CS562-Artificial Intelligence

Assignment 2

October 9, 2018

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 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 ./<path to bot1.so >./<path to bot2.so >"
- 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 atmost 2 seconds to return a move. If this time limit is exceeded, the bot causing the timeout will be disqualified.

Tournament Details

Bots submitted by all groups will be contestants. In the trial round, the working of the bots would be checked and in the final round, each bot will play against every other bot twice, once as the first player (Black), and then as the second (Red). At the end of the tournament, each bot will be given a rank based on the total points scored.

Point System

Win: 64 + (winner's coins loser's coins)
Loss: 0
Disqualification: -64

Submissions

Trial Round

Upload a single ".so" file with the name "groupno.so", where groupno is the group number assigned to you in the previous assignments.

Final Round

Upload a zip file containing your source code as well as the "groupno.so" file. The zip file is to be named groupno.zip (e.g 23.zip)

Deadlines

Trial Submission

30th October @23:55

Final Submission

07th November @23:55