King + Rook vs King Endgames

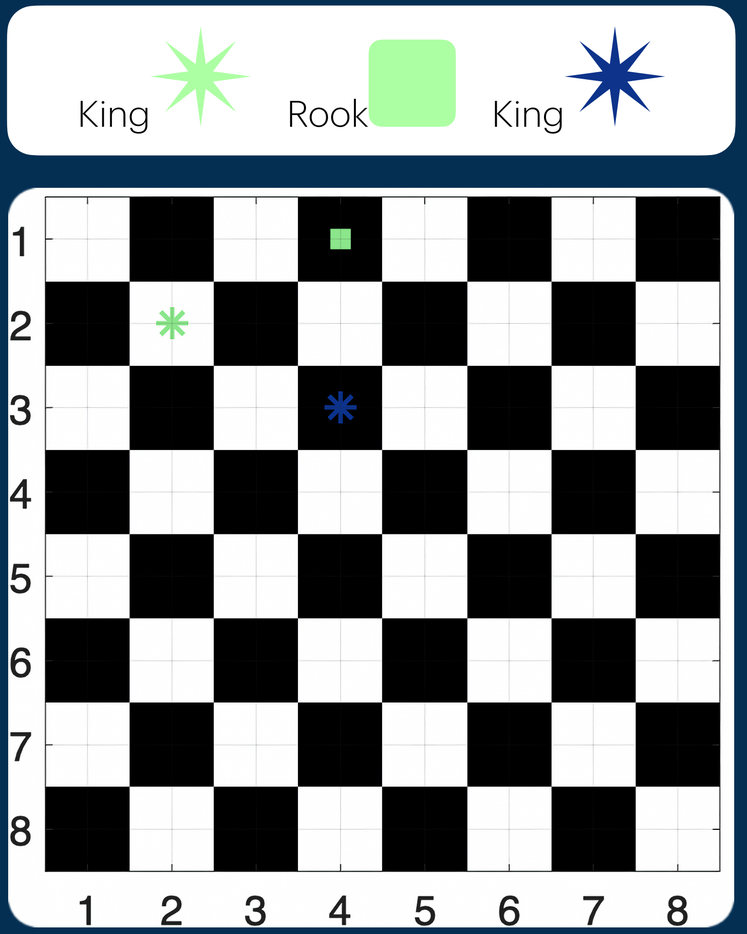
A **min – max** solver

Tien Nguyen, Melissa Osheroff, Raaghav Thirumaligai

Motivation

Why is this interesting?

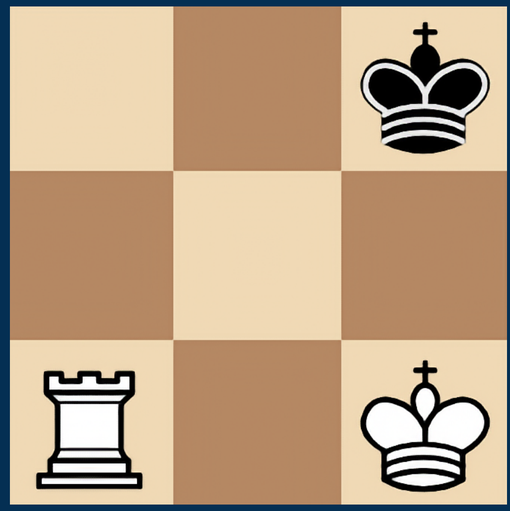
- There are approximately 10^{120} different possible games of chess
- Even only considering a KRK endgame, there are ~250,000 different unique configurations of the board and checkmate is nontrivial
- Can we build a software implementation of chess that utilizes game theory to find a faster solution than playing at random?



King+Rook vs King End Game

The situation

- Two player alternate zero-sum game
  - P1 White/Green (minimizer)
    - King + rook
    - Wins the game by forcing checkmate on P2
  - P2 Black/Blue (maximizer)
    - Lone king
    - Wins by capturing the rook and prolongs being put in checkmate



State Representation

Every chess position is encoded as a 65-element vector

- Elements 1-64: the vectorized 8×8 chessboard
  - +10 (white king)
  - +5 (white rook)
  - -10 (black king)
  - 0 (empty)
- Element 65: whose turn it is (+1 = white, -1 = black)

$$x = \begin{bmatrix} 0 \\ \vdots \\ 5 \\ \vdots \\ -10 \\ \vdots \\ 10 \\ \vdots \\ \pm 1 \end{bmatrix}$$

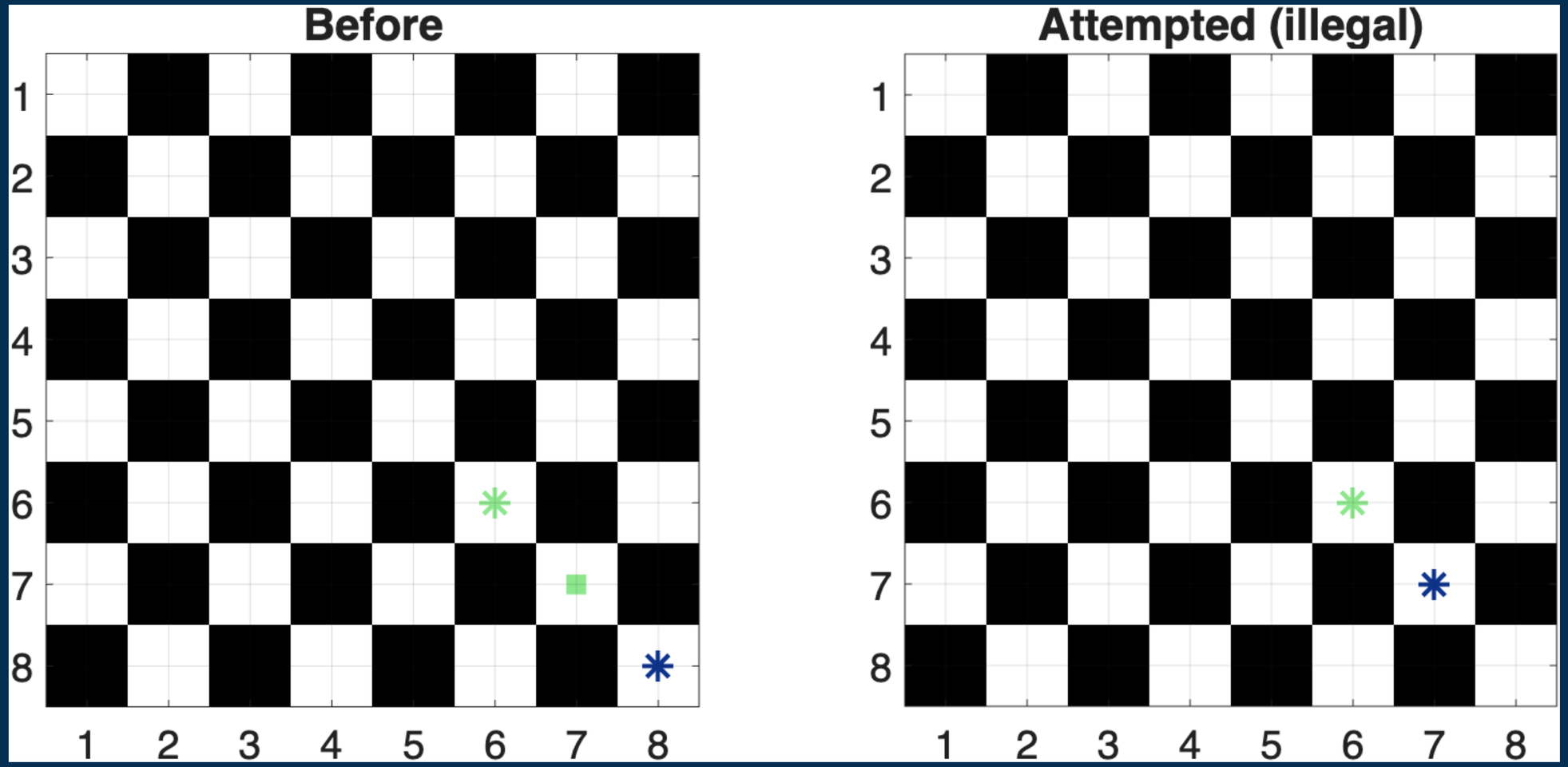
Action Spaces & Legal Moves

What moves can we make?

- P1 action space
  - $\Gamma_1 \in$ {move king to any adjacent space} $\times$ {move rook vertically or horizontally}
- P2 action space
  - $\Gamma_2 \in$ move king to any adjacent space
- Illegal moves
  - P1 and P2 cannot put themselves into check
  - P1 and P2 pieces cannot leave the board, unless captured

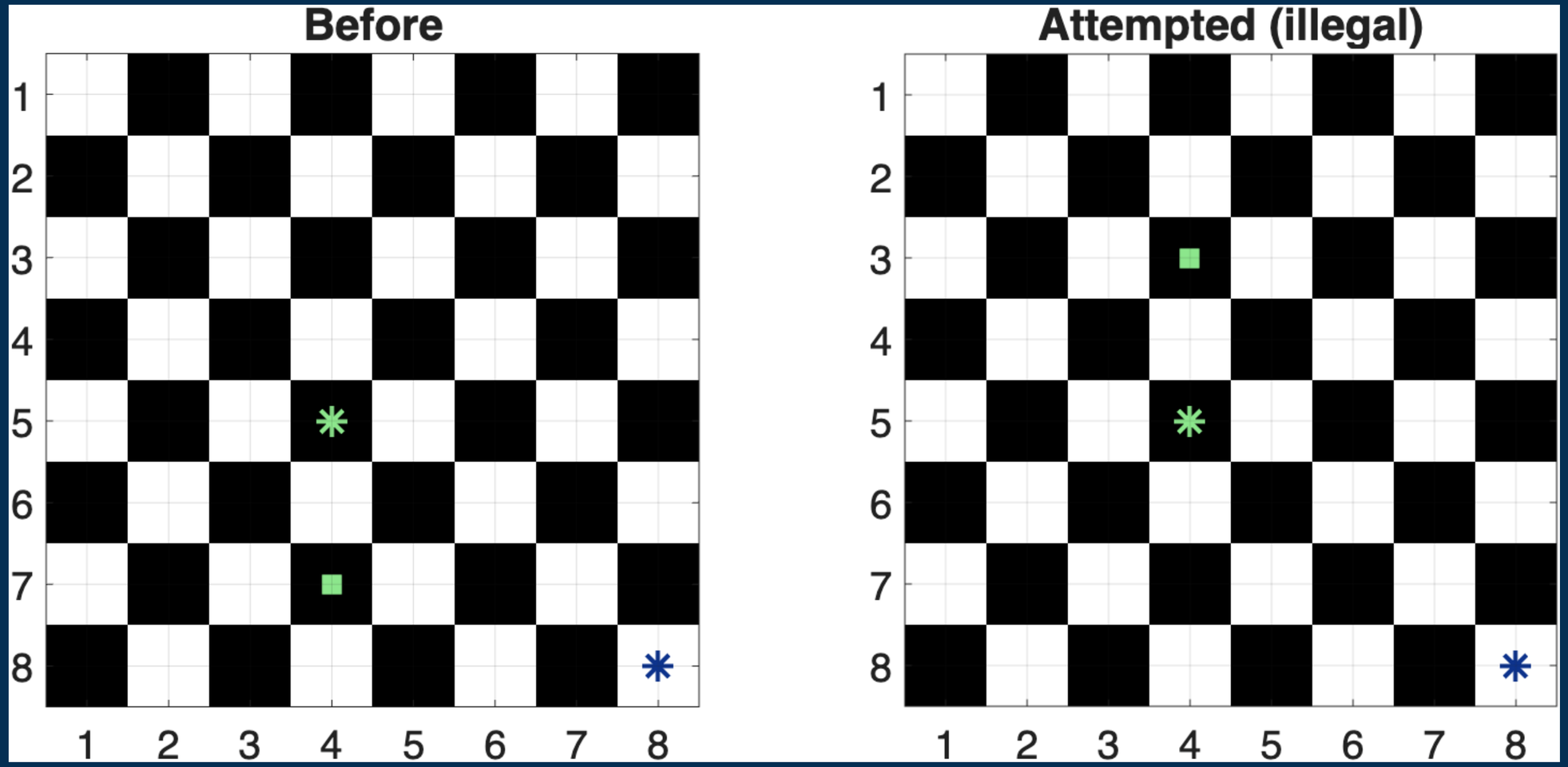
Illegal Moves

Green Rook is protected by Green King



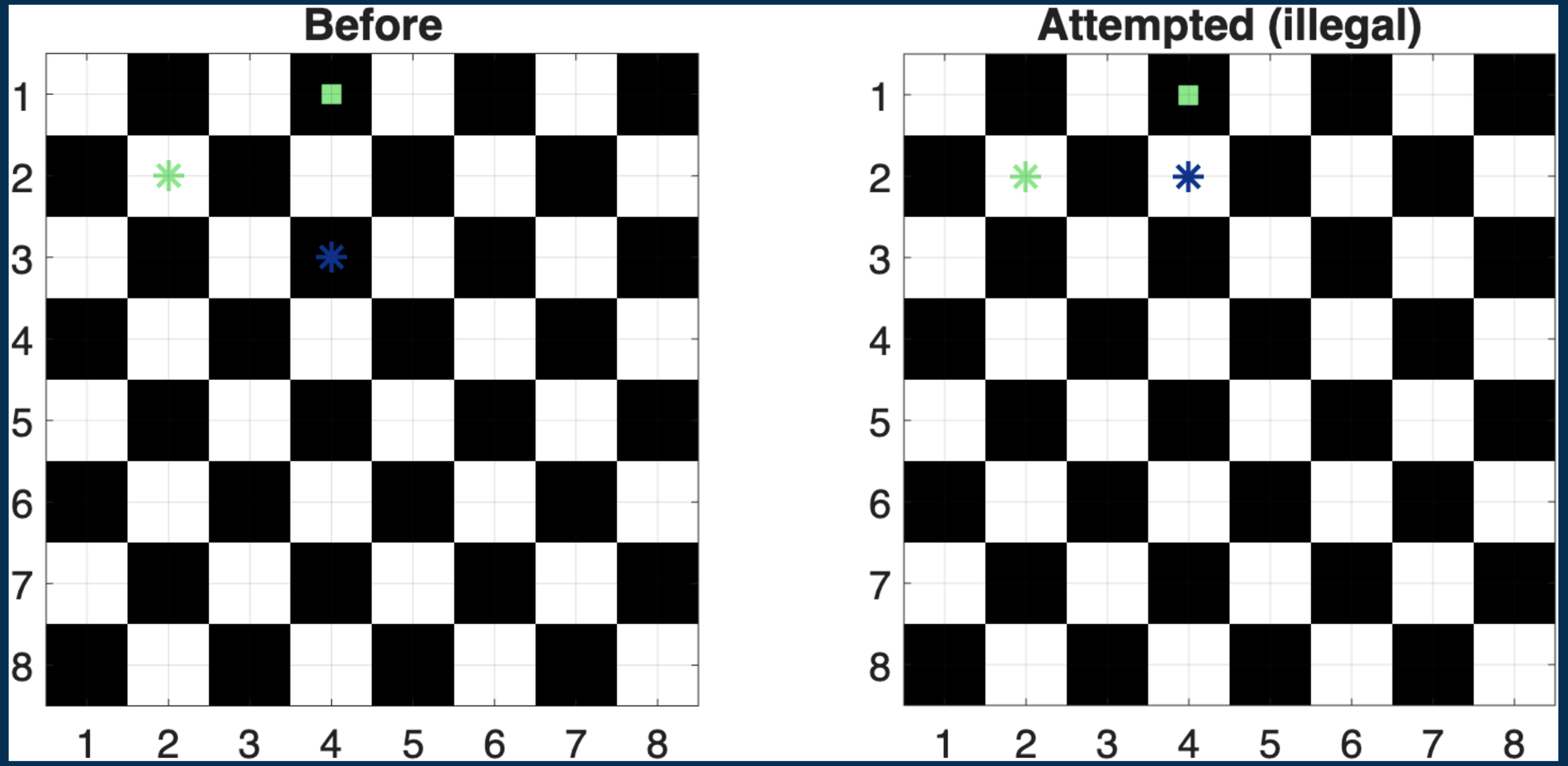
Illegal Moves

Rook cannot travel through a piece



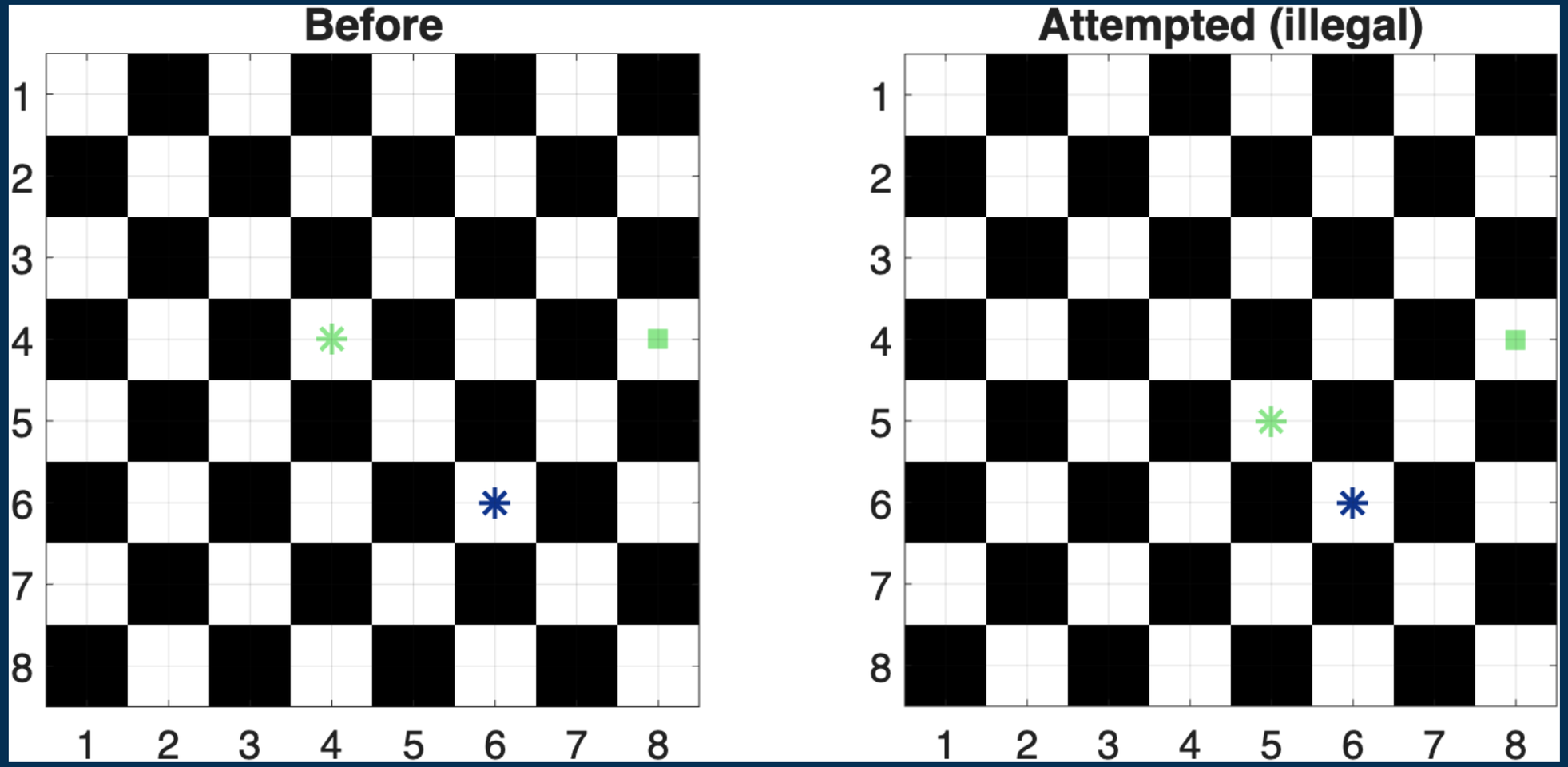
Illegal Moves

Blue King cannot move into check



Illegal Moves

Kings cannot move next to each other



The art of the cost function

We design a heuristic cost function to drive the minimax optimization

$$J = \text{inJail} + \text{oxygenP2} + \text{distKings} + i_{\text{mate}} + i_{\text{stale}}$$

$$\text{inJail}(\text{oxygenP2}(x)) = \begin{cases} 100 & \text{if oxygenP2}(x) = 0 \\ 0 & \text{if oxygenP2}(x) \neq 0 \end{cases}$$

$$\text{oxygenP2}(x) = \text{number of free spaces that P2 can reach with P1 frozen}$$

$$\text{distKings}(x) = \text{chebychev distance between the kings}$$

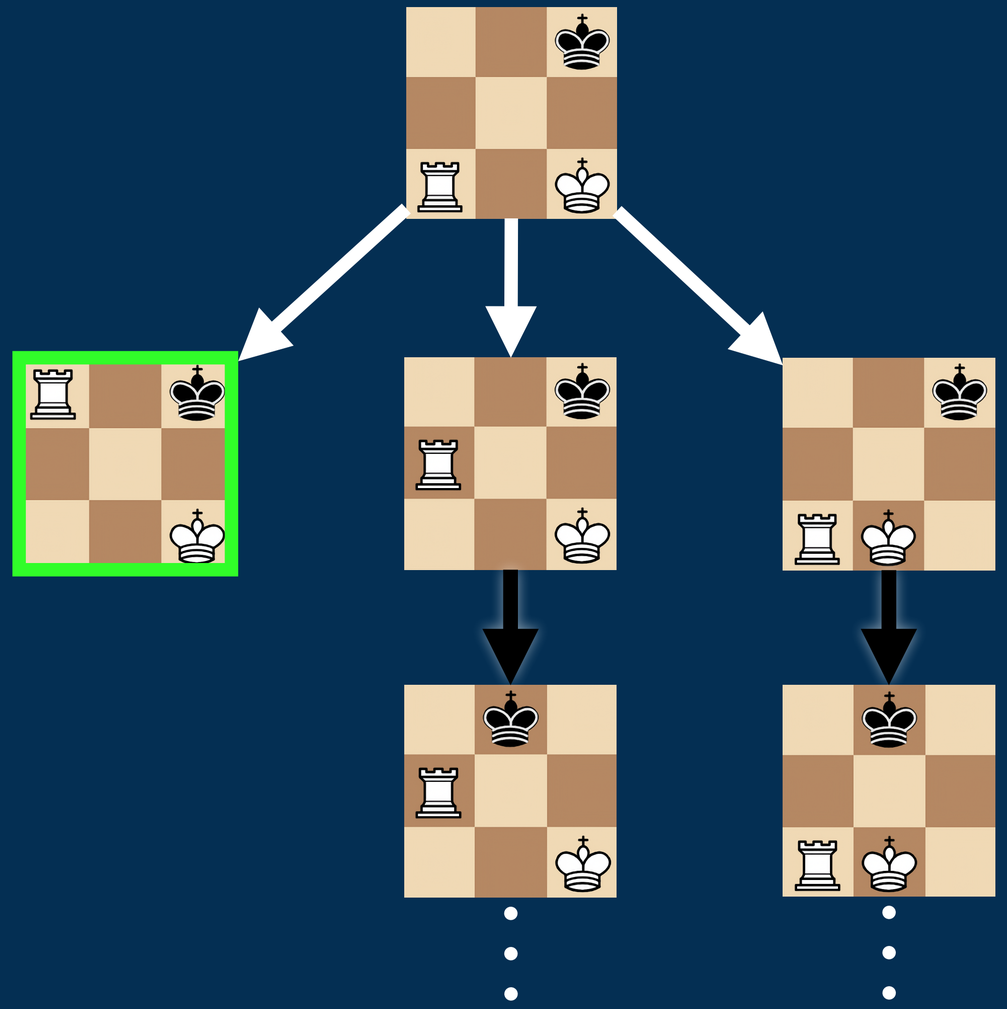
$$i_{mate}(x) = \begin{cases} -\infty & \text{if checkmate}(x) = 1 \\ 0 \end{cases}$$

$$i_{stale}(x) = \begin{cases} +\infty & \text{if stalemate}(x) = 1 \\ 0 \end{cases}$$

Minimax Algorithm

A decision-making algorithm for finite two-player zero-sum games

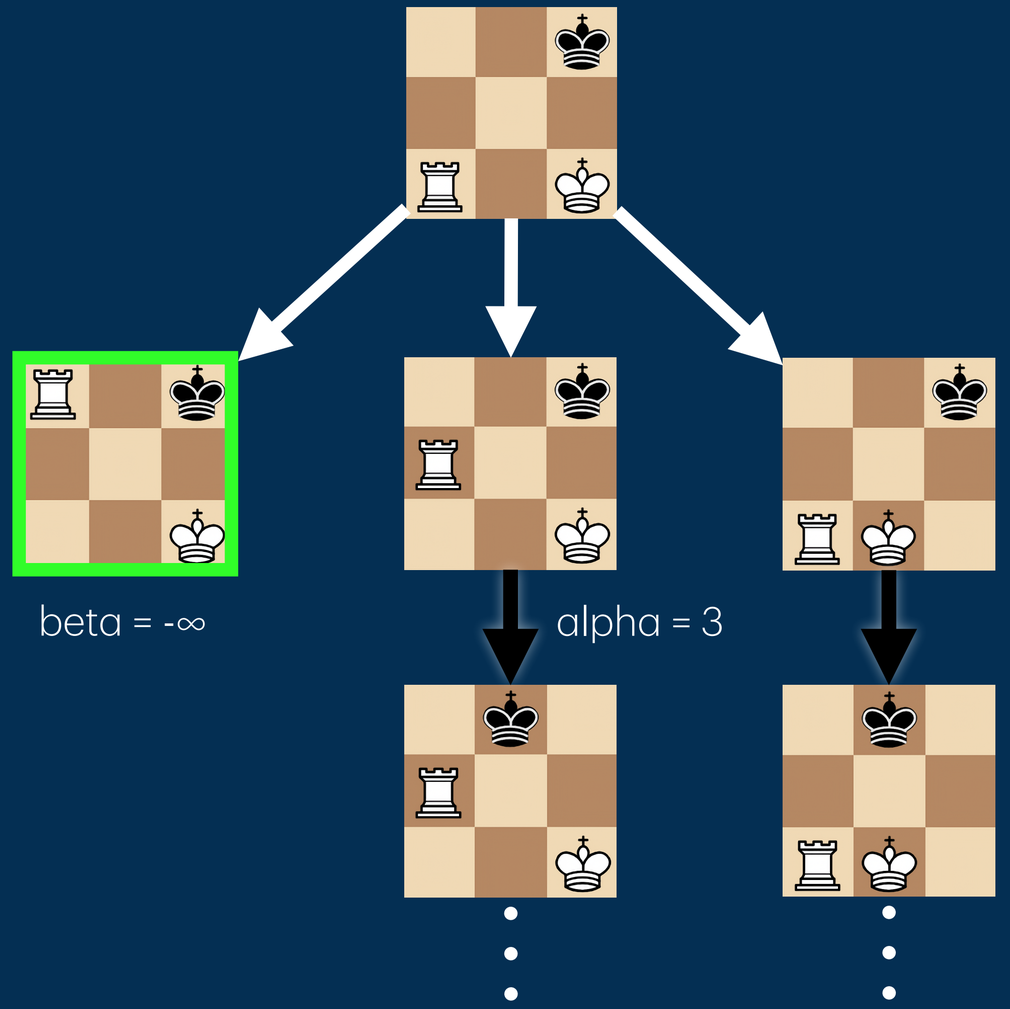
- White is the minimizer, black is the maximizer
- Assumes optimal play from both sides: opponent picks worst move, to fight objective
- Searches the game tree to a fixed depth
- Then backtracks and returns the move with the best valued cost function



Alpha-Beta Pruning

A search-time improvement to the minimax algorithm

- Alpha: the best score that the maximizer (black) can guarantee so far
- Beta: the best score that the minimizer (white) can guarantee so far
- Both start at their worst possible values: alpha = −∞, beta = +∞
- When beta ≤ alpha: current branch cannot be selected by a rational opponent, so prune it
- Reduces nodes evaluated without changing result



Further Optimizing

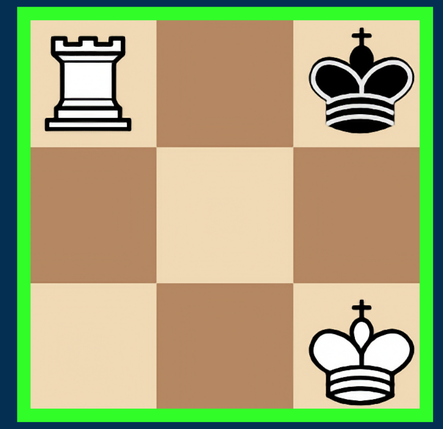
Recursive Tree Evaluation with Memoization

- Next move is chosen by setting up a multi-stage alternative game with n number rounds
- The values of the leaves are determined by the cost function given the state determined by the last turn
- Memoization - returns cached result for previously computed states
- Checkmate - once found, no other nodes are searched

Terminal Cases & Boundary Conditions

Ending the game

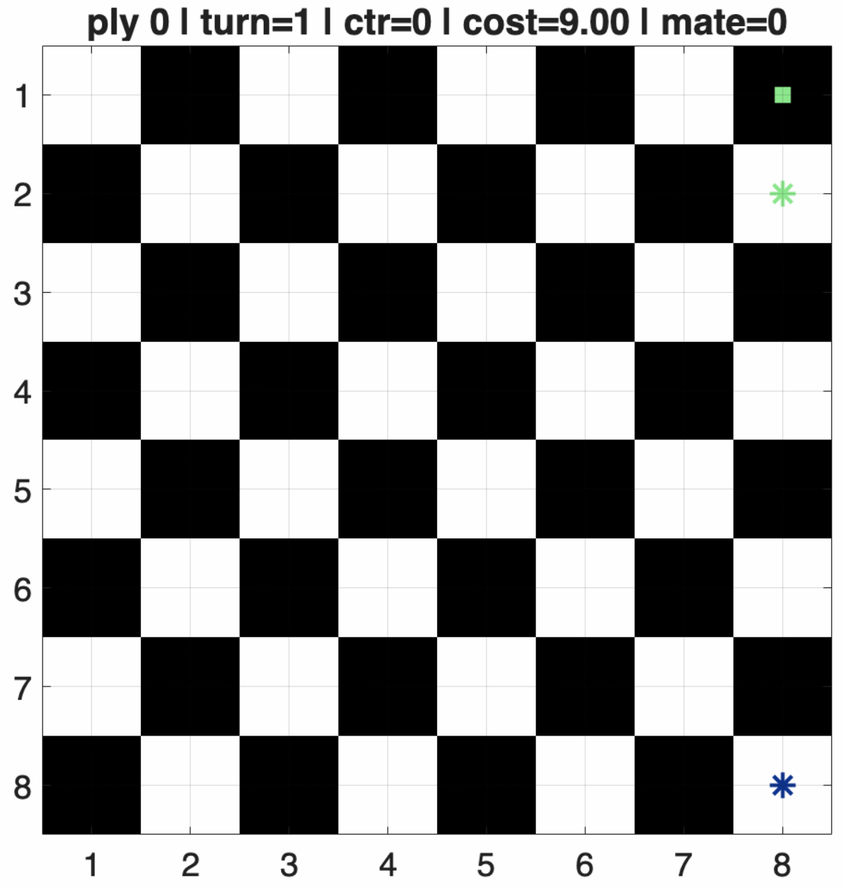
- Checkmate: black king is in check AND has zero legal moves
- Stalemate: (black king is not in check AND has zero legal moves) OR (rook captured)
- Length of state vector is fixed so pieces cannot accidentally leave the board
- Game starting in checkmate or stalemate are solved trivially



Examples

Interesting Behaviors

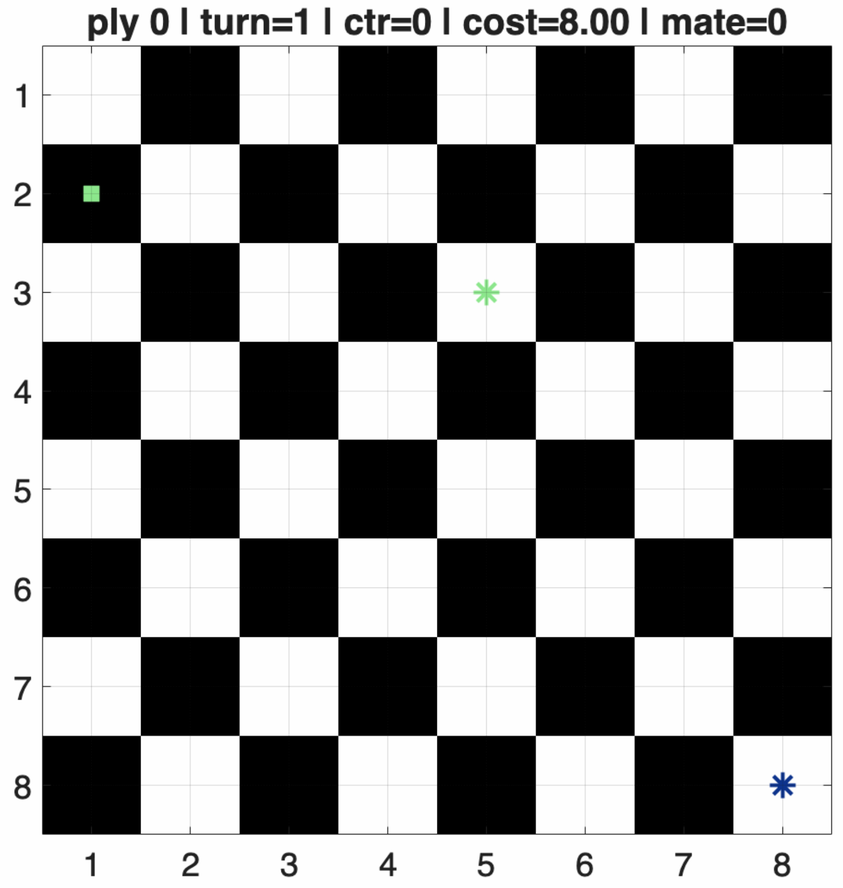
- Green King frees the Blue King to achieve checkmate



Examples

Interesting Behaviors

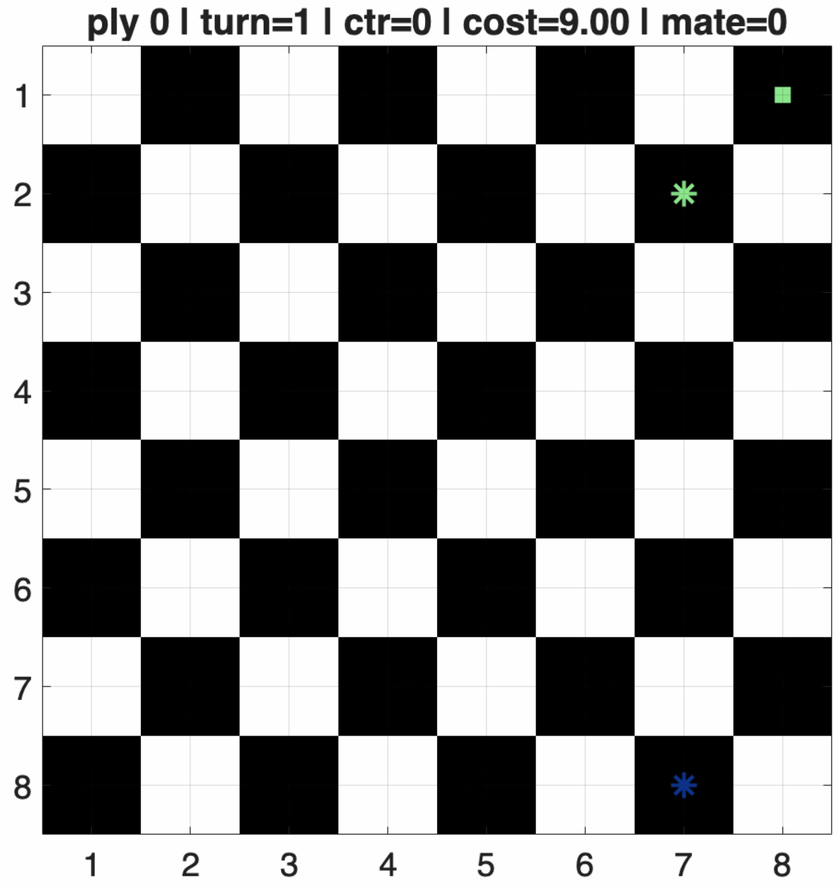
- Rook moves to trap then waits for Green King to close in



Examples

Interesting Behaviors

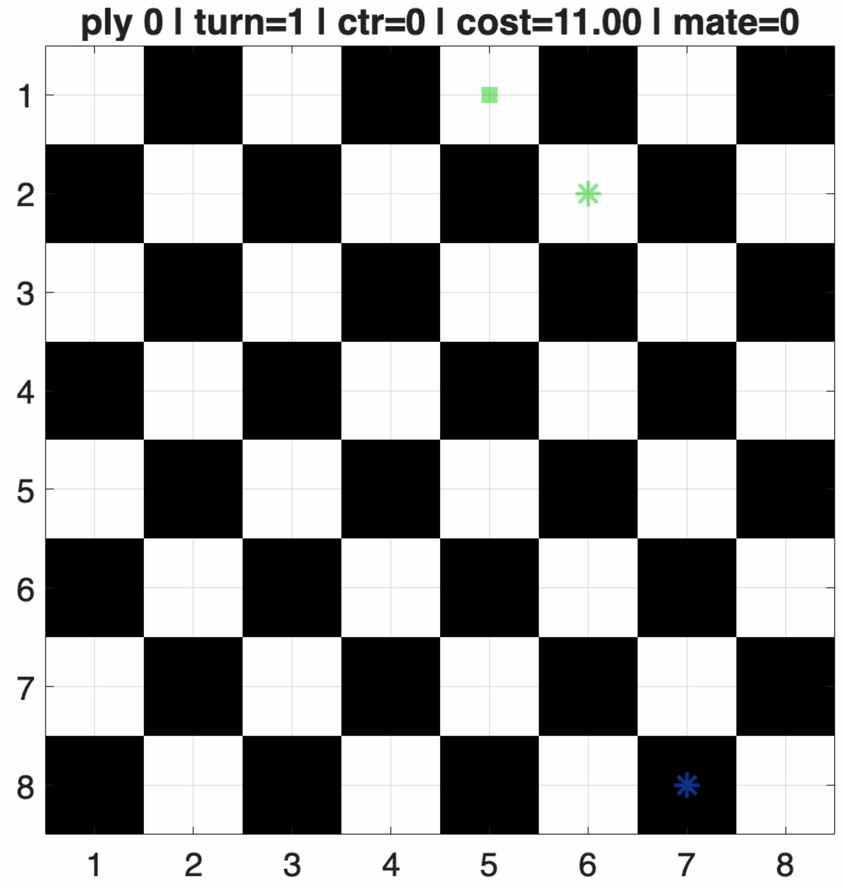
- Mate found without first blocking the Blue King against an edge with the rook



Examples

Interesting Behaviors

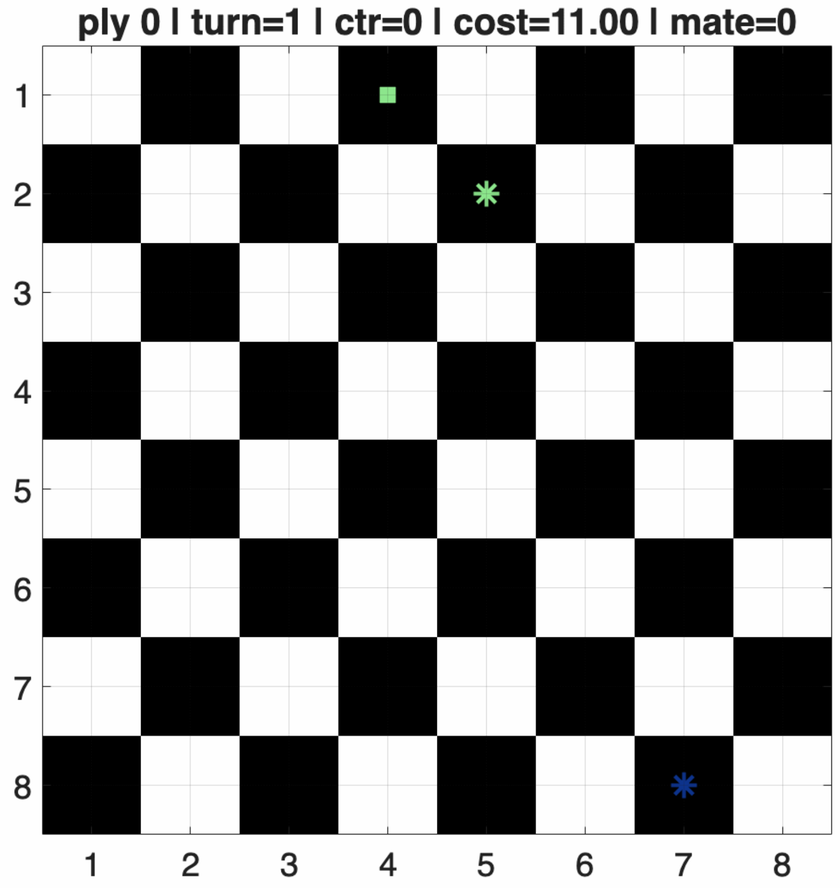
- Blue King chases unprotected rook



Examples

Interesting Behaviors

- Blue King stops chasing rook to delay mate
- Green Rook backs away to force mate in future move



Future Work

- Play game against better chess solvers like stockfish
- Incorporating more pieces
- Implementing dynamic program
- Implementing Monte Carlo Search Tree
- Testing and optimizing cost function (even more!)
- Optimizing runtime and memory

Thank You!

Questions?