LINGI2261: Artificial Intelligence
Assignment 3: Part I: Sarena Instructions

Jean-Baptiste Mairy, Cyrille Dejemeppe, Yves Deville
September 2012

Guidelines

- These instructions are provided to give you an overview of the Sarena game.
- The content of assignment 3 will be described in another document.
- You should become familiar with the Sarena framework because it will be used in the third assignment.
- If you encounter a bug while using the provided framework, report it clearly to cyrille.dejemeppe@uclouvain.be.
- *Do not* spend too much time implementing an artificial agent to play Sarena. Future lectures will provide you with the material required to implement an adversarial search agent. For now, *focus on current assignments*, the third assignment is due four weeks after the assignment statement will be released. You will have a sufficient amount of time to implement a great artificial player during these four weeks.

1 Rules

The rules provided here slightly differ from the original Sarena rules. The original Sarena game allows two to four players to play. Furthermore, players are not aware of the color played by opponents at the beginning of the game. For the third assignment, we consider a simpler version of the game in which only two players are allowed to play and both players are aware of the color played by their opponent.

1.1 Components

- 1 game board made of 36 circles (6x6). The circles are alternatively plain or containing two arrows.
- 36 chips with a different color on each side. There are originally four colors for the sides of the chips: white, black, red and yellow. Each type of chip is a combination of two different colors. There are six different types of chips, each of these types coming in six copies. As we will only consider the two player version, the white and black colors have been replaced by grey. This has no importance for the players since none of them owns those colors (they are assigned red or yellow).

1.2 Object of the Game

Own more chips than your opponent at the end of the game. You own all the chips that are in a pile that shows your color on top.

1.3 Setting Up the Game

1. Determine who will play first (can be done randomly or according to any other discriminating criterion).
2. Randomly place the 36 chips on the board (their orientation - i.e. which side of the chip is up or down - is also random), one per circle.

1.4 Gameplay

Starting with the first player, the two players take their turn alternatively. Each player must play during his turn and perform one movement on the board.

1.5 Movement Rules

A movement consists of moving a single chip or a pile of chips from one circle to an adjacent circle. Two circles are considered as adjacent if they are side to side along the horizontal or the vertical axis (not along a diagonal axis).

As part of this movement, a player can

- Stack the chips from one circle onto the chips on the adjacent circle
- Move the single chip or pile of chips to an empty circle *if it has arrows*. In doing so, he must also flip over the chip or pile of chips once it reaches the destination circle. *Note that if the destination circle is not empty, the move happens normally and without flipping*

As part of this movement, a player *cannot*

- Move a single chip or a pile of chips to a plain empty circle (a circle containing no arrows)
- Split up a pile of chips. Once a pile of chips is formed, it must move together as one
- Create a pile containing more than 4 chips

1.6 End of the Game

The game ends when no more chips or pile of chips can be moved.

Each player takes possession of the single chips and piles of chips showing his color on the top. Then each player counts how many chips he has. The player with the most chips wins.

In case of tie break, the player having the most chips with his color on one face is the winner. If the tie remains, the game is declared draw.

2 Python Framework

In the third assignment, you will have to imagine and implement an AI agent to play the Sarena game.

2.1 Files provided

The `sarena.zip` archive on the iCampus site of the course (Document › Assignments › Assignment 3) contains some Python code to get you started. The following files are provided:

- `sarena.py`: module containing the board representation, the agent interface and some other stuff. This will be the most useful file for you, so it is recommended to have a look inside and read the specifications thoroughly.
- `minimax.py`: implementation of the MiniMax and Alpha-Beta algorithms.
- `random_player.py`: an example player making dumb random moves.
- `player.py`: a skeleton of an Alpha-Beta agent you can use as a base for your own.
- `game.py`: main program for launching the game. Run with `-h` to show the usage notice.

- `gui.py`: graphical user interface. This file is used by `game.py`.

2.2 Internal Representation

Internally, the board is represented by a matrix of lists. We represent the state of a cell with a list which elements are as follows:

First element (index 0)

This element represents the type cell on the board. As explained in the rules, there are two different types of cells: plain circle or circle containing arrows. To represent the type of the cell, we use a single integer. The integer 3 represents a plain circle and 4 represents a circle with arrows.

Second to fifth elements (indices 1-4)

These elements represent a pile of chips. A pile of chips is represented such that the first element (index 1) is the bottom chip of the pile and the last element (index 4) is the top.
Each chip is represented as a list of two elements. The first element of the list (index 0) is the down side of the chip while the second element (index 1) is the top. The color of the side of a chip is represented with an integer. As we play a version of Sarena with only two players, we only need three colors: yellow (for player 1, represented by integer 1), red (for player two, represented by integer -1) and gray (for neutral color - represented by integer 2).
To represent a tower of height smaller than 4, we fill the remaining elements of the list with lists [0, 0].

Summary

The state of a cell will be represented in python as follows:

$$[\ \underbrace{\text{cell}}_{\text{cell type}}\ ,\underbrace{[b_1, t_1], [b_2, t_2], [b_3, t_3], [b_4, t_4]}_{\text{pile of chips}}\ ]$$

where *cell* represents the type of the cell (3 for plain circle and 4 for circle with arrows) and each pair $[b_i, t_i]$ represents a single chip with b_i and t_i respectively for the bottom and the top of the chip. $[b_1, t_1]$ is the bottom of the pile of chips and $[b_4, t_4]$ is the top. The values b_i and t_i can take are as follows:

- -1: red half-chip (player 2 color)
- 1: yellow half-chip (player 1 color)
- 2: gray half-chip (neutral color)
- 0: absence of chip (empty top of the pile)

Note that if b_i has a 0 value, then t_i is also 0 and vice-versa (a chip *cannot* be split in two half chips).

Examples

Two examples of how cell states are encoded in python are provided in Figures 1 and 2.

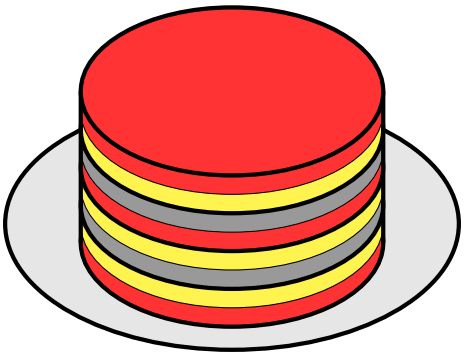

Figure 1: Example of pile of chips. The big gray circle below the pile of chips represents a cell with no arrows.

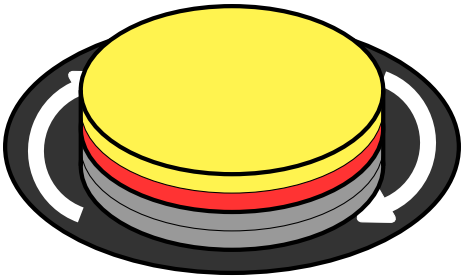

Figure 2: Example of pile of chips. The big black circle with the white arrows below the pile of chips represents a cell with arrows.

The state of the cell containing the pile of chips represented in Figure 1 would be encoded in python as follows:

```
cell1 = [3, [-1, 1], [2, 1], [-1, 2], [1, -1]]
```

The state of the cell containing the pile of chips represented in Figure 2 would be encoded in python as follows:

```
cell2 = [4, [2, 2], [-1, 1], [0, 0], [0, 0]]
```

3 Playing with the framework

The way the framework works is the following. The file `game.py` contains the implementation of the rules of the game and the graphical interface. It has to be launched with the URI's of the two players, or the keyword 'human' for you to play directly on the computer. Each non-human player is a server which communicates with `game.py` for the moves.

You can play Sarena against another human on the same machine using the following command line:

```
python3 game.py human human
```

As each AI agent is a small server to which `game.py` connects, it allows you to try your agent against the agents of another group without giving away your precious source code! To launch the random agent, you can enter in a terminal:

```
python3 random_agent.py -p 8000
```

To play against the random agent you just launched, you can enter in a different terminal:

```
python3 game.py human http://localhost:8000
```

To use the graphical user interface, you will need the Tk bindings for Python. Those are installed in the INGI infrastructure. On Debian-based systems, you just have to install the `python3-tk` package. To install the tkinter module on other OS's, refer to the tutorial provided at `http://www.tkdocs.com/tutorial/install.html`.
To see the different options available to launch a Sarena game, you can use the `-h` option as follows:

```
python3 game.py -h
```

The same holds for the random agent.

Licensing

The provided classes are licensed under the General Public License[1] version 2.

We will ask you to also license your code under the GPL version 2.

[1] `http://www.gnu.org/licenses/old-licenses/gpl-2.0.html`