AI 3603 Artificial Intelligence: Principles and Techniques

Homework 2

Adhere to the Code of Academic Integrity. You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. Moss (Measure Of Software Similarity) will be used for determining the similarity of programs to detect plagiarism in the class (`https://theory.stanford.edu/~aiken/moss/`) If you encounter some difficulties or feel that you cannot complete the assignment on you own, discuss with your classmates in Discussion forum on Canvas, or seek help from the course staff.

You are required to complete this homework ***individually***. Please submit your assignment following the instructions summarized in Section 6.

1 Task Introduction

For this homework assignment, you are required to implement the **minimax algorithm with Alpha-Beta pruning** in a new checkers environment. Your agent should evaluate the current board state and select an action that maximizes its chances of winning against the opponent. Please refer to Section 2 for detailed rules and guidelines.

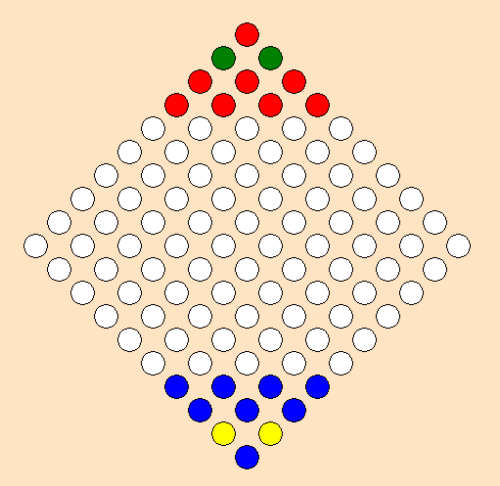

Figure 1: Checkers Board

2 Checkers Rules

2.1 Game Setup

- The game is played on a checkers board, as shown in Fig. 1.
- **Player 1** controls 8 blue pieces and 2 special yellow pieces.
- **Player 2** controls 8 red pieces and 2 special green pieces.
- **Player 1** goes first, followed by alternating turns between the players.

- The objective is for each player to move all their pieces into the opponent's starting area:
  - Player 1 must move the yellow pieces to the green area and the blue pieces to the red area.
  - Player 2 must move the green pieces to the yellow area and the red pieces to the blue area.

2.2 Movement Rules

Each turn, a player can move one piece. Pieces can move in two ways, following standard Chinese Checkers rules:

- **Move to an adjacent empty space**: A piece can move to any of the directly adjacent positions if they are unoccupied.
- **Jump over a piece**: A piece can jump over an adjacent piece (belonging to either player) if the space directly on the opposite side, along the same line, is empty. Multiple jumps are allowed in a single move if conditions allow.

2.3 Special Rules for Extra Turns

- When a player successfully moves one of their special pieces (yellow for Player 1, green for Player 2) into the corresponding target area on the opponent's side for the first time, they are awarded an extra turn.
- During this extra turn, the player may move one of the opponent's pieces. The movement of the opponent's piece follows the same rules of adjacency or jumping.
- Each player can earn up to **two extra turns** throughout the game (one for each of their special pieces reaching the target area).

2.4 Winning Condition

- The game ends when one player successfully moves all of their pieces into the opponent's starting area, with both special pieces occupying their corresponding target zones. The player who achieves this wins the game.
- The maximum number of rounds is **200** (**excluding additional rounds**). If the maximum number of rounds is reached, the player with more pieces in the correct zones wins the game.

3 Code Description

We have provided the code for the checkers environment and implemented two agents: the `RandomAgent` and `SimpleGreedyAgent` classes, which represent random and greedy strategies, respectively. These agents are implemented in the `agent.py` file. We have tested the code in **Python==3.8**. You are required to complete the `YourAgent` class in the `agent.py` file to execute the minimax algorithm with Alpha-Beta pruning. You will run `runGame.py` to execute a match between the two agents (as defined in the callback function). Complete the task according to the following requirements:

1. Implement the `YourAgent` class to apply the minimax algorithm with Alpha-Beta pruning.
2. Test your agent as the first player (Player 1) and as the second player (Player 2) against the `SimpleGreedyAgent`.
3. **(Optional)** Let two `YourAgent` instances compete against each other.
4. Document the results of these matches in your report.

4 Grading Criteria

The grading criteria for this project are as follows:

- Code: 70%, which must include basic comments, and only `agent.py` is allowed to be modified.
- Report: 20%, which should include the algorithm implementation process and necessary test results, limited to a maximum of **eight pages**.
- Performance: 10%, where performance evaluation will be determined through a match between YourAgent and an undisclosed Baseline.

5 Discussion and Question

You are encouraged to discuss your ideas, ask and answer questions about this homework. If you encounter any difficulty with the assignment, try to post your problem **on Canvas** for help. The classmates and the course staff will try to reply.

6 Submission Instructions

1. Complete the code and write report in **English**.
2. Zip all your files to a file named as **HW2_ID_name.zip**. The file structure should be as follows:

```
HW2_ID_name.zip
   code/
      agent.py
      other_files
   report.pdf
```

3. Upload the file to the homework 2 page on the Canvas.
4. **Deadline**: 2024.10.19 23:59