Last modification: January 27, 2020

CS4341: Intro to AI
C-Term 2019/2020
Project 1: Connect N

1 Introduction

The aim of this project is to **make an AI for the game Connect-N**, which is a more general version of Connect-4. If you don't know Connect-4, try it out here: https://www.gimu.org/connect-four-js/jQuery/alphabeta/index.html.

2 The Game Code

The code is stored in a GitHub repository: https://github.com/NESTLab/CS4341-projects/. You can download it from the command line with the following command:

```
$ git clone https://github.com/NESTLab/CS4341-projects/
```

The code for this project is located in `CS4341-projects/ConnectN`. There are five files:

- `run.py`: the main script that creates and executes a game;
- `game.py`: defines the `Game` class, which manages the board and the players and declares the winner;
- `board.py`: defines the `Board` class, which contains the state of the board;
- `agent.py`: defines the `Agent` class, along with two sample agents that you can use to test your AI: `RandomAgent`, which picks a move completely at random, and `InteractiveAgent`, which allows a human to play the game;
- `alpha_beta_agent.py`: defines the `AlphaBetaAgent` class, which you must implement to complete this project.

Bug bounty: This code was made by the professor on purpose for this project, mostly by night and fueled by coffee. There might be bugs. For each bug you might find, submit a GitHub pull request with the fix and you will get **5 extra points** on your grade. The extra points will be awarded only to the first one to find a specific bug and provide a solution.

3 Your Task

Your task is to complete the `AlphaBetaAgent` class, whose definition is as follows:

```
import math
import agent

############################
# Alpha-Beta Search Agent #
############################

class AlphaBetaAgent(agent.Agent):
    """Agent that uses alpha-beta search"""

    def __init__(self, name, max_depth):
        super().__init__(name)
        self.max_depth = max_depth

    def go(self, brd):
        """Search for the best move (choice of column for the token)"""
        # Your code here

    def get_successors(self, brd):
        """Returns the reachable boards from the given board brd.
           The return value is a tuple (new board state, column number
           where last token was added)."""
        freecols = brd.free_cols()
        if not freecols:
            return []
        succ = []
        for col in freecols:
            nb = brd.copy()
            nb.add_token(col)
            succ.append((nb,col))
        return succ
```

As you can see, you must implement the `AlphaBetaAgent.go()` method. This method must perform a minimax search with alpha-beta pruning, as explained in class.

To help you with the implementation, the method `AlphaBetaAgent.get_successors()` is already written: this method, given a `Board` object, goes through all the legal moves for the current player and returns a list of tuples. Each tuple has, as first element, a new board and, as second element, the action performed to get to that board. If the board given as input is full, i.e., no more tokens can be added, `AlphaBetaAgent.get_successors()` returns an empty list.

You can modify `AlphaBetaAgent` in any way you want. For instance, you can rewrite `AlphaBetaAgent.get_successors()` if you prefer a different definition. However, **do not modify the other files we provide**. Those must remain the same, because we will set up a tournament after the deadline of project.

In the tournament, we will limit the time to pick a move to 15 seconds. If your AI takes more than 15 seconds to return a move, it will lose the game. Your AI will lose the game also if it returns a wrong move.

4 Finding a Good Evaluation Function

The main challenge in your work will be to devise a good **evaluation function for non-terminal boards**. In other words, given a board in which no player has already won, who is most likely to win?

You can get some ideas from these websites, which focus on Connect-4 rather than Connect-N:

- http://tromp.github.io/c4/c4.html
- http://blog.gamesolver.org/
- https://www.gimu.org/connect-four-js/

You are encouraged to look on the Internet for research, inspiration, and tips. Also, engage with the professor and the TAs about your approach.

5 Deliverables

The Design Document

The design document must report your final design. In particular:

1. How did you test your agents?
2. What did your tests reveal? Which heuristics turned out to the best among those your tried?
3. How would you improve your solution? Heuristics, code structure, ...

The document does not need to be long. 3–4 pages including diagrams is a reasonable length.

Directory Structure

Structure your code in a folder called `LastnameFirstname` as follows:

```
LastnameFirstname/
  __init__.py
  alpha_beta_agent.py
  <other files you created>
```

For example, if your name is Donald Duck, then the directory name is `DuckDonald`.

The file `__init__.py` is used by Python to initialize modules and packages. For the purposes of your project, this is just an empty file — the important thing is that it is present in the folder.

Writing the Code

The file `alpha_beta_agent.py` contains your AI. A skeleton for this file is provided for you in the repository. Copy it in your folder, and modify it. There are three important guidelines regarding editing this file.

The last line of your script

Your file must have as last line the following:

```
THE_AGENT = AlphaBetaAgent("LastnameFirstname", ...)
```

The purpose of this line is to create a module-global variable called `THE_AGENT` that we can import in our tournament. The call to the class constructor `AlphaBetaAgent()` allows you to set the parameters of the class. The first parameter must be the name of the agent (just use your `LastnameFirstname`), and the other parameters are decided by you. For instance, the basic skeleton provided to you expects a value for `max_depth`, so the call would be:

```
THE_AGENT = AlphaBetaAgent("LastnameFirstname", 4)
```

If you have more parameters, modify this line as needed.

Importing custom files

Say that your `alpha_beta_agent.py` script must import a file you created called `heuristics.py`, stored in the same directory as `alpha_beta_agent.py`. The correct way to import it is as follows:

```
from . import heuristics
```

This instructs the Python interpreter to look for `heuristics.py` in the current directory (`ConnectN/LastnameFirstName/`) rather than in `ConnectN/`.

There will be a time limit

Your AI will be given a total of 15 seconds to complete its choice. If the choice is not completed within this time limit, your AI will lose the game. This means that your AI should be relatively fast at evaluating a board configuration. Therefore, simpler solutions might be better than complex ones, and limiting search depth will be important.

How We Will Test Your Code

Make sure that your code works out-of-the-box. We will run your code doing the following operations:

```
# Go into ConnectN folder
$ cd CS4341_Projects/ConnectN/

# Unzip all the submissions
$ unzip /path/to/submissions/*.zip

# Generate tournament among all players
$ python3 gen_tournament.py

# Run tournament
$ python3 run_tournament.py
```

The script `gen_tournament.py` goes through all the unzipped archives and creates a script `run_tournament.py` with these elements:

```
from BloomJoshua.alpha_beta_agent import THE_AGENT as BloomJoshua
from MouGuanyi.alpha_beta_agent import THE_AGENT as MouGuanyi
from PinciroliCarlo.alpha_beta_agent import THE_AGENT as PinciroliCarlo
... more people

... tournament code

agents = [
    agent.RandomAgent("random"),
    BloomJoshua,
    MouGuanyi,
    PinciroliCarlo,
    ...
]

... run tournament using 'agents' as list of participants
```

Grading and Tournaments

Design document. We will evaluate your document according to its clarity, correctness, and completeness with respect to the guidelines mentioned above.

AI code. The AI code will be given full points if it defeats the random agent consistently. A dedicated tournament will be run between your AI and a few random agents. If your AI wins it, you'll get full points.

Penalties. Any mistake in following the submission guidelines (file names, directory structure, etc.) will incur in a **10%** point loss. If your code doesn't work out-of-the-box, you won't be allowed in the tournament.

Tournaments and bonuses. The tournament will be performed with

- Two grid sizes: 7×6 and 10×8; and
- Two values of N (i.e., the number of tokens to line up): we'll use $N = 4$ and $N = 5$.

This generates 4 separate tournaments. Each tournament will produce cumulative bonus points:

- The top ranked (1st) will receive a **15%** bonus;
- The next (2st) will receive a **10%** bonus;
- The next (3rd) will receive a **5%** bonus;

This means that, if your AI won all of the tournaments, you'd get $15\% \times 4 = 60\%$ bonus.