UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

School of Computer Science & Applied Mathematics

COMS4033A/7044A Assignment Report

1 Introduction

At this point, we have all the ingredients for our baseline agent. You will now be required to put everything together into an agent that can play the game against an opposing agent.

2 Baseline Agent

First, read the instructions for creating a bot here: `https://reconchess.readthedocs.io/en/latest/bot_create.html` . You will need to use the following structure when implementing your baseline:

```
class MyAgent(Player):

    def __init__(self):
        # setup agent as you see fit
        pass

    def handle_game_start(self, color, board, opponent_name):
        # function that is run when the game starts
        pass

    def handle_opponent_move_result(self, captured_my_piece, capture_square):
        # feedback on whether the opponent captured a piece
        pass

    def choose_sense(self, sense_actions, move_actions, seconds_left):
        # write code here to select a sensing move
        pass

    def handle_sense_result(self, sense_result):
        # This is where the sensing result returns feedback
        pass

    def choose_move(self, move_actions, seconds_left):
        # execute a chess move
        pass

    def handle_move_result(self, requested_move, taken_move, captured_opponent_piece, capture_square):
        # this function is called after your move is executed.
        pass

    def handle_game_end(self, winner_color, win_reason, game_history):
```

```
    # shut down everything at the end of the ga,e
    pass
```

You will need to implement the following:

1. `__init__`: setup anything you require, including a connection to Stockfish
2. `handle_game_start`: the starting position is provided to you. There is therefore no uncertainty about what the current state is. Record this as the current state of the game
3. `handle_opponent_move_result`: whether or not your opponent has made a capture provides you with information. Use this to narrow down the list of moves that could have been executed, and hence the possible list of states the agent is currently in. Note that if your agent plays as white, then this function is called at the beginning of the game, even though your opponent has not previously moved.
4. `choose_sense`: this is where you will choose where to sense on the board. Your baseline agent should select a square uniformly at random, but should ignore squares on the edges of the board (since this would make the window smaller than it needs to be)
5. `handle_sense_result`: this returns the window of the board around the sensing point. Use this to narrow down the list of possible states
6. `choose_move`: select a move to play using the majority voting strategy described previously. Since there will be many boards stored, you should set the time limit for Stockfish to be $10/N$, where N is the number of boards. Should the number of boards exceed 10000, randomly remove states to reduce the number to 10000. This will ensure the number of states being tracked doesn't expand exponentially.
7. `handle_move_result`: this is where you can update your states based on the outcome of the move, if the move was taken. Note that the move requested and the move actually executed may differ (for example, you wanted to move a piece from one square to another, but there was another piece in the way, and so the moving piece ended up at a different square). You could therefore use this information to rule out certain states.
8. `handle_game_end`: close the connection to Stockfish here

Hint: there are many ways to arrive at the same position, so storing the possible boards in a set is advisable

Submit a single Python file with your agent to Moodle. This will be run against a similar baseline and marks will be awarded only for agents that perform on par. Agents that fail to run (see `https://reconchess.readthedocs.io/en/latest/bot_play.html` for more on how to run) will receive 0.

3 Comparing to baselines

Once you have done this, use your baseline agent and run it in a round-robin tournament against the `RandomBot` and the `TroutBot`. See here for how to run games: `https://reconchess.readthedocs.io/en/latest/bot_play.html`. Call this agent `RandomSensing`.

4 Improvements

The current agent you've implemented senses randomly. But we can definitely do better than that. You are now required to implement the `choose_sense` in any way you want to improve performance. You may use the strategy adopted by `TroutBot`, but in that case, it cannot be your only improvement.

Some ideas for what to implement, such as picking sensing moves to minimise uncertainty, can be found in the report here: `https://proceedings.mlr.press/v176/perrotta22a/perrotta22a.pdf`. Whatever your choice, you should implement these changes to create the `ImprovedAgent`, and compare them to `RandomSensing`, `RandomBot` and the `TroutBot` in a round-robin format, where all agents play against each other twice, once as white and once as black.

Note that you are encouraged to make more than one improvement. If you do so, please document it in the report.

5 Report

Write a (maximum 2-page) report detailing the results from your double round-robin tournament involving the four agents, as well as describing all the improvements that went into creating `ImprovedAgent`.

Submit to Moodle the following files:

1. A PDF of the report. Be sure to include your names and student numbers. Only one member for each group needs to submit
2. All of your code. Ensure all of your code is in a single file. Note that your code must be runnable (i.e. the code must successfully work when running `rc-bot-match`). Any submission without code or whose agent fails to run will receive 0.

6 Competition

Finally, your `ImprovedAgent` agent will be run in a tournament against everyone else's submissions (which is why it may be a good idea to make multiple improvements). Your mark for this portion will depend on your outcome in the final ranking. However, your agent will be required to defeat the `RandomSensing` agent to score any marks whatsoever.