# HW2

Due: 03/31/2023 at 11:59pm EST

**Collaboration Policy:** As mentioned in the syllabus, you are encouraged to work in pairs. If you have a partner, each pair should submit **one** submission on gradescope. Therefore, please make sure that in your source code (preferably in a comment above the class name), you list the names and BU IDs of each person who worked on it!

## 1 Setup

After downloading hw2 from Piazza and unzipping the .zip file, you should notice that there are two directories in the starter code: `src` and `data`.

### Moving Files to the Right Place

There are quite a few files in `src`, but suffice to say that you will only be working on three, maybe four or five of them. Contained within this directory is all of the java code that I have written to force SEPIA to play chess. It may not be pretty, but that is because I have not yet figured out how to change the texture of squares.

Within this massive java package is one sub-package called `src/hw2/agents` Inside this sub-package are all of the files that you will need to actually touch. The three you have to care about the most are `src/hw2/agents/AlphaBetaAgent.java`, `src/hw2/agents/heuristics/CustomHeuristics.java`, and `src/hw2/agents/moveorder/CustomMoveOrderer.java`. These three files contain your alpha-beta agent, heuristics, and custom move ordering code respectively. More on this later. Please copy the entire `hw2` java package to whatever workspace you are using. If you are using Eclipse or IntelliJ, you should be able to drag and drop it into the "src" folder in your workspace.

Back to the top level of your extracted .zip file, if we go inside the `data` sub-directory, you will see just one sub-sub-directory also called `hw2`. Fear not, this is intended, and is only to help your `data` directory be less cluttered. Please copy the `data/hw2` directory over to the `data` folder you are using in your workspace.

### Modifying the `.xml` files

I am pretty sure that you will not need to modify any .xml files here.

### Creating `hw1` Run Configuration

We will need to create several run configurations for this hw. Please create one run configuration to execute `data/hw2/RandomvsRandom.xml` (by specifying this path in the "Arguments" section of the run configuration. I would call it something like "hw2_random_vs_random". When you run this configuration, you are pitting two random chessbots against each other. This is pretty close to how fast the game is going to run in the best case on your machine. These games are pretty quick.

Please add a new configuration for `data/hw2/MinimaxvsRandom.xml`. Here you will be pitting one minimax agent (with a max depth of 3) against a random agent. Please pay attention to how slow minimax is. This is pretty close to the worst case, and minimax will not be able to play timed chess (it may even lose the game you're running if you let it run long enough)!

Please add one configuration for each .xml file with "AlphaBeta" in the name. These configurations will be used to test your alpha-beta solution against opponents of varying difficulty.

## The Goal

Your goal is to write an agent that implements the alpha-beta pruning algorithm to find the optimal chess move to play. To do so, you only need to modify the three files mentioned previously:

1. `src/hw2/agents/AlphaBetaAgent.java`
2. `src/hw2/agents/heuristics/CustomHeuristics.java`
3. `src/hw2/agents/moveorder/CustomMoveOrderer.java`

All other functionality is implemented for you, and I will be providing developer documentation if you are curious about how I got this to work. It wasn't easy: I spent the majority of my time since the beginning of spring break bug hunting (my favorite pastime).

I recommend that you divide and conquer this. I have written the minimax agent to use your custom heuristic functionality to assign utility values to non-terminal nodes. Therefore, when developing your heuristics, I recommend using the minimax configurations so you don't have to worry about the algorithm. Likewise, whe you are developing your alpha-beta pruning algorithm, I recommend either using my default heuristics (if you start with the algorithm first) until you get the algorithm working, and then pivot to the heuristics. Once both of those are working, then I would focus on move ordering.

## 2 Alpha-Beta Pruning Agent (50 points)

Inside the alpha-beta pruning agent, there are two things I want you to implement (that are all marked with a "TODO"):

1. `AlphaBetaAgent.AlphaBetaSearcher.alphaBetaSearch` (45 points): This method is responsible for calculating the plan as a stack of `MapLocation` objects. We have taken care of all implementation that executes this plan. Please see the description of the method for more information.
2. `AlphaBetAgent.DEFAULTMAXDEPTH` (5 points): This hyperparameter is how far your agent will think into the future. The larger the value, the better your agent will be, but the longer it will take to compute. After you have written all of the code, please spend a little bit of time trying to tune this, this will matter in the tournament.

## 3 CustomHeuristics (25 points)

Inside the `CustomHeuristics` class, I want you to put your own heuristics functionality. These may involve board position, strategy, piece formation, etc.

## 4 CustomMoverOrderer (25 points)

Inside the `CustomMoverOrderer` class, I want you to put your own move ordering functionality. I recommend doing this **last**. My default version is pretty crappy, and there are way better solutions to be found. My only advice is that you may want to consider making it piece dependent, and maybe even context dependent.

## 5 Grading

We will be grading your submission based not only on the quality of your code, but on the quality of the performance that your agent has. In this assignment, quality of code includes how easy it is for us to read your code (please be descriptive and leave comments), the algorithm itself (for correctness), and the quality of your custom heuristics/move orderings. To earn full points, your agent must be able to play a full game of chess (untimed). We will be awarding bonus points if your agent can play a complete game in under 8 minutes. Reminder that any solution that compiles and is turned in on time will participate in the extra credit tournament!

As always, please feel free to ask us any questions you may have! Good luck!