HW 3

ARNAB DEY
Student ID: 5563169
Email: dey00011@umn.edu

Random Player

The *RandomPlayer* class is implemented in *othelloplayer.py* file. The instance of *RandomPlayer* randomly chooses one of the possible legal moves from a given state. If there is any possible move, it surely chooses one of them otherwise skips it's turn.

Utility Function

The utility function of Othello game can be determined by evaluating the following heuristics:

1. Coin Parity
2. Stability
3. Positional Strategy
4. Mobility
5. Corners Capture
6. Terminal State

We give a brief description of the above heuristics in subsequent sections.

Coin Parity

Coin parity computes the difference in coins between the player and its opponent. The heuristic value can be calculated as follows:

$$\text{Coin Parity} = \frac{K_1(\text{no. of player coins on the board} - \text{no. of opponent coins on the board})}{\text{no. of player coins on the board} + \text{no. of opponent coins on the board}},$$

where K_1 is a positive constant.

Coin parity enables the player to take those moves which result into states having more number of player coins on the board.

Stability

Coin parity does not capture the scenario where in the immediate next step taken by the opponent can alter the number of player coins drastically. Stable coins are the coins which cannot be flipped at any point in time in the game from the given state. Unstable coins are the coins which can be flipped at the very next move by the opponent. We also consider semi-stable coins which could potentially be flipped at some point in future but not in the immediate next move by the opponent. Stable coins have a positive utility, unstable coins have a negative utility and the semi-stable coins have zero utility. Stability can be calculated as follows:

$$\text{Stability} = \frac{K_2(\text{player stability} - \text{opponent stability})}{\text{player stability} + \text{opponent stability}},$$

where K_2 is a positive constant.

Positional Strategy

Here we assign static weights to each positions on the board. Each position on the board has a notion of stability given the position of other player coins on the board. Player coin at a certain position adds the corresponding static value to the positional utility, otherwise if the opponent has a coin at that position, static weight of that position is subtracted from the player positional utility.

Mobility

The goal of mobility is to restrict opponent moves while mobilizing player moves. It can be calculated as follows:

$$\text{Mobility} = \frac{K_3(\text{no. of possible player moves} - \text{no. of possible opponent moves})}{\text{no. of possible player moves} + \text{no. of possible opponent moves}},$$

where K_3 is a positive constant.

Mobility can consume additional processor time. Therefore, by default I have disabled mobility in my code.

Corners Capture

Coins at the corner cannot be flipped. Therefore, a high positive utility is give to a player coin that is at any corner. If opponent coin is there at any corner, a high negative value is added to the player utility thus discouraging moving to such states.

Terminal State

If a terminal state is encountered where the player wins, a high positive utility is given to the player. If the player looses in that terminal state, a high negative value is given to the player.

For my code, I have used coin parity, Positional strategy, corners captured and terminal state. Mobility is also implemented, however, by default I have disabled it.

Minimax Agent

The *MinimaxPlayer* class is implemented in *othelloplayer.py* file.

Alpha-Beta Agent

The *Alphabeta* class is implemented in *othelloplayer.py* file.