Assignment-3

Pratik sisodiya(234101040)
Nihal Shaikh(234101032)
(Group-11)

Problem Defination:

The goal of this project is to implement the game of Reversi (also known as Othello) using Python programming language. Reversi is a two-player strategy game played on an 8x8 grid board. The objective of the game is to have the majority of discs of your color (either black or white) on the board at the end of the game.

Rules of Reversi:

- **Board Setup:** The game board is initially set up with two black discs and two white discs placed diagonally in the center of the board. This creates an alternating pattern of colors.
- **Player Turns:** Players take turns placing discs of their color on the board. Black always moves first.
- **Legal Moves:** A player must place their disc on an empty square such that it creates a straight line (horizontal, vertical, or diagonal) between the newly placed disc and any of the player's existing discs. This line must enclose one or more of the opponent's discs, forming a straight line of the opponent's discs with the newly placed disc at one end and another disc of the player's color at the other end.
- **Flipping Discs:** When a player places a disc on the board, any opponent's discs that are enclosed between the newly placed disc and another disc of the player's color are flipped to the player's color.
- **Turn Skipping:** If a player has no legal moves, they must pass their turn to their opponent. If neither player can make a legal move, the game ends.
- **Game End:** The game ends when the board is filled with discs or when neither player can make a legal move. The player with the most discs of their color on the board at the end of the game wins.
- **Winning:** The player with the most discs of their color on the board at the end of the game wins. If both players have the same number of discs, the game is a tie.

Input Description:

The input to the Reversi game consists of the following components:

- **Player Moves:** Players input their moves by specifying the row and column where they want to place their disc on the game board. This input is provided interactively during the game, either by a human player or by the AI making decisions.
- **Game Board State:** The game board is represented as an 8x8 grid, where each cell can contain one of the following values:
  0: Empty cell
  1: Black disc **(X)** (belonging to Player 1)
  2: White disc **(O)**(belonging to Player 2)

- **Initial State:** The game board is initialized with two black discs and two white discs placed in the center of the grid, as per the standard starting configuration of Reversi.
- **Current State Updates:** As the game progresses, the state of the game board is updated after each move made by the players. Discs are placed on empty cells, and opponent's discs are flipped if they are flanked by the player's discs.
- **AI Decision Making:** When the AI player is making decisions, it analyzes the current state of the game board to determine the best move using the Minimax algorithm with Alpha-Beta Pruning. This decision-making process involves recursively evaluating potential moves and selecting the one that maximizes the AI's chances of winning. Here we are providing depth for which minimax algorithm will work if we increase depth it will increase difficulty level and time to generate next best move.

Implementation Details:

ReversiBoard Class:
- Represents game board, initializes, places discs, flips opponent's discs.
- Methods:
  - **__init__:** Initializes the board with starting positions.
  - **get_board:** Returns the current board state.
  - **is_empty:** Checks if a cell is empty.
  - **place_disc:** Places a disc on the board and flips opponent's discs if flanking occurs.
  - **flip_discs:** Flips opponent's discs in all eight directions if flanking occurs after placing a disc.

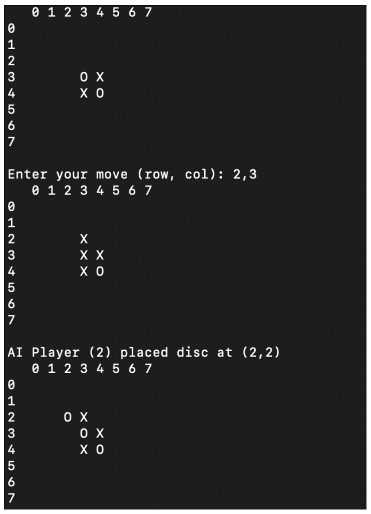
is_valid_move Function: Checks if a move is valid by analyzing flanking.
evaluate_game_state Function: Evaluates game state by counting player discs.
is_game_over Function: Determines if the game is over.
Minimax Algorithm: Finds best move for AI player with Alpha-Beta Pruning.
- **Parameters:**
  - **depth:** The depth of the search tree.
  - **player:** The current player making the move, if we increase depth it will increase difficulty level and time to generate next best move.
  - **maximizing_player:** A boolean indicating whether the current player is maximizing or minimizing.

    - **alpha and beta:** Alpha-Beta pruning parameters to optimize the search.

find_best_move Function: Utilizes Minimax to find optimal AI move.
Print Board Function: Displays current game state.
Main Function: Controls game flow, alternates player turns, and determines winner.

Output/Results:

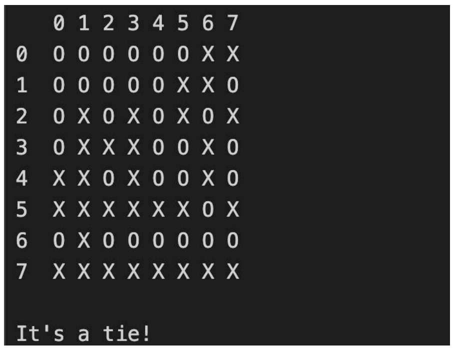
This is the output of our code with few states of game:



```
   0 1 2 3 4 5 6 7
0
1
2
3        O X
4        X O
5
6
7

Enter your move (row, col): 2,3
   0 1 2 3 4 5 6 7
0
1
2        X
3        X X
4        X O
5
6
7

AI Player (2) placed disc at (2,2)
   0 1 2 3 4 5 6 7
0
1
2      O X
3        O X
4        X O
5
6
7
```



```
   0 1 2 3 4 5 6 7
0  O O O O O O X X
1  O O O O O X X O
2  O X O X O X O X
3  O X X X O O X O
4  X X O X O O X O
5  X X X X X X O X
6  O X O O O O O O
7  X X X X X X X X


It's a tie!
```

Here you can see the after 2,3 move the 0 has flipped and then after AI move X has flipped.

References:

- Reversi/Othello Rules: https://en.wikipedia.org/wiki/Reversi
- Minimax Algorithm: https://en.wikipedia.org/wiki/Minimax
- Alpha-Beta Pruning: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning