CS 214: Artificial Intelligence Lab
Spring 2022-23, IIT Dharwad
Assignment-6
Game Playing
Artificial Intelligence Lab

Problem Statement

Code a Bot to play the game of Othello in an optimal way, in order to win the game.

Description: In this assignment, you will code a bot 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.

Programming Language: C++

System specifications: 64-bit Linux distribution Instructions

Setting up the framework: We will be providing you with a framework (Desdemona.zip) that lets two bots compete against each other.

1. Extract the contents of Desdemona.zip into a suitable directory.

2. Set up the framework by issuing a make command in the root of this directory.

Programming the bot

- You will modify “MyBot.cpp” to return a valid move whenever the function “play" is called. The file is located in “bots/MyBot”.
- The makefile is also provided at this location. Use it to generate a “.so” file.
- All other source files are to be left untouched.
- You can test your bot against another bot by issuing the command “./bin/Desdemona ./.”
- By convention, the first bot is BLACK and the second RED.
- A random bot (bots/Random Bot) has been provided for testing.
- At the end of the game, a “game.log” file is created that contains the sequence of moves made.
- There should be NO print statements in the code submitted.
- If a bot returns an invalid move, it will be disqualified.

Helper functions :

The following functions have already been written to assist you:

- bool OthelloBoard::validateMove(Turn turn, int x, int y) : True if the move (x,y) is valid for the turn, False otherwise
- bool OthelloBoard::validateMove(Turn turn, Move move): True if the move is valid for the turn, False otherwise
- void OthelloBoard::makeMove(Turn turn, int x, int y): Updates the board configuration by making the move (x,y); throws an exception if the move is not valid
- void OthelloBoard::makeMove(Turn turn, Move move): Updates the board configuration by making the specified move; throws an exception if the move is not valid
- list<Move>OthelloBoard::getValidMoves(Turn turn): Returns a list of valid moves that can be made given the turn
- int OthelloBoard::getBlackCount(): Returns the number of black coins on the board
- int OthelloBoard::getRedCount(): Returns the number of red coins on the board
- void OthelloBoard::print(Turn turn): Prints the turn, the board configuration, and the number of black and red coins. 'X' is BLACK, 'O' is RED, and unfilled locations are blank

Time Constraints: Each bot can take at most 2 seconds to return a move. If this time limit is exceeded, the bot causing the timeout will be disqualified.

Submission:

Apply a minimax algorithm to code a bot (minmax_bot.so) and alpha-beta pruning to code the other (AB_bot.so). Test the two bots against each other and record the observations.

Evaluation Criteria:

Heuristic Functions: 5
Correctness: 20
Report: 20
Demo: 5

Report Format :

1. Brief description of the algorithms
2. Heuristic functions considered
3. Trees to show my particular moves are chosen for any 6 moves given the board configuration. (3 for each algorithm) .
4. Compare the two algorithms and justify which is better in terms of
    1. Space and Time complexity
    2. Winning criteria

NOTE:

- Download *Desdemona.zip*, unzip and rename it with your *group number*.
- Write your code in respective files.
- Test your code and submit it on moodle.
- Due date for the Assignment is *18th March 2023 (11:59PM)*
- Penalty for late submission is *10%* of secured marks.
- We will run a plagiarism check for all the submissions, if found copied *100%* penalty will be applied.
- Viva and demonstration of your submitted code is mandatory and we will share the time slots and date for the same.