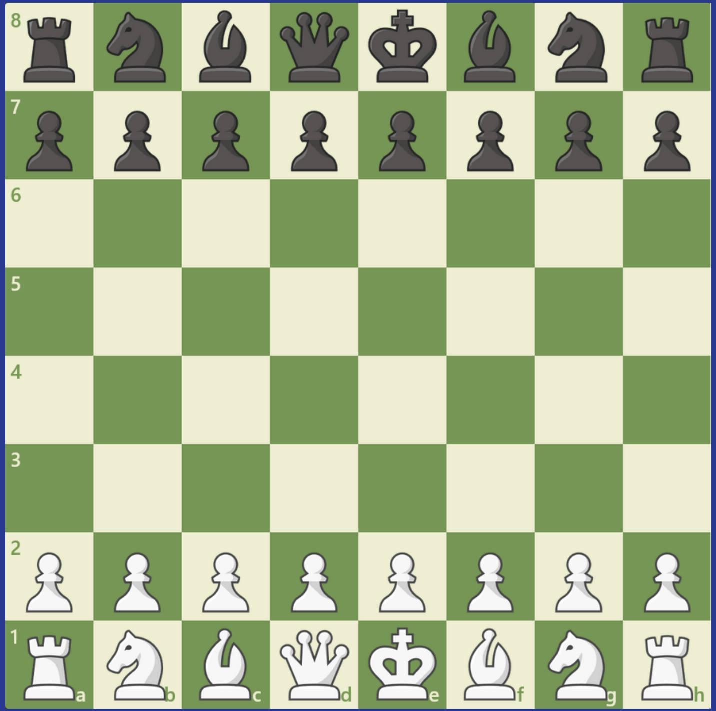
Chess!

Lukas, Spencer



Algorithm, Tools, and Notation

Algorithm

- Minimax with alpha-beta pruning.
- Depth limit is required

Other considered algorithm was Monte Carlo Search

Tools

"chesslib" library (maven dependency)

- Create board objects
- Apply moves
- Many built in helper methods

Notation

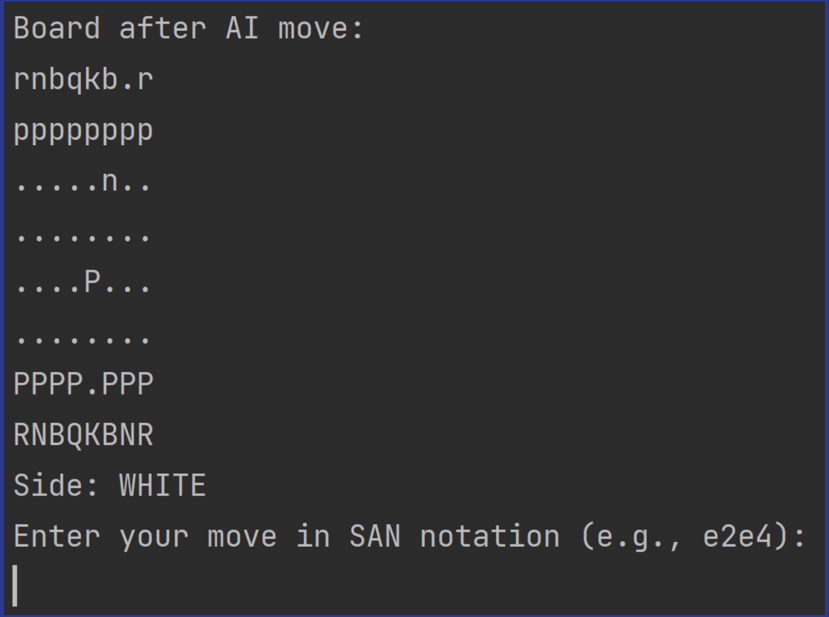
User input is in SAN notation

Square to square coordinates

Ex: a3b4, move from square a3 to b4

It works!

Minimax with alpha-beta pruning greatly reduces computation time!



```
Board after AI move:
rnbqkb.r
pppppppp
.....n..
........
....P...
........
PPPP.PPP
RNBQKBNR
Side: WHITE
Enter your move in SAN notation (e.g., e2e4):
```

Problems (a few of them)

Chess is hard

Chess is not a solved game.

Our AI is not very good at chess.

Evaluating Moves

Take into account :

- Piece values
- Piece coordination
- Centralization
- Offensive or defensive moves
- The sheer amount of moves
- Many more

Making the choice

Many moves have the same or similar evaluation but result in massively different boards.

Main Heuristics

Piece Values

- PAWN: 100
- KNIGHT: 320
- BISHOP: 330
- ROOK:500
- QUEEN: 900
- KING: 10000

Used in many places, helps keep material on the board.

Prioritize Center

The center of the board is a much stronger position than the outsides

Inner 4 square have large bonus value, next closest 12 squares have small bonus value

Captures

Using piece values, have a heavy bonus value depending on the piece captured.

Quick Overview

- Rewrote Minimax for chess application
- Applied and tested many heuristics
- A lot of bug fixing