Artificial Intelligence Assignment-6 Report

GROUP NO. – 26: RAJAN GUPTA (210020041), ARYAN GUPTA (210010005)

DESCRIPTION OF OTHELLO GAME IN AN OPTIMAL WAY

To win the game of Othello, given a board configuration and a turn, your bot will return a valid move. The game ends when neither of the players can make a valid move. The player with the maximum number of coins is the winner.

MINIMAX ALGORITHM

Minimax algorithm is a backtracking algorithm used in game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.

1. The algorithm creates a tree of game positions by exploring all possible moves by the opponent.
2. The algorithm continues exploring until it reaches a specific depth in the tree.
3. At each depth, the algorithm evaluates the position using a heuristic or evaluation function.
4. Evaluates the best move such that it minimizes the best move of the opponent in the game.

ALPHA BETA PRUNING ALGORITHM

Alpha-Beta pruning is an optimization technique used in minimax algorithm. The idea behind this algorithm is cut off the branches of game tree which need not to be evaluated as better move exists already.

1. This algorithm minimizes the number of positions explored by the Minimax algorithm in a game tree.
2. It achieves this by stopping the evaluation of a move as soon as it finds at least one possibility that is worse than a previously evaluated move.
3. Once such a move is found, there is no need to evaluate further moves on that path, and the algorithm prunes that branch of the search tree.
4. Pruning the search tree in this way does not change the outcome of the algorithm, but it significantly reduces the number of positions that need to be evaluated.

HEURISTIC FUNCTION

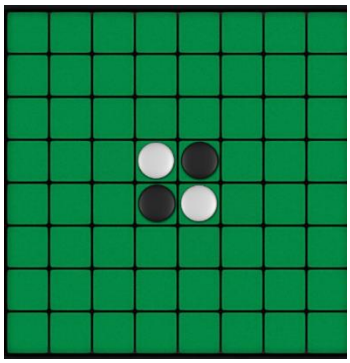
The algorithm assigns values to each square on the board based on how desirable it is to have a piece there, and then sums up those values to get a score for the current state of the board.

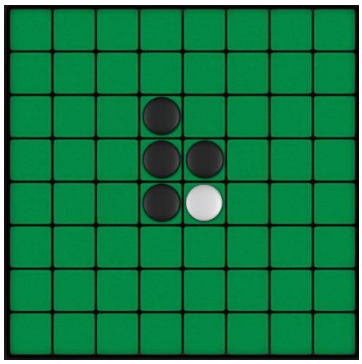
1. A 2D array *optimal_board* is defined with pre-computed values for each square on the board. These values represent the desirability of having a piece in that square.
2. A variable *h_value* is initialized to 0.
3. The algorithm loops over each square on the board and checks its current state :
    - If the square is occupied by the current player's piece (as returned by *board.get(i,j) == turn*), its corresponding value from *optimal_board* is added to *h_value*.
    - If the square is occupied by the opponent's piece (as returned by *board.get(i,j)==other(turn)),* the corresponding value from *optimal_board* is subtracted from *h_value.*
4. After looping over all squares on the board, the final value of *h_value* is returned.

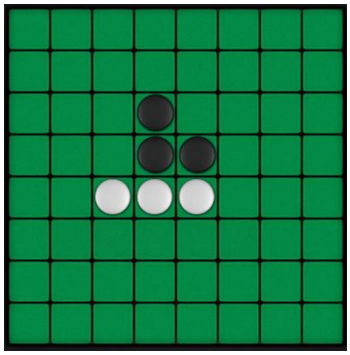
TREES MOVES FOR A GIVEN BOARD CONFIGURATION

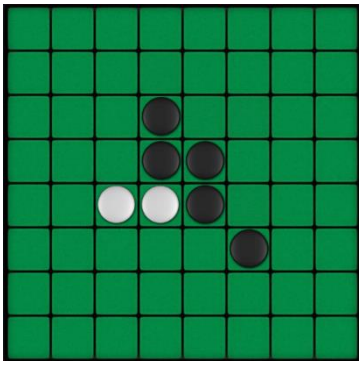
Moves for minimax algorithm against random bot :

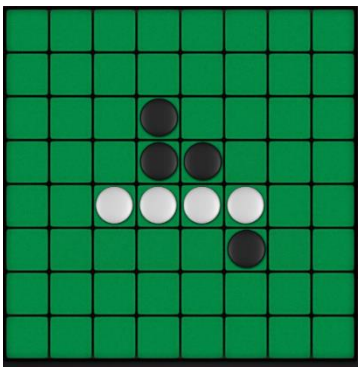
```
bajrangbali@bajrangbali:~/Desdemona$ ./bin/Desdemona ./bots/RandomBot/RandomBot.so ./bots/minimax_bot/minimax_bot.so
Loading libOthello...
Loading bot...
Loading bot...
Game Over
[Win]: Red
-22
bajrangbali@bajrangbali:~/Desdemona$ cat game.log
d5
c3
f2
f3
c2
c1
g2
c5
b5
f5
b3
e2
g6
e5
f6
c4
b2
c6
b1
```

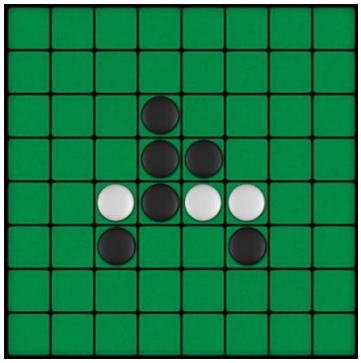


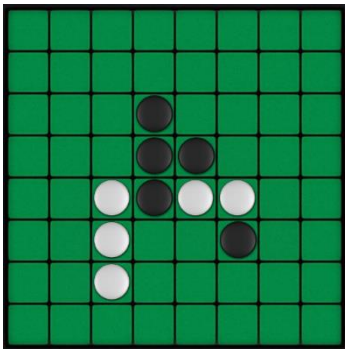






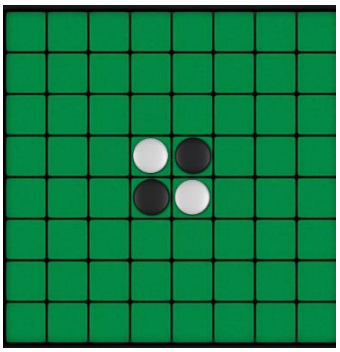


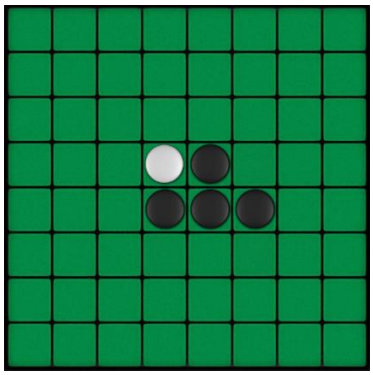


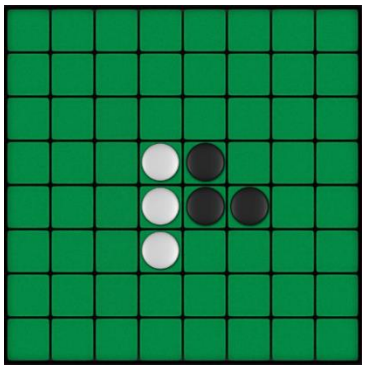


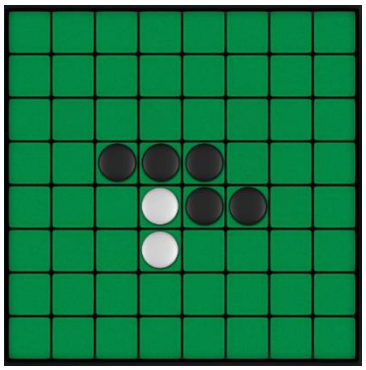
Moves for alpha beta pruning against random bot:

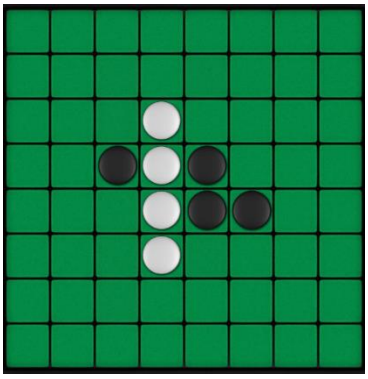
```
bajrangbali@bajrangbali:~/Desdemona$ ./bin/Desdemona ./bots/RandomBot/RandomBot.so ./bots/AB_bot/AB_bot.so
Loading libOthello...
Loading bot...
Loading bot...
Game Over
[Win]: Red
-46
bajrangbali@bajrangbali:~/Desdemona$ cat game.log
f3
d2
c4
d5
c2
b2
b1
f4
e6
b4
b3
f2
a4
a2
d1
e2
f1
```

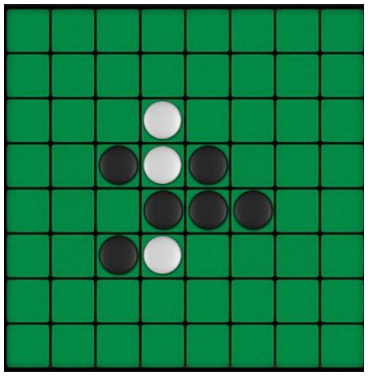


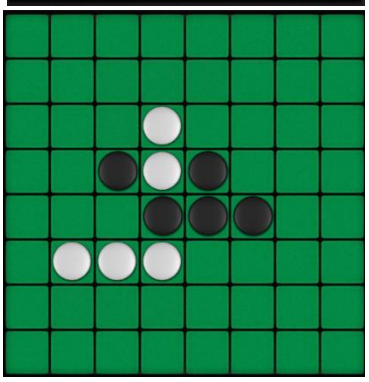












COMPARISON BETWEEN MINIMAX AND ALPHA-BETA PRUNING

---

1. **Space and Time Complexity:**
   The Alpha-Beta Pruning Algorithm has less time and space complexity since it does not explore moves that are guaranteed to be worse than previously examined moves. Hence, the space complexity is reduced by eliminating worse states that do not need to be explored. Because the Alpha-Beta Pruning Algorithm explores fewer states than Minimax, the time complexity is also less.

2. **Wining Criteria:**
   The Alpha Beta bot is able to search deeper into the game tree than the Minimax bot due to the limited time constraint of 2 seconds per move. When the time constraint is relaxed, both bots are expected to play equally well. Multiple trials indicate that the two bots play nearly equally well and that the winning bot is usually the one who starts the game first. This trend is due to the heuristics used by the bots, which have an impact on their chances of winning the game.