Assignment 2

Topic: Adversarial Search

The task of this assignment is to implement a game player which uses adversarial search algorithms to play a two-player game. The game that will be used is Mancala (*https://www.mathplayground.com/mancala.*html provides an online implementation. You are strongly encouraged to have a look at this site to know the rules)

The basic implementation includes minimax algorithm with alpha-beta pruning. You can experiment with different search strategies. For example, you can experiment with iterative deepening search. You can also carry out experiments by varying depth-limits and changing move-ordering.

You need to implement various heuristics and find out which one is better by running experiments. For example, you can determine the win-loss ratio by running 100 games with computer vs computer auto-play.

A heuristic function estimates how good a particular state is for a player. A number of factors determine whether a given state of the game is good for a player.

How to Play:

In Mancala, the board consists of six (6) bins on each side, and a home position (called storage) on the right of the bins. The board is laid out typically starting with four stones in each bin and both storage bins empty.

The following link contains a description of rules of Mancala.

https://www.thesprucecrafts.com/how-to-play-mancala-409424

Mancala heuristics:

For Mancala, the following strategic factors can be used to design a good heuristic to determine how favorable a particular position is for a player:

1) First valid move [furthest valid bin (a bin on my side which is not empty) from my storage]
2) How far ahead of my opponent I am now [the difference in stone count between my storage and opponent's storage]
3) How close I am to winning (If my storage is already close to containing half of total number of stones)
4) How close opponent is to winning
5) The total number of stones residing in the six bins of my side
6) The total number of stones residing in the six bins of opponent's side
7) Number of stones close to my storage (a stone, although residing in a bin on my side, is

not
8) close to my storage, if it is going to be overflowed to my opponent's side)
9) Number of stones close to my opponent's storage
10) Have I earned an extra move
11) Have I captured any stone
You can experiment with the following four heuristics (and some of your own heuristics):
heuristic-1: The evaluation function is
(stones_in_my_storage – stones_in_opponents_storage)
heuristic-2: The evaluation function is
W1 * (stones_in_my_storage – stones_in_opponents_storage) + W2 * (stones_on_my_side – stones_on_opponents_side)
heuristic-3: The evaluation function is
W1 * (stones_in_my_storage – stones_in_opponents_storage) + W2 * (stones_on_my_side – stones_on_opponents_side) + W3 * (additional_move_earned)
heuristic-4: The evaluation function is
W1 * (stones_in_my_storage – stones_in_opponents_storage) + W2 * (stones_on_my_side – stones_on_opponents_side) + W3 * (additional_move_earned) + W4 * (stones_captured)