ASSIGNMENT 2: ENTROPY (Phase 1)

Goal: The goal of this assignment is to learn the adversarial search algorithms (minimax, expectiminimax, and alpha beta pruning), which arise in sequential deterministic/probabilistic adversarial situations.

The Game of Entropy(N):

Setup:

The game board is NxN. There are N colors and N tiles of each color. Typical values for N are 5 and 7.

Objective:

There are 2 players, Order and Chaos. Order's objective is to move colored tiles on the board such that the number of palindromes made on the game board when all tiles have been placed is maximized. Chaos's objective is to place these colored tiles on the board such that this is prevented.

A palindrome is a sequence of tiles (in the horizontal or vertical direction) that is exactly the same when looked at from either end. Some examples of valid palindromes (assuming (R)ed, (B)lue, (G)reen as tile colors) are RGBGR, RRR, RR, GGBGG. Note that we will *not* be considering diagonal moves, or scoring in this game.

Player Roles:

Chaos - On each turn, Chaos randomly draws a single tile from the bag of remaining tiles and places it on an empty space of his/her choice on the game board. For our game, the server will be giving Chaos a colored tile drawn uniformly at random from the bag of remaining tiles.

Order - On each turn, Order is allowed to move a single tile already on the game board, either horizontally or vertically (not diagonally). The tile can be moved an arbitrary number of spaces in either direction (>= 1) as long as no tile is jumped over (i.e. occupied slots on the board block tile movement).

The game proceeds turn by turn, beginning with Chaos. When all tiles have been placed on the board, the total score is calculated, and then the players switch sides. *The player scoring higher as Order wins*.

Scoring:

All palindromes of length >= 2 are counted. Some examples,

GGGGG = 5 + 4*2 + 3*3 + 2*4 = 30

Explanation - All contiguous substrings are palindromes.

RGBGR = 5 + 3 = 8

Explanation - GBG is a substring palindrome of length 3.

RGRGR = 5 + 3*3 = 14

Explanation - RGRGR/RGR/GRG/RGR.

Algorithm: Implement this game as an instance of expecti-minimax search with cutoff and alpha-beta pruning. Learn a rudimentary evaluation function (define features and set weights of features either by hand or by a learning procedure pitting your player against yourself). You are encouraged to gain insight in your player by pitting it against other teams.

Note that you are required to code up 2 AI agents, one for Order and one for Chaos -- they may use the same algorithm and evaluation function, or not.

What is being provided:

Your assignment packet contains code for the game server in python and starter code for your player in Python that: (1) interacts with the game server, (2) allows you to manually send moves to the server. At the server end, you are provided a GUI which allows you to visualize the proceedings of the game.

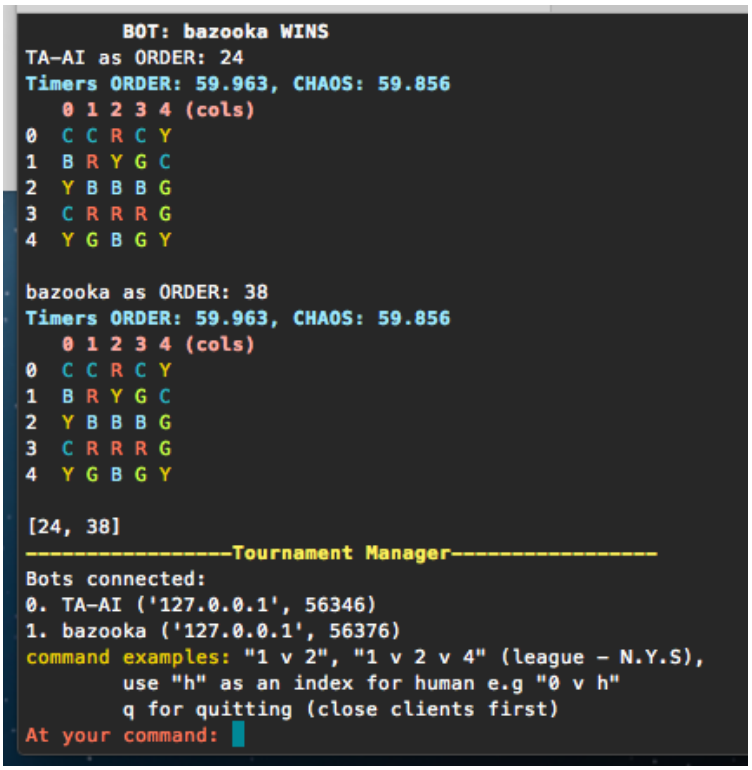
The server code can be found at: https://bitbucket.org/donraj/entropy/overview

Interaction with Game Server:

Note that you will require Python 2.x. The server won't work with Python 3.x.

Running the server:
The server that you are given will run via the command:
python server.py <port> <max-clients>
Make sure to cd into the server folder. You are also given a config.txt file, where you can specify the same arguments as above, if you would like to fix them (i.e. port and max-clients).

Running the client:
cd into the client folder
python client.py <port-no> <server-ip>
Open the server and client from two different terminals. The server will allow <max-clients> number of clients to connect to it.



Initialisation:
The server sends initialization information to the clients in the format:
N
player

where player is either equal to ORDER or CHAOS.
Eg -
5
CHAOS
tells you that the board size is 5x5 with 5 tiles each belonging to 5 colors and that you must play as Chaos.

Sending/Receiving moves to/from the server:
As Order:
You will receive Chaos's move from the server in the following format:
row column C
where (row, column) is the coordinate (0-indexed, standard matrix notation) where Chaos placed his colored tile. C is a single upper-case alphabet (A-Z) denoting the tile color.

The format for sending a move to the server is:
row1 column1 row2 column2
where the piece is being moved from (row1,column1) to (row2,column2). Note that the following moves are invalid:
a) (row1,column1) is empty.
b) (row2,column2) is occupied.
c) (row2,column2) does not lie in the same row or column as (row1,column1).
d) There is an occupied spot between the destination square and the initial square (i.e. the path is blocked).

As Chaos:
The server will first send you a colored tile (drawn uniformly at random from the bag of remaining tiles) in the format:
C
where C is a single upper-case alphabet (A-Z) denoting the tile color.

On receiving the tile, the format for sending a move is:
row column
where you are placing the colored tile given to you by the server on (row,column). Note that it is invalid to place this tile on an occupied square.

The server will then send you Order's move in the following format:
row1 column1 row2 column2

The process is then repeated.

You are also expected to keep track of opponent's moves and game state on your own. There is a total time assigned to you that gets decremented when your turn is going on which you should aim to stay inside.

Important:

- Your program needs to flush output buffers at every newline. This can be done by using endl at every move in c++ and using the -u flag for python Eg - python -u code.py
- Common mistake: Since stdin and stdout are reserved for communicating with the tournament program, please do not redirect stderr to stdout, and you can use stderr (and not stdout) for personal debugging purposes.

Code: Your code must compile and run on **machine named 'todi' or any machine with similar configuration present in GCL.** Please supply a compile.sh script for compilation. Also supply a shell script run.sh. Executing the command ./run.sh server port should start your player and start interacting with game server.

What to submit?

1. Submit your code for your game player. **The code should be contained in zip file named in the format <EntryNo>.zip.b64** If there are two members in your team it should be called <EntryNo1>_<EntryNo2>.zip.b64 Make sure that when we unzip the following files are produced:

   compile.sh

   run.sh

   writeup.txt

   You will be penalized for any submissions that do not conform to this requirement.

   Your code must compile and run on our VMs. They run amd64 Linux version ubuntu 12.04. You are already provided information on the compilers on those VMs. These configurations are similar to GCL machines like 'todi' (have a RAM of 16 GB)

2. The writeup.txt should have two lines as follows
   First line should be just a number between 1 and 3. Number 1 means C++. Number 2 means Java and Number 3 means Python.

   Second line should mention names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None.

   After these first two lines you are welcome to write something about your code, though this is not necessary.

Code verification before submission: Your submission will be auto-graded. This means that it is absolutely essential to make sure that your code follows the input/output specifications of the assignment. Failure to follow any instruction will incur significant penalty. The details of code verification will be shared on Piazza (similar to A1).

Evaluation Criteria

1. In Phase 1 we will test your code against simple baseline players for Entropy(5). Your final score will be scored as "your score as Order – opponents score as Order".

2. Extra credit may be awarded to standout performers.

3. The Phase 1 and Phase 2 of the project jointly carry weight of two programming assignments. Phase 1 carries only 25% of this weight and Phase 2 carries 75%.

What is allowed? What is not?

1. You may work in teams of two or by yourself. If you work in a team of two then make sure you mention the team details in the write-up. Our recommendation – this will lead to the much more open ended final project (phase 2); hence best to work with a partner with whom you communicate well. You will be allowed to use the same partner for the final project. Also, you cannot use the partner from previous assignments.
2. While you are allowed one of three languages we recommend that you use C++ since it produces the most efficient code. This assignment requires you to play the best game within a given time constraint.
3. Your code must be your own. You are not to take guidance from any general purpose AI code or problem specific code meant to solve this or related problem.
4. It is preferable to develop your algorithm using your own efforts. However, we will not stop you from google searching.
5. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.
6. You get a zero if your player does not match with the interaction guidelines in this document.
7. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.