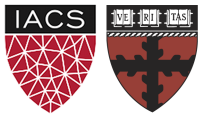

HARVARD
School of Engineering and Applied Sciences

ComputeFest 2014 Computational Challenge

ComputeFest is an annual extravaganza of skill- and knowledge-building activities for students and researchers interested in computational science and engineering, organized by the Institute for Applied Computational Science at SEAS.

Since January 2012, IACS has organized the Computational Challenge. In previous years, students have developed models and algorithms for providing medical aid to the maximum number of victims stranded by a hurricane striking the city of Cambridge, and developed fast algorithms to compete against one another in a virtual foosball arena.

This year, the challenge is to develop strategies and algorithms for a mind-bending two-person puzzle that we call Stack'Em. This game involves two players trying to arrange their bricks in a particular way while, at the same time, preventing the opposing team from accomplishing the same feat.

2014 Algorithmic Stack'Em Competition **2**
- Definitions . . . 2
- Gameplay . . . 2
- Wall . . . 3
- Game Engine . . . 3
- Competition . . . 4
  - Dry Run and Fun . . . 5
  - Final Competition . . . 5
  - The final . . . 5
  - Time duration . . . 5
- Contact Information . . . 6
- Fair-play . . . 6

2014 Algorithmic Stack'Em Competition

Your goal is to provide an "artificial intelligence" (AI) that can play our version of Stack'Em described in the rules below. Your computational strategy for determining each move made in the game will be pitted against other teams in the Stack'Em playoff tournament.

In the following sections, we define the game that you will be playing in this competition. There are many possible approaches to the problem that speak to the core of computational science: stochastic analysis, optimization, software design, high performance computing, etc. We hope that you use your talents to crush your opposition.

Definitions

Brick: A brick is used to build the wall. The bricks that we have available however are all of different weights. In all, we have a pile of 100 bricks, each with a different integer weight from 0 to 99.

Wall: A wall consists of a 4×4 grid of bricks. Each team is attempting to build their wall. A completed wall has all of its bricks in the right order. Each brick in the wall has a coordinate such as "A0", "D3", etc.

Completed Wall: A completed wall is a wall in which the weights of the bricks are in *decreasing* order along the bottom of the wall and in *decreasing* order reading up the wall.

Discard: As the wall is being built, the discarded bricks go into the discard pile. The weight of the last brick in the discard pile is known.

Pile: The pile is the random assortment of bricks we have available. The weight of these bricks are not known.

Gameplay

At the start of each game, each team receives an initial random wall.

One of the two players is randomly selected to go first. On each turn, a player

1. Receives the weight of the discard brick and the current state of the opponent's wall.
2. Chooses between using the discard brick or drawing from the pile.
3. Submits the choice and receives either the discard brick or a random brick from the pile.
4. Chooses where to place the brick in the wall. This may take into account the opponent's wall as well.

5. Submits the coordinate of the brick in the wall to replace. The wall is updated appropriately and the old brick is discarded. If the submitted coordinate is not valid (i.e. '-1') no brick in the wall is replaced and the new brick is discarded instead.

6. The first player to arrange his wall in decreasing order horizontally and vertically wins the game.

Wall

A wall is a two-dimensional array composed of bricks.

The wall is indexed as in the table below. The game engine accepts coordinates in the form 'xy' where x is the row label and is a letter from 'A' to 'D' and y is the column label number from '0' to '3'. **Note this is 0-indexed.**

	0	1	2	3
D	(3,0) D0	(3,1) D1	(3,2) D2	(3,3) D3
C	(2,0) C0	(2,1) C1	(2,2) C2	(2,3) C3
B	(1,0) B0	(1,1) B1	(1,2) B2	(1,3) B3
A	(0,0) A0	(0,1) A1	(0,2) A2	(0,3) A3

A completed wall is composed of bricks of decreasing weights as the wall is traversed to the the right and up. For example the wall below is a completed wall. Not that D3 must be the smallest brick and A0 must be the largest brick in any completed wall.

	0	1	2	3
D	22	18	17	4
C	79	37	30	6
B	90	42	39	8
A	97	80	56	10

The `print_wall` function in the provided codes prints out a wall in a format similar to the above table. Additionally, for convenience, each provided code also provides a `rowcol2coord` function to transform row-col indices, which may be useful in your computations, to string coordinates, which the server expects.

Note that if any invalid coordinate is sent to the game engine then the proposed brick is discarded and the wall is left unmodified.

Game Engine

We will be using a centralized game engine that keeps track of the game state and allows any programming language to communicate with it. This makes it easy to pit teams of different types and languages against each other.

You may download a selection of skeleton code that has been written to get you started communicating with this game engine:

Download codes here

or

`wget http://crisco.seas.harvard.edu/ComputeFest2014/Codes.zip`

These codes provide communication clients and skeleton players in Python, Matlab, C++, and Java.

To play, launch two players with the same commandline game ID. For example:

```
$ python player.py 1234
Waiting for game 1234
```

launches a player into a game with ID 1234. Another player can join this game by launching with the same game ID:

```
# After compiling with 'javac *.java'
$ java player 1234
Waiting for game 1234
READY
```

The second player is using Java here, but could also be the same Python player from above.

Once two players have joined a game, `READY` is broadcast to both and the game loop is entered. Both teams receive their initial walls, the first player is decided and makes the first move, the next player makes their move, etc, until a player completes their wall.

You will be editing the `player` file in the language of your choice. Specifically, edit

- The `choose_discard_or_pile` function takes your wall, your opponent's wall, and the current discarded brick and returns 'd' for discard or 'p' for pile.
- The `choose_coord` function takes your wall, your opponent's wall, and the chosen brick and returns a coordinate of the wall such as 'A2' or 'B3'.

Competition

Dry Run and Fun

On Wednesday at noon in Pierce 301, we will provide pizza to give the teams a chance to test some of their strategies against one another, trade ideas, and generally have a good time.

Final Competition

The final competition will be held on Thursday at 1:30pm in Pierce 301.

There are 10 teams.

Each team will face all other teams in 45 *matches* (see Table 1). A match is composed of 10 games.

The team that wins the most games in a match is awarded 2 points. If the two teams win the same number of games, 1 point is awarded to each team. The losing team is awarded zero points.

Table 1: The round-robin competition

Round 1	1-10	2-9	3-8	4-7	5-6
Round 2	1-9	2-7	3-6	4-5	8-10
Round 3	1-8	2-5	3-4	6-10	7-9
Round 4	1-7	2-3	4-10	5-9	6-8
Round 5	1-6	2-10	3-9	4-8	5-7
Round 6	1-5	2-8	3-7	4-6	9-10
Round 7	1-4	2-6	3-5	7-10	8-9
Round 8	1-3	2-4	5-10	6-9	7-8
Round 9	1-2	3-10	4-9	5-8	6-7

At the end of the round-robin tournament the two teams with the most total points move on to the **final**. In the case of a tie (teams with the same number of points), the teams will play each other in tie-breaker matches, 5 games per match.

The final

The final match is composed of 12 games and will happen at the end of the round-robin phase. In case of a tie, there will be an additional 5 game match.

Time duration

Each team is given 1 minute to make their moves. This is the total accumulating time during the game and is calculated at server side. A team that exceeds the limit automatically loses the game.

Contact Information

Game Engine, Rules of the Game, Disputes:
Cris Cecka, Cruft 402, `ccecka@seas.harvard.edu`

SEAS computational facilities:
Kumar Indireshkumar, `kindires@seas.harvard.edu`

Competition:
Cris Cecka, Cruft 402, `ccecka@seas.harvard.edu`.
Pavlos Protopapas, `pprotopapas@cfa.harvard.edu`

Fair-play

No external collaborations are allowed. This includes consulting forums, friends, family, or other students.

Abuse of the game server is not allowed. This includes excessive connections or purposely invalid moves.