# CS 312: Artificial Intelligence Laboratory
# Task 5: Game Playing


Group **14**

Team Members

- 180010029 [ Rupesh Kalantre ]
- 180010031 [ Shagun Bera ]


## 1. Brief Description of Algorithms used

- **Minimax Algorithm** : Minimax is a decision rule used in artificial intelligence, decision theory, game theory and statistics for minimizing the possible loss for a worst case (maximum loss) scenario. This algorithm makes a tree of positions as it explores all possible moves by the opponent. When it reaches the required depth, it evaluates the position using a heuristic function. It evaluates the best move such that it minimizes the best move of the opponent.

- **Alpha Beta Pruning :** Alpha-Beta Pruning minimizes the number of states explored in the search tree made by the Minimax algorithm and prunes the search tree. It stops evaluating a move when it has been found that the current move is worse than a previously examined move. Such moves need not be explored further.

## 2. Heuristic Functions considered

- **Positions Heuristic**

We have observed that occupying the corners of the board has a strategic advantage and the more number of coins you have the better is your situation .So we have constructed a heuristic that takes both factors into account

```
int H(OthelloBoard board) {
    int corners = 0;
    corners += (board.get(0, 0) == TURN) * 5;
    corners += (board.get(7, 0) == TURN) * 5;
    corners += (board.get(0, 7) == TURN) * 5;
    corners += (board.get(7, 7) == TURN) * 5;
    return board.getBlackCount() - board.getRedCount() + corners;
}
```

- **Possible Moves Heuristic**

The player who has more number of possible moves in his hand has an advantage. This heuristic tracks that.

```
int H1(OthelloBoard board) {
    return (int)board.getValidMoves(TURN).size() - (int)board.getValidMoves(flip(BLACK)).size();
}
```

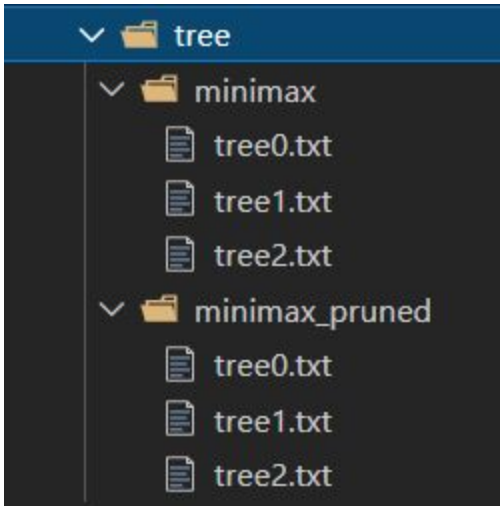
- **Coins Heuristic**

This heuristic returns the difference between the coins.

```
int H2(OthelloBoard board) {
    return board.getBlackCount() - board.getRedCount();
}
```

**3.Trees to show my particular moves are chosen for any 6 moves given the board configuration**

We have a separate folder “tree” which contains 3 trees ( using 3 different heuristics) for each algorithm




**Minimax Algorithm**

- [Positions Heuristic] -> tree/minimax/tree0.txt
- [Possible Moves Heuristic] -> tree/minimax/tree1.txt
- [Coins Heuristic] -> tree/minimax/tree2.txt

**Alpha Beta Pruning**

- [Positions Heuristic] -> tree/minimax_pruned/tree0.txt
- [Possible Moves Heuristic] -> tree/minimax_pruned/tree1.txt
- [Coins Heuristic] -> tree/minimax_pruned/tree2.txt

## 4. Comparison of both Algorithms

- **Space and Time Complexity** : The Alpha-Beta pruning algorithm has lesser space and time complexity as moves that are guaranteed to be worse than previously examined moves, are not further explored. By eliminating worse states that need not be explored, the space complexity is reduced. Since, lesser states are explored, in comparison to Minimax, the time complexity is also lesser. As we are using dfs for implementing minimax algo, time complexity is roughly $O(b^d)$ and space complexity is $O(d * b)$.

- **Winning Criteria** : The 2s deadline to play the next move gives the Alpha-Beta pruning bot the advantage of exploring greater depths compared to the Minimax Bot. When unbounded by time constraints, both the bots are expected to play equally well. The two bots are compared using the same heuristic.
  When the two algorithms play against each other , we observe that the bot with alpha beta pruning wins game as it is able to search till a greater depth.

```
→  Desdemona git:(main) bin/Desdemona bots/MyBotabp/bot.so bots/MyBot/bot.so
Loading libOthello...
Loading bot...
Loading bot...
Game Over
[Win]: Black
48
→  Desdemona git:(main) ✗ bin/Desdemona bots/MyBot/bot.so bots/MyBotabp/bot.so
Loading libOthello...
Loading bot...
Loading bot...
Game Over
[Win]: Red
-8
```