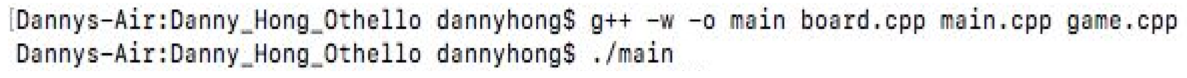
ECE-469 Artificial Intelligence

Project 1: Othello Writeup

Danny Hong

Othello (aka Reversi) is a strategy board game invented in 1883 that is played on a 8x8 checkerboard designed for 2 players. For this project, the board game was programmed in C++, printing out the board to a shell. Compiling the program involves just two lines, as shown below:



```
[Dannys-Air:Danny_Hong_Othello dannyhong$ g++ -w -o main board.cpp main.cpp game.cpp
 Dannys-Air:Danny_Hong_Othello dannyhong$ ./main
```

Figure 1: Compiling source codes (-w was added so that warnings would not be displayed)

The related source code files along with their header files (except main) are board.cpp/board.h, game.cpp/game.h, and main.cpp. Board.cpp/board.h are responsible for initializing the board through creating and implementing a class that is called Board. Additionally, Board contains a variety of functions including checkOnBoard: which checks if a given coordinate pair is within the confines of the board, traverse: which traverses through the surrounding area of a given space (square) to look for and determine whether certain spaces (squares) are free/occupied, legalMoves: which looks for the legal moves a player/CPU can make, changePlayer: which handles alternating turns between a user and the CPU, applyMove: which applies a specific legal move that is made along with handling the flipping of black or white pieces upside down (depending on the moves that are made), print: which prints the present state of the board along with the placement of the black/white pieces, and checkGameEnd: which checks if the game has reached the end.

Game.cpp/game.h is responsible for handling the actual gameplay along with designing the AI for the CPU to verse the user. Minimax Alpha-Beta Pruning based on recursive deep search iterations was implemented in the alphaBeta function and move costs were calculated from the heuristic function. Other important functions include set: which sets the board up for playing and even allows for the user to insert an old saved game file for play in which the game will start from the present state of the board from that file, play: which allows the game to continue on while the game does not meet the end conditions, computerMove, which displays legal moves that the CPU can make and handles the move that will be made by the CPU, and userMove: which displays legal Moves that a use can make and handles moves that will be made by the user. Finally, main.cpp executes game.cpp as it calls on game.cpp's set and play functions and handles user input from the command line as well.