JIGGA JIGGA – BOVINE BORDERS

@LemonKat & @Vodiboi

SHORT SUMMARY

In this year's Jigga Jigga, four players, henceforth called "farmers", compete for cow grazing territory on a 20x20 grid of land by moving their cows or placing new cows. Farmers are eliminated when their barn is captured. Last farmer remaining wins.

GAME RULES

LAYOUT

Each square on the grid may be open land or have one cow on it. Each farmer starts with a cow in a corner of the board. This corner is their barn, and should be protected.

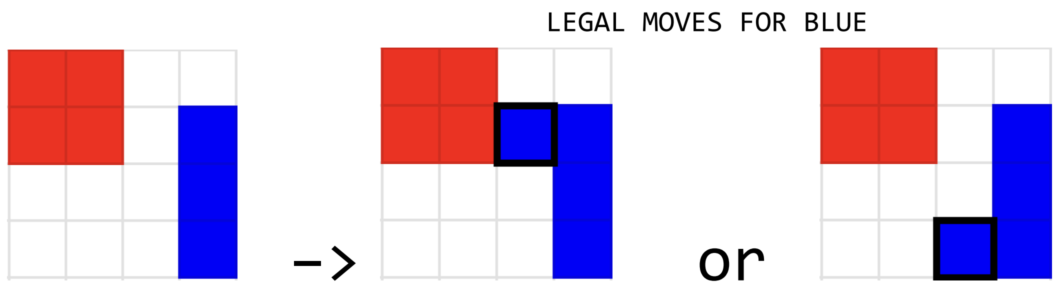
GAME FLOW

The game consists of 10,000 rounds. At the start of a round, each farmer specifies their move. Once moves from all farmers are collected, they are executed in random order.

MOVING

A move consists of a source and a destination. The farmer specifies the destination, and can choose to either specify the source, or leave it out. If the source is left out, a random legal source will be chosen automatically. The source must be a square they already occupy (contains one of their cows), and the destination must be a square they currently do not occupy that shares a side with the source, as shown in Figure 1.

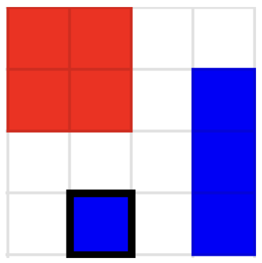
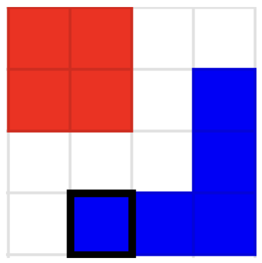
Figure 1.



If the destination is *empty*, the farmer *places a new* cow at the destination. If the destination is *occupied* by a rival farmer's cow, the farmer's cow *moves* from source to destination. The rival farmer's cow at the destination square suddenly suffers a critical existence failure and ceases to exist.

Things change if the source square is *not connected* to the farmer's barn (i.e. there is no path of squares the farmer's cows occupy from the source back to their barn, see Figure 2). If the source is *not connected* to the barn and the destination is *empty*, the cow *moves* from source to destination. However, if the source is *not connected* to the barn and *there is* a rival cow at the destination, *both cows* in source and destination have critical existence failures.

Figure 2:

NOT CONNECTED TO BARN	CONNECTED TO BARN
	

If a move cannot be legally made (e.g. out of bounds, farmer doesn't occupy the source, farmer already occupies the destination), it is not made.

A farmer can choose not to make a move during a round.

ELIMINATION

Once a rival farmer's barn suffers from critical existence failure, that farmer is eliminated from the game and all of their cows suffer from critical existence failure. All grazing territory formerly belonging to said farmer becomes open land.

WINNING

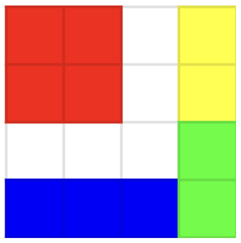
Rounds repeat until only one farmer (the winner) remains. If multiple farmers remain by round 10,000, the winner is the farmer with the most territory. If multiple farmers occupy the same amount of territory at the end of the game, they are declared tied.

MAKING A BOT

FORMAT

For those who are new to Jigga Jigga, every participant (single person or team) codes a python "bot", and in this Jigga Jigga your bot will play the role of a farmer. It will be given the game board as input, and should return a move as output.

The board will be given as a nested list of integers. -1 represents an empty square, 0 is one of your squares, and 1 through 3 are the other farmers' squares. For example, if you are red, and the other farmers are yellow, green, and blue, then the board:



would be given as

```
[
    [0, 0, -1, 1],
    [0, 0, -1, 1],
    [-1, -1, -1, 2],
    [3, 3, 3, 2]
].
```

To return your move, return a tuple of the source position and the destination position. Each position should then be a tuple of the form (row, col) where (0, 0) is the top left corner.

For ease of use, you may alternatively only return the destination position. To do this, simply return the destination tuple. If you do this, a random valid source will be automatically chosen. If no source is valid, your bot will pass its turn.

To pass your turn, return None.

It’s totally fine if your bot has some form of memory, such as global variables.

HELPER FUNCTIONS

We have created a few helper functions to assist you in making your bot.

dist(a, b):
returns the manhattan distance (X difference + Y difference) between positions a and b.

nearby(pos):
returns a list of all positions adjacent to the given position. These may be out of bounds.

in_bounds(pos):
returns whether pos is within the bounds of the 20x20 board.

reachable_from(board, pos):
returns a list of all the coordinates from which you can move to pos from.

allowed_moves(board):
returns a list of all the possible legal moves you could return.

allowed_dsts(board):
returns a list of all the possible legal destinations you could pick.

connected(board):
returns a 2d list of boolean values, where a value is True if and only if it is connected to your barn.

pick_random_source(board, dest):
returns a random source cell if you desire to move to dest. Note that this is just a random choice from reachable_from(dest).

SAMPLE BOTS

We have four sample bots available. Don’t question the names, I did it for the memes. – @LemonKat

pass_pompom: always passes its turn.
random_raven: returns a randomly chosen legal move.
defensive_dan_heng: presents a hard-to-penetrate diagonal front-line.

AdjacentAsta: returns moves are adjacent to its previous move.

CONFIGURATION

In the competition, your bot will be run like this, in main.py:

```
from board import Board
from <program0> import <bot0>
from <program1> import <bot1>
from <program2> import <bot2>
from <program3> import <bot3>

print(
    Board(
        [<bot0>, <bot1>, <bot2>, <bot3>],
        ["<bot0 name>", "<bot1 name>", "<bot2 name>", "<bot3 name>"],
        [<color0>, <color1>, <color2>, <color3>],
        display_mode="terminal",
        delay=-1,
    ).run()
)
```

where <botX> is the bot function of farmer X, <programX> is the program that bot is defined in, <botX name> is its name, and <colorX> is a tuple of 3 integers representing the bot’s color in RGB. Change the delay parameter to have the game wait delay seconds each round.

LOGGING

Because the display is in the terminal by default, logged events will be written to log.txt.

Logged events include bots throwing errors, taking too long, or returning invalid moves. The log will also record farmers being eliminated or the game ending.

To see log.txt updated in real-time, open a new terminal, navigate to the folder, and enter “tail -f log.txt”. When the game ends, exit by hitting ctrl-c.

COMPATIBILITY & DISPLAY

The NumPy library is **required** for the code to run. To install it, enter “pip3 install numpy” into a terminal. It provides fast, fixed-size, fixed-dtype, multi-dimensional arrays (stuff that makes the game faster).

There are two ways to display the game: Terminal and Tkinter.

Terminal mode will display the game in the terminal. This may not work in every terminal/IDE, but the built-in MacOS terminal (aka terminal.app) works.

Tkinter mode will open a tkinter window and display the game in that window.

Note that tkinter mode is ~3x slower than terminal, which is because tkinter is not really meant for games. The preferred mode is terminal, which will be used in all rounds of the actual tournament, and we highly recommend you use it if possible.

LIBRARIES

Feel free to use any libraries which can be installed using PIP. Examples:

1. NumPy
2. TensorFlow (if anyone pulls this off, I’ll be very impressed)
3. SciPy

SOME FINAL WORDS

There will be two divisions, a standard division and a hacker division. The standard division will be the normal main competition. The cheater division only has two rules:

1. Don’t damage my computer. E.g. consume extreme amounts of CPU, mess with the configuration or filesystem, hack it, etc.
    a. Specifically, all your actions must be cleared by the time Python shuts down, and none of your actions may make the Python shutdown harder.
2. Win.

A preliminary round will be held towards the end of block 2 so participants know who they’re up against, and the final competition will be held sometime during the next build week. To submit bots, just send ‘em by email to @LemonKat **and** @Vodiboi. Don’t forget to specify a name and a color in your email or in your code.

You thought that cows were peaceful.
Now you know.