Artificial Intelligence

COMS 4701 Section 2 – Fall 2014

http://www1.ccls.columbia.edu/~ansaf/4701/

Home Work n°2: Adversarial search

Due Sunday October 19th, 2014 @11:55pm

Problem: 2048 Puzzle

Have you ever played 2048 puzzle? If not, give it a try!
http://gabrielecirulli.github.io/2048/)

The 2048 puzzle is played on a simple gray 4×4 grid, with numbered tiles that slide smoothly when a player moves them using the four arrow keys.Every turn, a new tile will randomly appear in an empty spot on the board with a value of either 2 or 4. Tiles slide as far as possible in the chosen direction until they are stopped by either another tile or the edge of the grid. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. The resulting tile cannot merge with another tile again in the same move. Higher-scoring tiles emit a soft glow.

http://en.wikipedia.org/wiki/2048_(video_game)

In this assignment, you are to implement the AIs programs for this game. Unlike some chess games, two players in this game (computer and human) take different actions in their turn. But adversarial search can be applied to this game as well. Have fun!

1 Implementation Detail

To simplify your work, a skeleton code is provided to help you test you program on your own machine. This code includes:

1. GameManager.py - Read-only. This module will load your AIs and have it compete automatically.
2. Grid.py - Read-only. This module provides Grid object with some useful operations - move, getAvaliableCells, insertTile and clone (some of them may not be efficient enough). You may use these operations in your AI code directly. However, to get a better performance, you are encouraged to write your own code which can ONLY be used in your AI internally. (This means your AI's getMove function still takes a Grid object as parameter, but it does not matter whether you use those provided operations or not.)
3. BaseAI.py - Read-only. All your AIs should inherit from this module and implement getMove function which takes Grid as parameter, return a move (different move for different player).
4. ComputerAI.py - Writable! Inherited from BaseAI, returns a computer action which is a tuple (x, y) indicating the place you want to place a tile. You NEED to modify this file to make it as intelligent as possible.

5. PlayerAI.py - Writable! Inherited from BaseAI, returns a number which indicates the player's action (0 - > UP, 1 - > DOWN, 2 - > LEFT, 3 - > RIGHT). You NEED to modify this file to make it as intelligent as possible.
6. BaseDisplayer.py, Displayer.py - Print the grid, feel free to change them to make the appearance better and you are encouraged to share your displayer's code. You don't need to submit those two files.

[Note: If you find bugs in the "Read-only" file(s), please let us know as soon as possible.]

2 Deliverables

2.1 Prepare required files

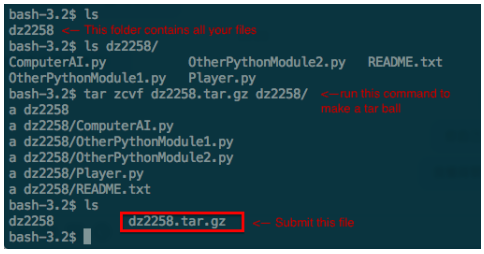
Here is a list of files you need to include in your submission.

1. ComputerAI.py – Name should be EXACTLY the same. Your computer AI module.
2. Player.py - Name should be EXACTLY the same. Your player AI module.
3. README.txt - Name should be EXACTLY the same. Your name, uni, email and a brief discription of your AIs.
4. Other .py file if you have extra modules.

2.2 Make a tar ball or a zip file

Use tar command to compress all your files, all your code should be contained in a folder which name is your uni.(e.g. as my uni is dz2258, the folder name should be dz2258).

Here is an example for uni = dz2258



2.3 Submit!

Submit your tar ball to `https://courseworks.columbia.edu`.

3 Basic Requirements

1. MUST use adversarial search in your AI (MinMax and $\alpha - \beta$-pruning).
2. MUST provide your move in 1 sec .
3. Your code must be tested on clic linux machine or other machines with equivalent environment. (Python version: Python 2.7.3)
4. Grade depends heavily on the max tile value your program gets to.
5. IMPORTANT: We will test your two AIs separately, so try to make them as aggressive as possible.

4 Hints

Here is a basic AI written in Javascript. `http://ov3y.github.io/2048-AI/` (you are supposed to do better!).

More hints? Sure! Here is one answer in stackoverflow which may be helpful. `http://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048`