CS 6601: Assignment 1

Due September 11, 2016 by 11:59PM UTC-12 (Anywhere on Earth)

You will write game playing agents for a game called Isolation.

1. The Game

The rules of Isolation are simple. In our version, two players take turns placing their own game piece on different squares of a 7-by-7 grid. After the first two moves (in which each player puts their piece on any unoccupied square), **the pieces move like knights in chess**: two squares vertically and a square horizontally, or two squares horizontally and a square vertically (an L-shape). Each time a player moves their piece, the square that they were previously occupying is blocked and cannot be moved to, but can be moved *through*, for the remainder of the game. In addition, the players may not occupy the same square but they may move through each other. The first player who is unable to move loses.

2. Your Assignment

Your task is to create an AI that can play and win a game of Isolation. Your AI will be tested against several pre-baked AIs as well as your peers' AI systems. You will implement your AI in Python (from a provided IPython notebook), using our provided code as a starting point.

The assignment is available at: https://github.gatech.edu/omscs6601/assignment_1. Read the README for instructions on how to utilize the repository. See the attached IPython notebook for a full reference of useful methods you might need.

In this repository, we provide:

- A class for representing the game state
- A function for printing the game board
- A function for generating legal game states
- A class for running unit tests
- A random AI (baseline test)

Your goal is to implement the following parts of the AI in the class CustomPlayer:

- Evaluation functions (OpenMoveEvalFn() and CustomEvalFn())

- The minimax algorithm (minimax())
- Alpha-beta pruning (alphabeta())

Your agent will have a limited amount of time to act each turn (500 ms).
Note that we only time each turn, but if your agent takes more than a few minutes at construction time, for example because you're loading the entire set of possible board states from memory, you will be penalized.

These are the bare minimum requirements for your AI, and the rest is up to you. You will be scored according to how well your AI performs against some baseline AIs that we provide (see "Grading"). If you want to improve over the base performance, here are a few suggestions:

- Storing the evaluation scores for past moves.
- Modifying your evaluation function to account for "killer moves".
- Ordering terminal nodes to maximize pruning.

3. Grading

The grade you receive for the assignment will be determined as follows:

10 points	:	You write an evaluation function that scores based on the maximum number of moves that the AI can make, and your evaluation function performs correctly on some sample boards we provide.
30 points	:	Your AI defeats a random player >= 60% of the time.
30 points	:	Your AI defeats an agent using OpenMoveEvalFn that searches 4 levels deep >= 60% of the time.
20 points	:	Your AI defeats an agent using OpenMoveEvalFn that uses iterative deepening and alpha-beta pruning >= 60% of the time.
5 points	:	Your AI defeats an AI that uses Thad's secret evaluation function, iterative deepening, and alpha-beta pruning (a.k.a. Thad 2.0) >= 60% of the time.
5 points	:	Your AI defeats Thad 2.0 >= 90% of the time.
1 bonus point	:	Your AI defeats Thad 2.0.1 >= 90% of the time

4. Botfight!

In addition to the basic assignment, you will have the option to compete against your peers for the glory of being the Fall 2016 AI-Game-Playing champ. We'll set up a system to pit your AI against others, and we'll be giving out prizes for the top players. May the odds be ever in your favor.

If you wish to compete in the tournament, simply include a plaintext file with a description of your agent, entitled 'AI.txt', and your CustomPlayer instance will be enlisted.

If you compete in the AI tournament and earn 1st, 2nd or 3rd place, you will receive a bonus:

- **1st place**: A letter of recommendation from Thad, plus 3 bonus points on your final grade.
- **2nd place**: 2 bonus points on your final grade.
- **3rd place**: 1 bonus point on your final grade.

5. Due date

The assignment is due September 11th, 2016 by 11:59PM UTC-12 (Anywhere on Earth time) on Bonnie (https://bonnie.udacity.com/, see Section 6 for details) and on on T-Square. [How to change your T-Square time zone] as a zip file.

The deliverables for the assignment are:

- A completed player_submission.py file with code copied from the relevant sections of player_notebook.ipynb
- A brief plaintext description of the inner workings of your agent, if you are competing in the tournament (AI.txt).

6. Testing

Udacity has been kind enough to setup test servers for us at https://bonnie.udacity.com/. Login with your GT username and password to access this service.

The instructions in the README detail the process for submitting your assignment for grading. We are going to limit you to 2 submissions within any 1 hour window, and the last submission that you make will be used to evaluate your grade. When you submit, you will be asked to acknowledge both the honor pledge and the assignment late policy. After that, your agent will run against our suite of tests and your grade will be output to the console. This output will necessarily be a truncated version of the full output. To view the full game logs, login to bonnie (link above) and navigate to *assignment_1*. You will have access to the logs in JSON format from this portal.

7. Tips and Notes

- There are various tests we will be running on your agent. You should not change the signatures of any method provided in the notebook.

- You can use the ‘time_left’ argument passed to the move() function to get the amount of time remaining if you are using depth limited search or iterative deepening. It gives the time remaining in milliseconds. This is a function, so to get the time you will need to call it as time_left().

- If you are having issues with setting up and running the code, Udacity has also provided us with Vagrant (https://www.vagrantup.com/) images that you may use. To use the vagrant image for this assignment, navigate to your local copy of the repository. Then run the following:

```
vagrant init scbrubaker02/compphoto
vagrant up --provider virtualbox
vagrant ssh
# Your code lives in /vagrant within this new virtual machine
```

8. Resources

Links for installing Python on your computer:
- OS X (http://docs.python-guide.org/en/latest/starting/install/osx/)
- Windows (http://docs.python-guide.org/en/latest/starting/install/win/)
- Linux (http://docs.python-guide.org/en/latest/starting/install/linux/)

Installing IPython notebook (we recommend using pip):
- http://ipython.org/install.html (get version 3 or above)

IPython Tutorial:
- http://ipython.org/ipython-doc/stable/interactive/tutorial.html

Git:
- https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository - Getting started with git
- http://stackoverflow.com/questions/tagged/git - Stack Overflow is a great resource
- Ask your friendly neighbourhood TA if all else fails