COMS 4701 – Artificial Intelligence – Fall 2014

Assignment 2: Game Playing in Python

(version 1, October 1, 2014)

Due: Wednesday, October 15, 11:59 PM

The aim of this assignment is to create an agent that can accurately play Othello using a minimax strategy with alpha-beta pruning.

To get you started, we've provided three agents in the *~cs4701/Project2/engines* directory. These agents can already play against each other. To start up an Othello game, please follow the instructions of the `README` file located on *~cs4701/Project2/README*.

The *engines* directory also contains a student agent. For now, this is a just a copy of the greedy agent that you can use as a starting point for your project.

The lecture notes include a description of the minimax search and alpha-beta pruning algorithms. The book includes more detail on these topics.

I) Implementation of the algorithms

You have to edit the `get_move` function of the student engine. It should return a numeric pair (x,y) of coordinates that indicates the move that your agent chooses to make.

You can implement any helper function you want in the `StudentEngine` class but you shouldn't modify the skeleton of the `get_move` function.

1) A minimax Othello player **(40 points)**

Using the starter code of the student engine, implement **a player for the Othello game that uses the minimax search algorithm** to select its move. Document your heuristics and algorithmic choices in your report.

Your agent should be capable of consistently beating the random agent. It should also be efficient and select its move in a reasonably short amount of time.

You may have to find a compromise: an algorithm that perhaps doesn't always find the best possible outcome, but that makes a good decision in a reasonable amount of time while using a reasonable amount of memory (you can enforce a cutoff at a fixed depth by defining a class attribute in the `StudentEngine` class).

In writing your Othello player, you may find useful to examine and understand some methods of the `Board` class located on *~cs4701/Project2/board.py*. We especially suggest that you take a look at:

- `count` – get the number of pieces for a specified color
- `get_legal_moves` – return a list of all valid moves for the player whose turn it is

You're free to reuse these functions in your code.

Also, note that there are several research papers and web sites that discuss possible heuristics. You're welcome to make use of these as long as they are properly cited in your report. Using any external code is strictly prohibited and can result in a grade of 0 on this assignment.

2) An alpha-beta Othello player **(40 points)**

Modify your implementation so that it uses **alpha-beta pruning** during search.

Keep the alpha-beta pruning part optional. The `StudentEngine` class already defines a class attribute `alpha_beta` for this purpose:

```
class StudentEngine(Engine):
    (…)
    def __init__(self):
        self.alpha_beta = False
```

You may then use the `get_move` function as a wrapper function for your two implementations:

```
def get_move(…):
     …
     if (self.alpha_beta == False):
          do_minimax(…)
     else:
          do_minimax_with_alpha_beta(…)
     …
```

Testing your agents

Develop your program somewhere in your home directory that is not the "key folder" you had to create for Project1. Create a symbolic link to it in the *engines* directory and use appropriate permissions so that other students can't see your code:

```
$ ln –s ~uni/uni.py ~cs4701/Project2/engines/uni.py
$ chmod 711 ~uni/uni.py
```

Remember that you should not create any soft link to a file located in your "key folder". Otherwise, running the `ls` command on the *engines* directory would leak your key.

Once the soft link is created, you can use your agent in an Othello game:

```
$ python ~cs4701/Project2/othello.py uni greedy
```

You can enforce the use of alpha-beta pruning for the black and white engines by specifying the command line options `–aB` and `–aW` respectively. The default mode is minimax only:

```
$ python ~cs4701/Project2/othello.py uni greedy -aB
```

Calling these options on the other agents (greedy, human and random) has no effect.

Only when you think that you're ready to submit, copy your program to your hidden directory (as mentioned later in the assignment). But once again, do not create a symbolic link to it.

II) Comparison of your implementations (20 points)

Design and conduct some **experiments to determine the following statistics** on the search process, for both the minimax and alpha-beta players:

- the number of nodes generated
- the number of nodes duplicated (containing states that were generated previously)
- the average branching factor of the search tree
- the runtime of the algorithm to explore the tree up to a depth of D, for different values of D

Provide a short description of the experiments you performed and discuss your results.

III) Tournament (bonus points)

A class tournament will give you a chance to test your code against that of other students.

It will consist of several group stages, in which each student will meet all other contestants of his/her group. The initial format will be 12 groups of 12 players, including some players provided by the course staff. The matches will be timed, so that each player will have 60 seconds total of CPU time to use during each game. The less efficient programs will be eliminated after each stage and new groups will be formed consisting of the top 3 agents from the previous groups. A final group stage with the top 12 performers will then determine the champion.

A second tournament will also be conducted, in which your agents will be given less time to select their moves. This will help us to distinguish the most efficient implementations of the algorithms.

We will give **discretionary bonus points depending on your performance** in the tournaments. The authors of the three top-ranked programs will also be invited to say a few words in class about the secrets of their implementations.

Submission instructions

As in the first assignment, you will create a *Project2* directory in the folder named after your unique key. Put all your files in this directory.

Remember to make sure that we have the right permissions to access your files.

What to submit:

- Your custom `StudentEngine` class under the name *uni.py*. It has to run on the CLIC machines (use Python 2.7.3).
- A report *readme_uni.txt* documenting your code, and justifying your heuristics and other algorithmic choices.
- A file *q2_uni.txt* containing your written answer to part II.

NB: We will do a dry run of all submissions on October 16 and give you until October 18 to make your agents more robust or faster.