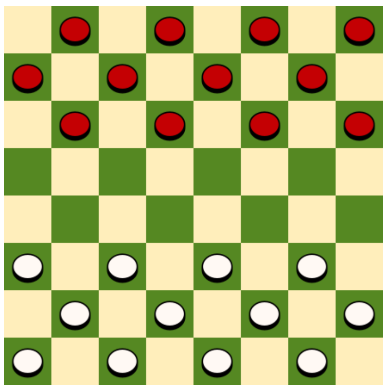
Project Proposal: CheckersBot

Kavika Krishnan - kk3526@columbia.edu
Catarina Coelho - mdc2234@barnard.edu

Overview

Context:
The game of checkers has been around for centuries and involves an nxn board with circular pieces. In our setup, we will use an 8x8 checkerboard. Each player starts with 12 pieces, either "pawns" or "kings," arranged on the three rows closest to them in the "straight checkers" starting position seen below. The goal is to either capture all of the opponent's pieces or block their moves. Pawns move diagonally forward, capturing opponents by jumping over them. When a pawn reaches the opposite end of the board, it becomes a "king." Kings gain the ability to move diagonally backward, enhancing their capturing potential. Opponent pieces are captured by jumping over them diagonally, and multiple captures can be made in a single turn, especially with kings.The game concludes when a player cannot make a move, which is done by capturing all opponent pieces or strategically limiting their moves.



Goal:
Our project, "CheckersBot", will involved creating a bot capable of playing and defeating a user at the classic game of checkers. Using parallelization in Haskell, our primary goal is to implement and compare two different algorithms for the bot, providing insights into their performance and efficiency.

Algorithm 1: Minimax with $\alpha - \beta$ pruning

We will first use the classic Minimax method to identify the best move for the checkers-playing bot. The method works by systematically traversing a N-ary game tree, which represents potential actions and counter-moves. The algorithm alternates between maximizing and decreasing the heuristic values associated with alternative outcomes at each level of the tree. The number of movements forward that the algorithm considers is determined by the depth of the tree, which influences the quality of decision-making. To optimize the search process, we will use Alpha-Beta Pruning. Alpha-Beta Pruning is a technique that removes branches from the game tree that have no bearing on the ultimate choice, hence minimizing the search space. By intelligently rejecting "useless" pathways, the method significantly improves computational efficiency without compromising decision quality.

Algorithm 2: Monte Carlo Tree Search

We will employ Monte Carlo Tree Search (MCTS) in our second algorithm. This method relies on heuristics and is centered on statistical sampling and random simulations. During decision-making, MCTS dynamically constructs a tree structure as opposed to expressly using a N-ary game tree. Four iterative steps are carried out by the algorithm: simulation, backpropagation, expansion, and selection. MCTS analyzes the worth of various moves by using random simulations from a given game state, facilitating more exploratory and adaptive decision-making.

How our bot will operate

- First the user will select an algorithm from the aforementioned two
- The initial game board will be printed, with the starting player (i.e whether the user or the boat is red/black) randomly chosen. It will look something like the following:

```
  a b c d e f g h
1 - r - r - r - r
2 r - r - r - r -
3 - r - r - r - r
4 - - - - - - - -
5 - - - - - - - -
6 b - b - b - b -
7 - b - b - b - b
8 b - b - b - b -
```

- The user/bot expresses their move in a form like "a6 to b5"
- The new board is printed, with any captured pieces omitted or kings denoted as "bk", "rk"
- Game continues until either the user or bot cannot make a move/

How we will parallelize

We will use parallelization to concurrently explore of potential moves within the Minimax and MCTS algorithms. Specifically, we can use parMap and rdeepseq to get the optimal move for the chosen algoirthm.

References

https://doi.org/10.29407/intensif.v5i2.15863
https://mendel-journal.org/index.php/mendel/article/view/163
https://github.com/alsoltani/Checkers
https://github.com/dimitrijekaranfilovic/checkers

Images

https://en.wikipedia.org/wiki/File:Draughts.svg
https://raw.githubusercontent.com/AboorvaDevarajan/Parallel-Checkers-Game/master/images/1.png