Project 1 Reflection

W200 - Fall 2020

by Young Ha Kim

Chess Game

(a) Introduction: A chess program where you can play against another human, a computer AI, or put the computer against itself.

(b) How to Run/Test: After cloning the repository, simply run 'python main.py'. The rest of the instructions are presented at the start of the program. I have tested on Windows 10 and MacOS.

(c) Completed Features:

- Chess Game: logic for all the legal moves and rules in chess
- Board Representation: Colored chess board using unicode symbols and displaying chess parameters and move history
- Human Input: User input with verification for playing 1:1 chess
- Random AI: Makes a random move unless a capture can be done, in which case a random capture is done. Mostly for testing purposes and for beginners who want to try out different moves to learn the rules of the game.
- Minimax AI with $\alpha\beta$ pruning, with search depth 2 running in under 1 second and search depth 3 running in under 30 seconds on average.
- Move History: Move history can be saved to file at the end of the game.

(d) Unfinished/Todo: There were alot of things I wanted to do, but simply could not given the time constraints. Some of the items I initially set out to do are as follows:

- Current minimax algorithm with alpha-beta pruning takes about 1 second to explore 2000 states. If time allows, I would refactor the move searching methods to have faster time complexity and speed up the code, with the goal of a 10x speedup or more, so that the algorithm can explore 4 or 5 depth within 1 minute per move.
- Addition of opening books and endgame books to enable better endgame playing.
- Output to ISO FEN notation and integration with a chess engine such as Arena using PyPi, so that I can compete with existing chess engines to further evaluate how far my simple python based AI can compete.
- Move history saving with more details such as the time taken for each move in a more versatile format such as JSON.

(e) Challenges:

- Lines of Code: The hardest part was trying to reduce the number of lines of code. After the first quick and dirty implementation of the core chess moves, chess boards, random AIs and user interfaces, the project was already over 1000 lines long. I spent over 80% of my time trying to figure out which feature to remove, which methods to prune, and how to refactor the code to get as close to 750 lines as possible.
- Code optimization: After the first initial coding, the minimax algorithm took quite a long time to run for just 3 levels of depth. To resolve this, I used profiling tools to identify bottlenecks. I discovered that most of the time was spent making a deepcopy of the Board class, so I removed the board class and instead put everything into a single game state tuple and did the copying via list comprehensions. This resulted in a 40% reduction in running time.