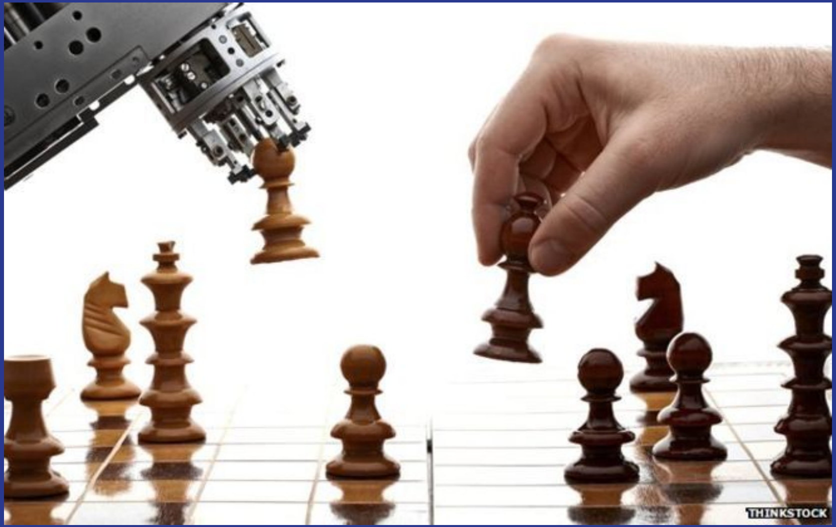
Chess AI

Alex Godziela, Henry Kwan, Elias Strizower



Our Project

- Create an AI that can win a chess game against a human player
- Two main components:
  - Chessnut
    - Chessboard model
    - Handles chess moves and mechanics of game
  - Chess AI
    - Heuristics
    - Minimax Algorithm
    - Using minimax to look 2-3 ply deep

Chessnut

- Simple chess model written in Python
- Not a chess engine, has no AI
- Can import/export games in Forsyth-Edwards Notation (FEN)
- Generates a list of legal moves for the current board position
- Intelligently validates and applies moves (including en passant, castling, etc.)
- Keeps track of the game with a history of moves

Iteration 1 - AI makes random moves

- Player can input moves
- AI will randomly select from a list of valid moves
- Really easy opponent to beat...

What makes a chess AI?

- Search - look ahead at different move sequences (because a perfect evaluation of the board is impossible)
- Evaluation - what position on a board is the “best”
- A good chess AI will use these two components in tandem
- Iteration 1 did none of this

Searching, Ply, and Depth

- Ply/Depth is a full move (ie. white, then black in order)
- The deeper the ply/depth, the more sophisticated the AI's search, and the "smarter" the AI will be

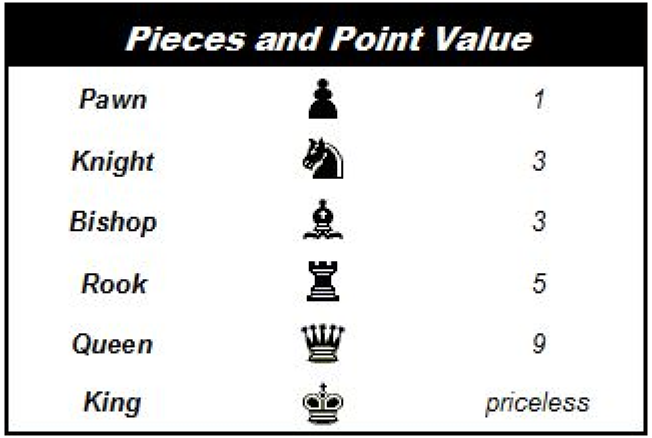
Chess AI's are like toilet paper,
The more ply the better

Evaluation and Heuristics/Logic

- Many different heuristics go into creating an evaluation function
- Simple: value of piece, number of moves available, etc.
- Complex: when to sacrifice a piece, when to take a piece vs. setting up for check
- A masterful chess AI would take into account many heuristics

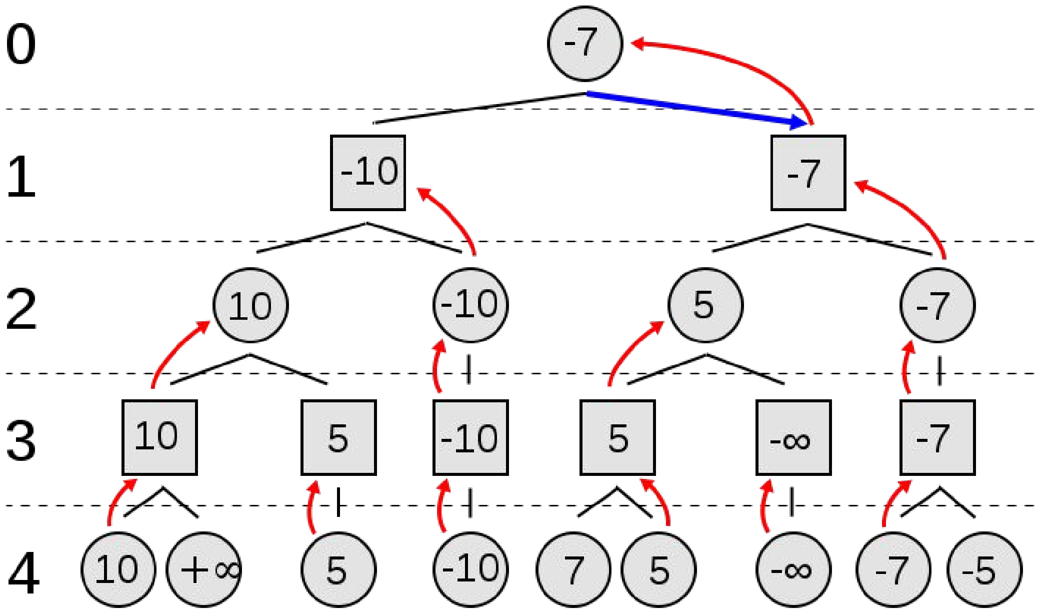
Iteration 2 - Basic Searching and Evaluation

- Search
- AI looks at next best move
- Evaluation
- Taking a piece gives a certain point value (see image)
- Uses highest point value to determine what is the “best” move



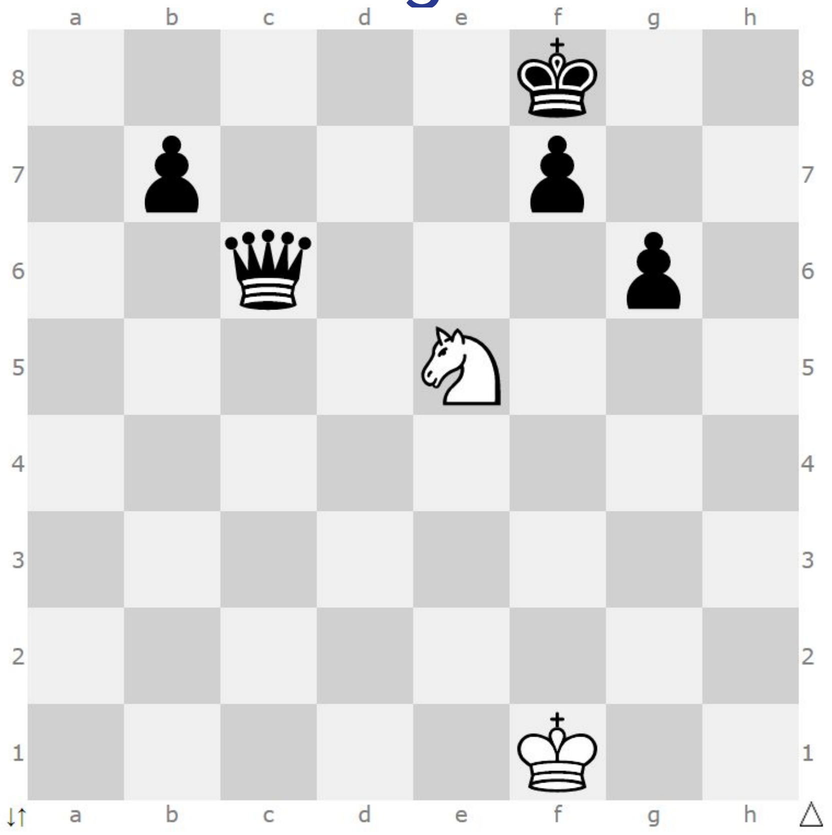
Pieces and Point Value		
Pawn	♟	*1*
Knight	♞	*3*
Bishop	♝	*3*
Rook	♜	*5*
Queen	♛	*9*
King	♚	*priceless*

Advanced Searching - The Minimax Algorithm



- Decision rule to minimize possible loss for a worst-case scenario
- For example, score of each move for two-ply search is the score of the worst that the opponent can do

Advanced Searching - The Minimax Algorithm



2-Ply Net Score: +6

Iteration 3 - Minimax Implementation

- An intelligent chess AI which uses minimax algorithm to choose its next move
- Uses combination of best AI move and worst human move to choose best possible move for itself
- Conceptually difficult
- Chessnut makes it challenging to implement

Advanced Evaluation - Additional Heuristics

- Combining simple heuristics makes a more sophisticated overall heuristic
- Handle a higher value piece taking a lower piece but putting itself into danger
- Advanced chess strategies and tactics

Future Work

- Searching can be improved with Alpha-Beta Pruning, Scouting, and Mate Searching for example
- Evaluation can have more complicated heuristics

Demo

Reflections

Good:

- ChessNut was a great library that helped kickstart our project
- The simple GUI is intuitive and helped with development

Bad:

- ChessNut makes it complicated to look deeper than 1 ply
- Minimax was tough to implement
- Bit off more than we can chew (Chess is mathematically complex and minimax is hard to implement)

Thanks for listening!

AI algorithms: https://github.com/lamesjim/Chess-AI
Chess board: https://github.com/cgearhart/Chessnut
Chess Programming Wiki: https://chessprogramming.wikispaces.com/Programming
Minimax Algorithm: https://en.wikipedia.org/wiki/Minimax
Stockfish API for python: https://pypi.python.org/pypi/stockfish
Niklasf engine api: https://github.com/niklasf/python-chess
Sunfish engine api: https://github.com/thomasahle/sunfish