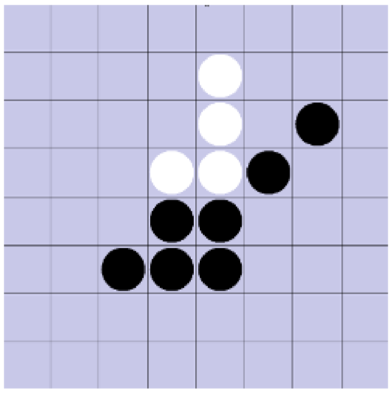
CP365 - Othello Championships

Assignment Overview



For this assignment, we're going to have an Othello bot tournament. You must code up your bot to play Othello as competitively as possible and then it will be pitted against the weak, inferior bots designed by your classmates. The winning team will earn the right to **drop a low assignment score**.

Othello Framework

Much like we did with the sliding numbers puzzle assignments, we will be using a simple framework to build and test our bots. There are several Java files you should download and use to base your agent off of:

- Othello.java - The base game that defines the board and manages the rules of the game. You should not need to modify this file. It will likely be helpful to look through this code to see how the game works, though, and which methods are available in the OthelloBoard class.
- RandomBot.java - An example bot that makes random moves. If your AI bot cannot beat this bot, then you are in trouble. The code for this bot also has some helpful comments that describe some of the useful methods you can call from the OthelloBoard class.
- GreedyBot.java - An example bot that always makes the move that captures the most opponent pieces. This bot beats RandomBot, but your AI bot should be able to defeat this bot without difficulty!
- HumanPlayer.java - Using this player you can play the game yourself. Try playing against your own bot!

Your Bot

Recall from our discussion in class that searching in two-player games is a little bit different from the searches we did a couple of days ago. A bot that performs a **minimax search using alpha-beta pruning** is probably a great place to get started....

Are further optimizations possible after alpha-beta pruning? You betcha. Because of the size of the game state space, you might not be able to search as deep in the game state tree as you would like. You might try adding a few heuristics for focusing your bot's search a bit.

You might try looking online for some common human strategies when playing the game and see if any of those can be incorporated into your design.

Testing your bot:

Test your bot using the provided framework. You have the ability to run a large number of trial games quickly or you can watch a game to see how your bot's actions are working out.

Run the game like this:

```
USAGE:
    java OthelloGame BlackBotClassName WhiteBotClassName BOARD_SIZE NUMBER_GAMES PLAYBACK_DISPLAY PLAY

EXAMPLE (play 10 games on an 8x8 board with RandomBot and GreedyBot):
    java OthelloGame RandomBot GreedyBot 8 10 0 0

EXAMPLE (watch a game on a 4x4 board with a 1 second delay between moves):
    java OthelloGame RandomBot GreedyBot 4 1 1 1000

EXAMPLE (Have your bot play 1000 games against the GreedyBot):
    java OthelloGame MyBot GreedyBot 8 1000 0 0
```

Things to consider:

- To what depth can your bot search in a reasonable amount of time on an 8x8 board?

- Can your bot defeat you?
- How will you celebrate your victory over your classmates?