Stanley Zheng

ECE469: Artificial Intelligence

Project 1: Checkers AI

Compilation & Execution:

All code related to the project is within one file, CheckersAI.cpp.

Compile and run the executable. The program will prompt the user to choose between the three game modes of Human vs. Human, Human vs. Computer, and Computer vs. Computer. The program will then prompt the user to enter a filename to load an existing board if they want to load one. The format of the input file should be identical to the ones provided to us; they were labeled as "Sample Checkers Board" on Professor Sable's online course description. The program will also ask for the starting player and time limit before starting the game.

Once the game starts, the program displays the current turn number and the current turn player. Then, it will print out the legal moves and the board. Human players will be prompted for a move in the form of two sets of characters referring to coordinates on the board (ex. '6e 5f'). Computer players will perform a combination of iterative deepening and an alpha-beta search evaluation function to find and make a 'best' move. It will then display the maximum depth that was searched to and the search time.

The game ends once there are no moves left to make. The program will display the winner and then ask the user if they want to play again.

Notes:

- There are times when the program will display a search time that is slightly above the time limit that is set (over by about 0.002 seconds).
- The program will call a draw when there is only one piece on each side.
-

"After every move by either the computer or the user, the program should display the updated board."

- I decided to not display the updated board immediately after someone's turn because there would an updated board for the next player as soon as their turn comes up. I thought this was a bit redundant and that it would look a little nicer if each state was only printed once, in terms of scrolling back to see all the moves.

Implementation:

The program was implemented using a bit board to represent the position of the pieces on the board. The bit boards were in the form of three unsigned integers storing the white pieces, the black pieces and the kings. Bitwise operators allow the program to do some neat things, such as getting the unoccupied squares (not(whites | blacks)), getting the positions of the white kings (whites & kings), and getting the available moves.

The AI portion of the program consists of a move function that implements iterative deepening and minimax with alpha-beta pruning. A SIGALRM signal is used to interrupt the iterative deepening after the specified time has been reached.

The heuristics function takes into account:

- number of kings
- number of pieces that can jump
- position of the pieces in terms of how far they are from being promoted
- discourages kings positioned along edges
- when both sides have less than 6 pieces, the side with more pieces will be more aggressive while the other will be more defensive in their playstyles.