beta, value)

    return value, best_move
```

```
has_valid_move(board, player):
    Check if the player has any valid moves left
    Return True if valid moves exist, False otherwise

check_game_over(board):
    Check if the game is over (no valid moves for both players)
    Return True if the game is over, False otherwise

count_board(board):
    Count the number of pieces for each player (black and white)
    Print the winner based on the piece count
```

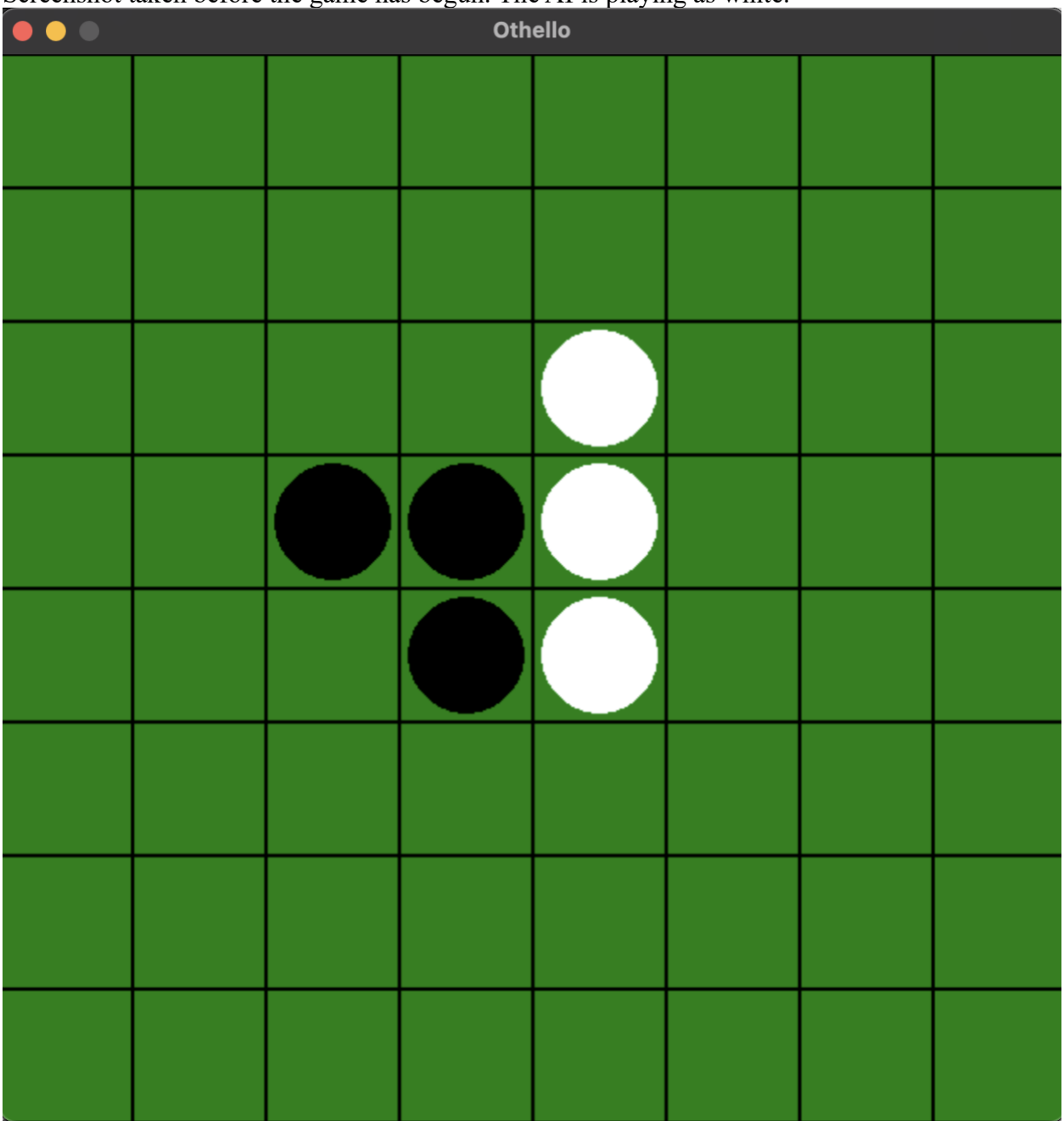
Sample Run Screenshots:

```
(othello_venv) ☁  othello  python3 othello.py
pygame 2.6.1 (SDL 2.28.4, Python 3.12.5)
Hello from the pygame community. https://www.pygame.org/contribute.html
What color should the AI play? (1 == Black or 2 == White): 2
Max Tree Depth? Input: 12
Current Player: black
```

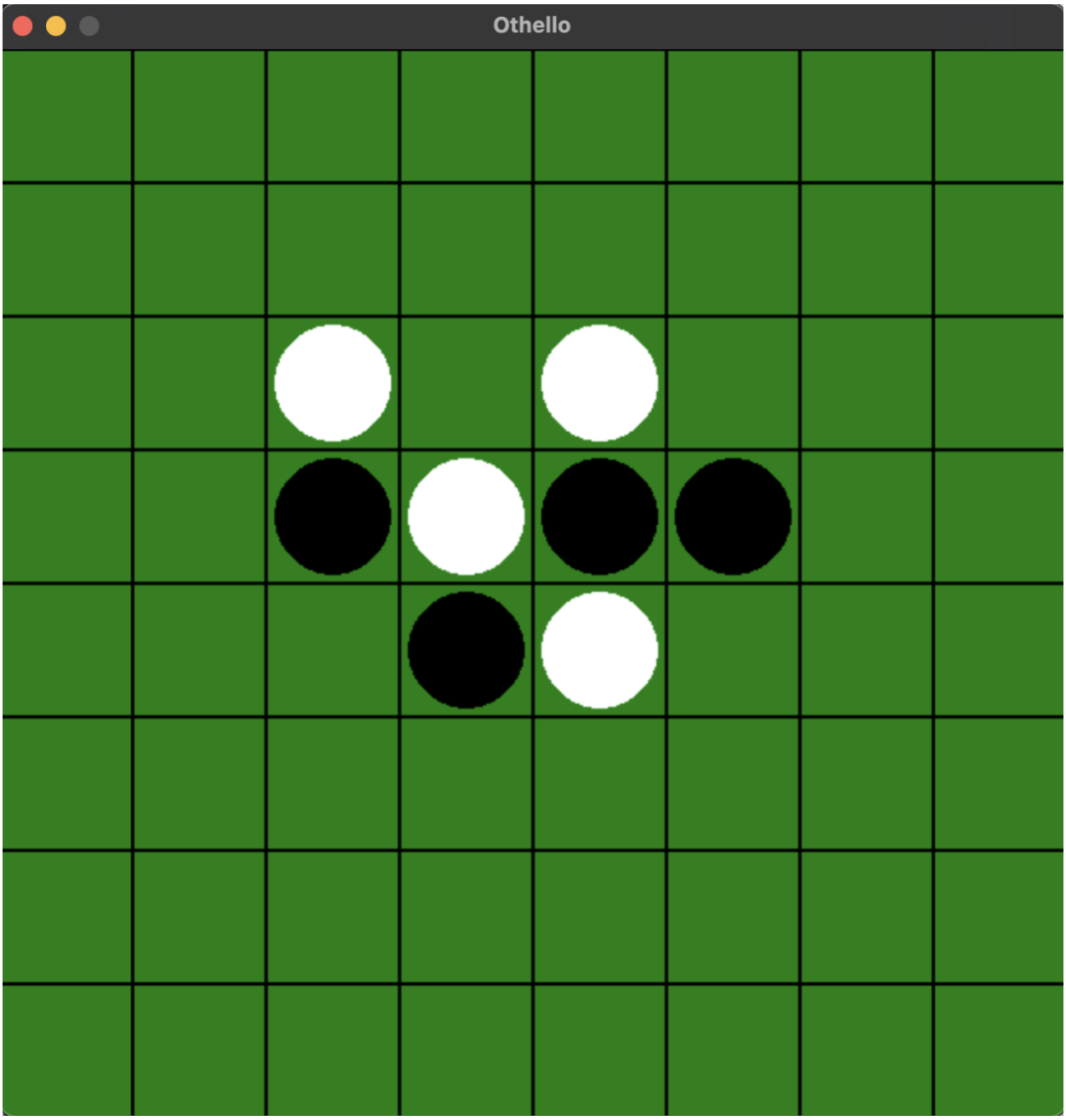
Screenshot of launching the game, implemented tree depth limit to reduce branching factor



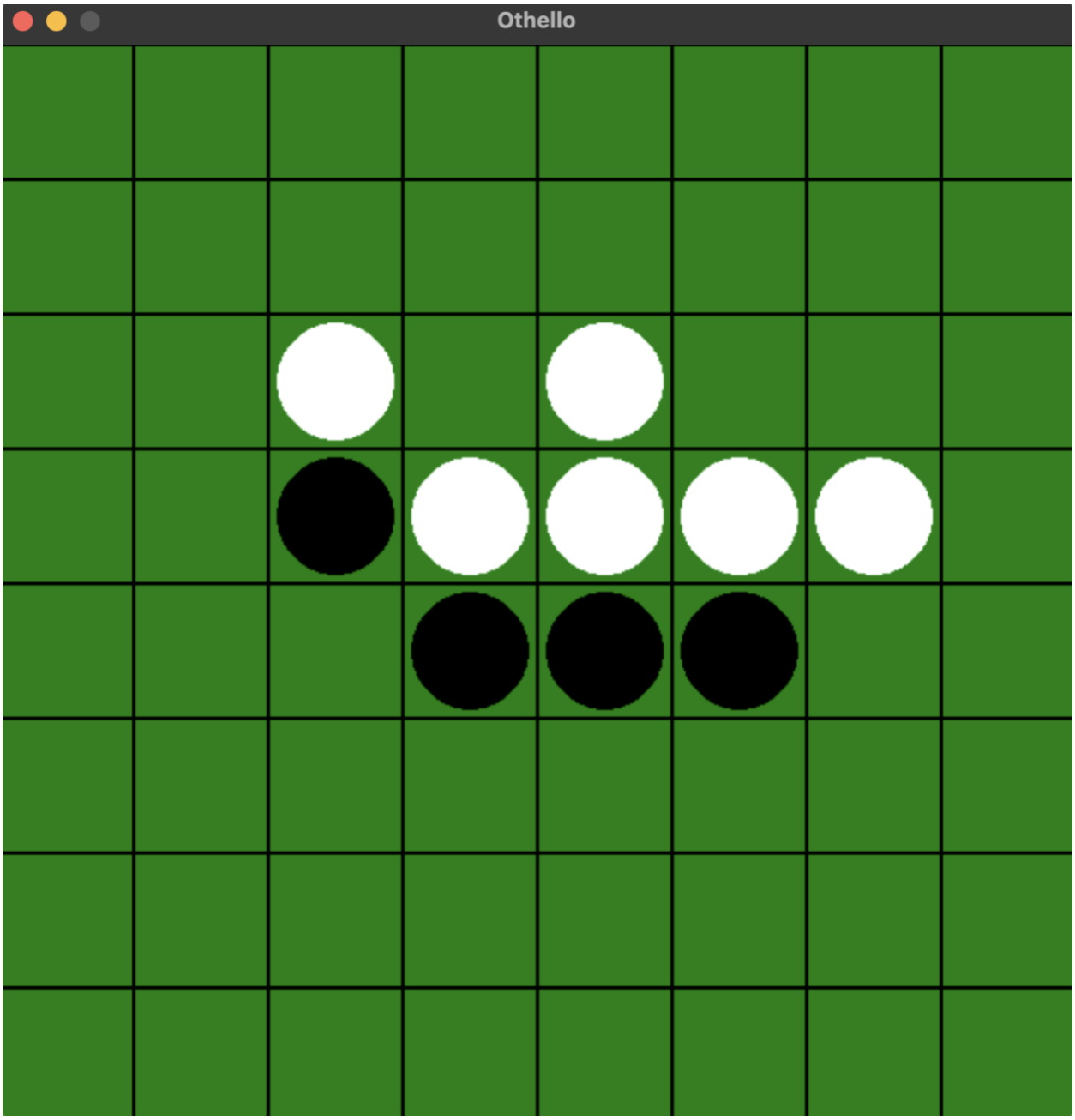
Screenshot taken before the game has begun. The AI is playing as white.



Screenshot taken after the first move. I chose (3, 2), and the AI chose (2, 4)



Screenshot taken after 2nd move. I chose (3, 5) and the AI chose (2, 2)



Screenshot taken after 3rd move. I chose (4, 5) and the AI chose (3, 6)

Free Form Discussion:

a) Board Evaluation Function

For the evaluation of the board at each step, I chose the heuristic known in the game as Disc Difference. This heuristic is the difference between the minimizing

player and maximizing player's discs on the current board. The function in pseudo code looks like such:

```
eval_board(board):
    ai_count = 0
    opp_count = 0
    For each row and col:
        if current space belongs to AI
            ai_count + 1
        else if current space belongs to opponent
            opp_count + 1

    return ai_count - opp_count
```

The function takes in a current board and returns the difference between discs on the board. The function is used in the minimax algorithm to determine the score of the best path found. The main reference I used for this algorithm can be found here: https://medium.com/@samharrison_58357/building-an-ai-to-play-my-favourite-board-game-othello-57f5aab1d6cf

b) Testing Plan

To test this algorithm, I will play various games with strategies learned online and determine where the AI is not making the best decisions. My current plan is as follows:

- Find online Othello strategy
- Attempt strategy against AI, draw decision trees
- Find where AI is lacking
- Debug code

c) Team Dynamics

Because my teammate was unable to be reached, there was not a focus of team dynamics for this project. However, since there are future assignments with the project, there will be room to improve Team Dynamics.