Software Development Documentation

for

Fast Food Frenzy

Prepared by AWAP 2018

ACM@CMU February 24, 2018

1 Introduction

1.1 Backstory

As tech gets more involved with the service industry, 4 local Pittsburgh restaurants have taken an initiative to use **coding and algorithms*** to expand into the city. The four restaurants:

- Five Bytes
- The Original Big-O Shop
- Queue'd
- Mercur-iOS

are competing for the same locations around the town and have hired CMU students to write an algorithm that can predict the best moves to become the largest chain in the city. As a team, you will be representing one of these restaurants and will be competing against the 3 other that have also hired CMU students. The teams will be competing in a tournament style competition to decide which team has the smartest algorithm.

* https://www.reddit.com/r/ProgrammerHumor/comments/5ylndv/so_thats_how_they_did_it_its_brilliant/

1.2 Before You Begin

Install python3 on your machine:

```
Mac OSX: docs.python-guide.org/en/latest/starting/install3/osx/
Windows: docs.python-guide.org/en/latest/starting/install3/win/
Linux: docs.python-guide.org/en/latest/starting/install3/linux/
```

After installation, run this code in your terminal:

```
sudo pip3 install virtualenv
git clone git@github.com:Nikhil-Jog/awap2018-competition.git
cd awap-2017
virtualenv venv
source venv/bin/activate
pip install -r requirements.txt
```

1.3 Rules

The objective of the game is to take over as many territories as possible. Each team starts off with 1 node and 10 units, gaining 4 recruits to your empire per turn. Every turn you can use your recruits to take over other areas in the neighborhood to grow your empire.

Each turn is split into two phases: a placement phase and a movement phase.

Placement:

1. You may only place nodes on nodes that you own.
2. The total number of nodes you place cannot exceed the number calculated from the following formula. The first node you own is worth 1 unit. Any additional nodes are worth 0.9* the previous node's unit value.

```
4 + floor((1-pow(.9,n))/(1-.9)
where n is the number of nodes your faction currently owns
```

Movement:

1. Each movement must begin on a node you own and end on one of its neighbors.
2. You may move units between nodes you own for strategic purposes or to enemy nodes to attack.
3. You may move up to one less than the number of units on that node (you must leave at least one unit on each of your nodes).
4. Each individual unit may only move once per turn.
5. When attacking enemy nodes, the number of remaining units will be equal to the difference between the number of units moved and the number of units originally on the node.
6. If the number of units moved is equal to the number of units on the node, the defender (team who originally owned the node) will retain ownership with 1 unit remaining on the node.

1.4 Tournament

All competitors will be sorted in to randomly selected brackets. Each group will play on all three maps and player turn ordering will also be randomized.

The top score from each bracket will automatically move on to the next round. Afterward, the next best n relative scores in the entire round will move on to fill the remaining slots. Relative scores look at your performance compared to the winner of your bracket. The closer you are to your bracket's winner, the better your score is.

1.5 References

If during the competition you have any questions regarding API usage and/or administrative reason, please email:

```
Nikhil Jog: njog@andrew.cmu.edu
```

Or message us on our slack channel:

```
https://awap2018.slack.com/signup
```

2 Player API

2.1 Player_Template

Before you start, take a look at base_player.py. Your code will interact with this super class, so please read through all the comments associated with each function. DO NOT EDIT ANYTHING IN THIS FILE.
Of note are the two functions:

```
place_unit(self, node, amount)
move_unit(self, start, end, amount)
```

Which will package and format the given action. These should be called every time you want to add an action to your turn.
Functions to implement in player_template.py:

```
# Initialize any variables necessary to run your code/algorithms
def __init__(self, p_id):

# Initialize any player-specific turn code here
def init_turn(self, board, nodes, max_units):

# Insert any algorithms/logic required to place units.
# Remember to use functions found in base_player.py
def player_place_units(self):

# Insert any algorithms/logic required to move existing units.
# Remember to use functions found in base_player.py
def player_move_units(self):
```

One important note to remember while implementing your algorithms: if your moves are invalid, the main game will reject your move set and ignore any actions you've performed for that turn. Make sure you validate your moves.

Our board is represented by a networkx graph where each node is an int that maps to a dictionary containing the following fields:

- 'owner': ID of the player who owns the node
- 'old_units': number of units originally on the square (i.e. number of units available to move from that node)

- 'new_units': internal, for board update function to keep track of which units have been moved

Some useful functions include

- board.nodes: gives a NodeView which can be indexed directly or converted into a list.
- board.neighbors(node): gives an iterator containing the neighbors of node

For more information, refer to the networkx documentation:

```
https://networkx.github.io/documentation/stable/reference/introduction.
    ↪ html
```

This means that you do not need to implement any of those pesky graph algorithms!

2.2 Running the code

Run your code with the following command

```
python gameMain.py <player1> <player2> <player3> <player4> <boardname>
Available boards: flower, grid, ring
```

The players are the filenames of your AI's (without the .py extension). The game can be run with multiple copies of one AI or with different versions. There is a built-in visualizer using matplotlib which can be enabled by setting the VISUALIZE flag to 1 in gameMain.py. You may also call the function board.draw() in gameMain.py, which will draw the current state of the board.

2.3 Server

To view a animated replay of your code, upload to our website to test out your code and submit your final code. We have built a simple server that can be accessed here:

```
https://awap-2018-competition.herokuapp.com
```

Once you are on the page, use the key that was emailed to you. If you have not received a key, reach out to us as soon as possible. Once you are logged in, you can submit your code in 3 different ways:

- Private: Basically for testing purposes. Lets you pick each player by uploading a file. The latest 4 submissions can be replayed.
- Public: Matches your code with other players. You can only replay the game you've submitted your code for. The other teams that get matched with your submission are kept anonymous(no team names are displayed except yours). You can view your latest captured node count on the leaderboard to get a sense of where you are. Your score will be highlighted in blue.

- Competition: Submit your final code here. The submissions close at 6PM. You will be able to replay all the games as the round is finished. We will notify you when they are up.

Warning! There is a 360 second timeout for each game. Make sure your turns take approximately less than 450ms each.

Warning! If you submit code that tries to tinker with the server, we will disqualify your team. However, if you submit a player file that creates an stderr while running, that submission will be deleted. The rule of thumb should be to run it locally before uploading. **Code that is malicious to the server is against the rules and will result in disqualification.**

Extra Info To logout from your current team, go to /logout

Extra Info Submissions take time. The status will change from 'In Progress' to 'Completed' when the output is received.

Extra Info If during the competition you have any questions regarding the website, please email:

```
Deniz Sokullu: dsokullu@andrew.cmu.edu
```

Or message us on our slack channel:

```
https://awap2018.slack.com/signup
```