Rollerball

(COL333 Assignment 2)

COL333 TAs

September 2023

1 Goal

The goal of this assignment is to implement the adversarial search algorithms learnt in class, such as minimax and alpha-beta pruning, in a real-game setting.

2 The Game of Rollerball

2.1 Starting Position

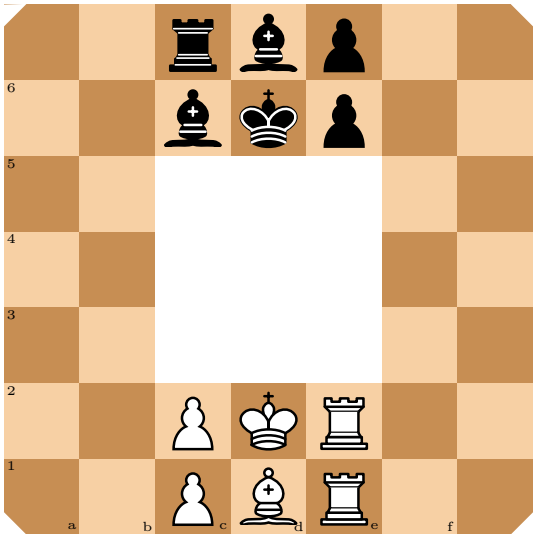

Figure 1: Starting position for 7x7 rollerball

Rollerball features a 7x7 board with a 3x3 section in the middle cut out. There are six pieces per side: two pawns, two rooks, one bishop and one king. The pieces move similarily to chess, with two changes:

- The pieces can move n steps in the clockwise direction and one or zero steps in the anticlockwise direction
- The pieces may reflect off the four corners (for rooks) or off the edges (for bishops) atmost once

The game proceeds similar to chess, with the objective being to checkmate the opponent king. Further details are described below.

2.2 Movement Rules

Due to four-fold symmetry of the board, we shall only describe the rules over these 10 squares of the chessboard. All other moves can be obtained similarly. The pieces moves clockwise most of the time.

The squares on the board form two rings - the outer ring and the inner ring. Details for each piece movements are given below.

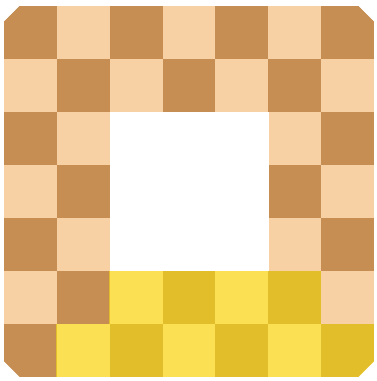

Figure 2: Four-fold symmetry of the board, and squares over which moves are described

2.2.1 King

The king moves and can capture one step in any direction. Similar to traditional chess, the king can not occupy any square where it can be captured. There is no castling in rollerball.

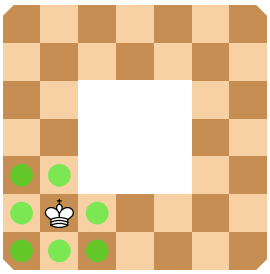

Figure 3: King moves

2.2.2 Rook

Rook moves any number of steps horizontally or vertically along the clockwise direction. It also moves one step in the anti-clockwise direction on its current ring. In rollerball, the rook reflects from the four corners of the board. Only one reflection is allowed in a single move.

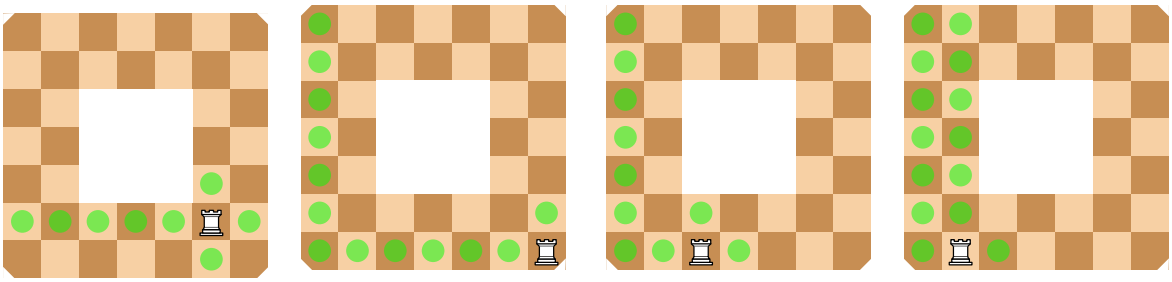

Figure 4: Rook moves

2.2.3 Bishop

The bishop moves any number of steps diagonally in the clockwise direction. It moves only one step diagonally in the anti-clockwise direction. In rollerball, the bishop reflects from the eight sides of the board (four outer sides and four inner sides). Only one reflection is allowed in a single move.

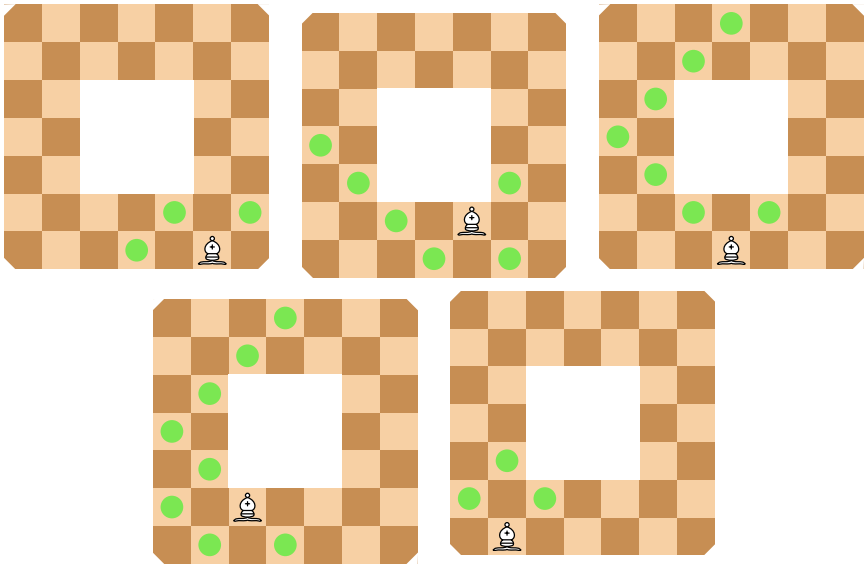

Figure 5: Bishop moves

2.2.4 Pawn

The pawn moves one step straight or diagonally in the clockwise direction. It does not move in the anti-clockwise direction. The pawn can be promoted to a rook or a bishop upon reaching the starting squares of the opponent's pawns.

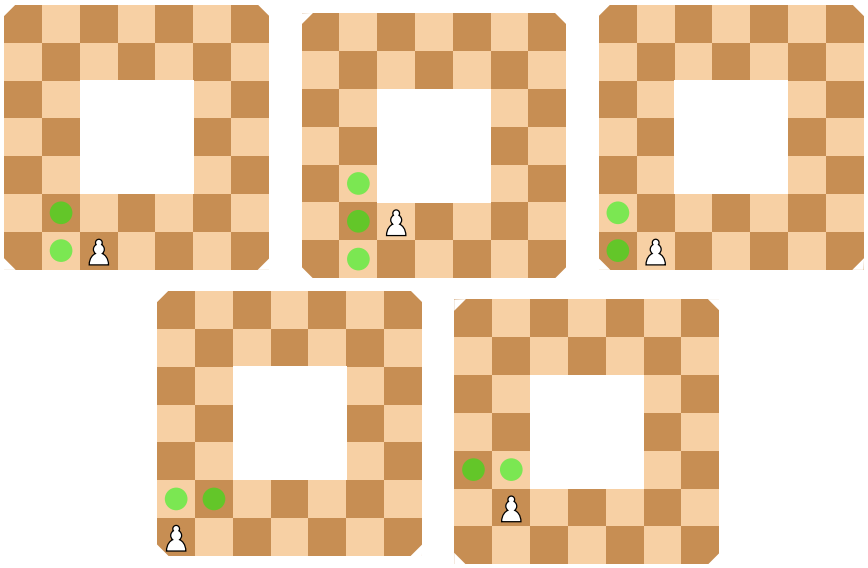

Figure 6: Pawn moves

2.3 Objective

The objective of the game is to capture the opponent's king while keeping the your king safe. Similar to traditional chess, the player who checkmates the opponent is declared the winner.

3 Scoring

The game ends when at least one of the following conditions holds true for the current board state.

1. There is a checkmate. In this case, the King that gets the checkmate loses, and the other player wins (as in classic Chess).

2. Threefold repetition of moves from both the players. Please read this for details of the rule. This condition ends in a draw.
3. Stalemate. If neither of the player is able to play a move on the board, it results in a draw.
4. Both the players have played 100 moves, and none of the above conditions hold for the current board state. This results in a draw.

3.1 Evaluation

Your final submission will be run against 3 TA Bots. Each game will be played twice, once as the white player and once as black. You will be given either one or two seconds per move during evaluation. Your total score is the sum of each of the 6 individual games. The scoring pattern for each game is described below.

3.1.1 Total Score

Total Score = Victory Score + Margin Score

3.1.2 Victory Score

The losing player gets a victory score of 100 - Victory score of the winning player. If the game draws, each player gets a victory score of 40 (this is to discourage drawing of games). Let the number of moves played by the winning player be n. Then, the victory score for either player is computed as follows. ($\mathbb{1}_{[X]}$ is the identity function for outcome X holding for a player.)

$$\textbf{Victory Score} = \mathbb{1}_{[Win]}100 + (\mathbb{1}_{[Lose]} - \mathbb{1}_{[Win]})(5\lfloor\tfrac{n}{20}\rfloor + \min\{10, n\}) + \mathbb{1}_{[Draw]}40$$

You can plot and see how the Victory score varies as a function of the number of moves it takes to win to better understand the scoring metric.

3.1.3 Margin Score

Each piece on the board (except the Kings) is allotted a score as in classic chess. The following table summarises the score value for each piece type. At the end of the game, the sum of the scores for the remaining pieces is calculated for each player. The game starts with a score of 13 each side (5 * 1 bishop + 3 * 2 rooks + 1 * 2 pawns) and can go up to 21 (5 * 3 bishops + 3 * 2 rooks) for each player, considering promotions of the pawns. This is **Margin Score** for the player.

Note that even the losing player gets assigned a Margin score. So if one makes a better judgement of their position at any game stage, they can protect their pieces and let the other player win to attain a Margin score. On the other hand, the winning player can sabotage the losing player by taking as many pieces as they wish.

Piece Type	Score
Bishop	5
Rook	3
Pawn	1

4 Starter Code Overview

The starter code can be found on the course webpage. Rollerball's GUI is implemented in JavaScript, and the core engine is written in C++, with Python bindings provided. Note that *for this assignment, we are only accepting solutions in Python and C++.* Also note that **Due to this assignment being competitive, solutions in C++ would have an advantage over those written in Python**, due to their speed and being able to evaluate a larger number of nodes.

4.1 Requirements

1. gcc $\geq$ 11
2. python 3.9

4.2 Quickstart

```
unzip rollerball.zip && cd rollerball
make rollerball
```

If all goes well, you should have an executable called `rollerball` in `bin`. To run the GUI, launch a web server from the `web` directory.

```
cd web
python3 -m http.server 8080
```

You can then open localhost:8080 on your browser to view the GUI.

To launch the bots (assuming you're in the directory)

```
./bin/rollerball -p 8181
```

You can then connect the GUI to the bots. You would also need to start another bot for black on port 8182 to join and start the game.

4.3 C++ API

The core datatypes used in the code are *positions* and *moves*. A position is an 8-bit unsigned integer, of which only the lower 6 bits are used, where bits 0-2 are used for the x-coordinate and 3-5 for the y-coordinate.

7	6	5	4	3	2	1	0
-		y			x		

Table 1: 8-Bit Position Layout (3 bits for `x` and 3 bits for `y`)

A move is a 16-bit unsigned integer which consists of two positions concatenated with each other. Bits 0-5 are for the final position, bits 8-13 are for the initial position and bits 6 and 7 represent promotions for pawns.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-		y0			x0			r	b	y1			x1		

Table 2: 16-Bit Move Layout (Concatenation of Two 8-Bit Positions)

Macros have been provided to construct and manipulate these types. Other than these two types, board state is represented using BoardData, which contains positions of the twelve pieces on the board and four rotated representations of the board, due to it's four-fold symmetry. Each board is made up of 64 bytes, of which only 40 are used to represent state. Each of these bytes is a combination of PlayerColor and PieceType. See the Board constructor and the two enums for details on how pieces are represented.

The Board class exposes four public methods:

1. `get_legal_moves`: This function computes and returns a set of legal moves for the current player based on the current board state. Legal moves are moves that do not leave the player's king in check. If there are no legal moves, an empty set is returned.
2. `in_check`: This function returns true if the current player is under check from the opponent
3. `copy`: Returns a pointer to a copy of the board. The pointer must be deleted after use.
4. `do_move(U16 move)`: performs a move on the board. Note that this alters the board state accordingly.

4.4 C++ Bindings

You would need to implement the `get_best_move` method in `engine.cpp`. This method will be called on an engine object when the server decides to search for a move, and it should do the following:

- Check the flag `search`. Proceed only if this flag is true. If this flag is false, then **terminate the search immediately**.
- Place the best move found at any point of time in `best_move`
- Not modify the board passed to it (Note that the board is declared `const`). You may make a copy of the board and modify that if needed.

The starter code's engine.cpp randomly picks a move from the moveset and sets the best move to it, as an example.

4.5 Python Bindings

- If you decide to code in python, you will be required to edit the `get_best_move` method in `engine.py` file. The function takes an instance of Board class, and is expected to return a single integer indicating the best move. The behavior of the function is similar as in C++ code.
- Before compiling, you would need to install pybind11. This can be done using

```
pip install pybind11
```

- To compile the code, instead of running 'make', you have to run 'make rollerball_py'. Then, if things work well, you can run the code using:

```
./bin/rollerball_py -p 8181
```

  We will finally evaluate your code on Python3.9.
- Instructions for running gui and server remains exactly the same as before.

4.6 Other details

- You are **not allowed** to use any libraries other than those that come with the language. This means that boost (for C++) and numpy (for Python) are not allowed.
- Your code must compile with the Makefile provided **without any modifications**.
- Your implemented algorithm should be **single-threaded**. You are not allowed to make use of multiple threads, asyncio or any other form of parallel execution.

5 Submission Details

- Submit a single zip containing two files: 1.) Readme.md and 2.) 'engine.cpp' if your code is c++, else 'engine.py' if in python.
- The zip file should be named 'entrynum1_entrynum2.zip' and running 'unzip file_name.zip' should extract the files in current directory.
- The 'Readme.md' must detail: a.) Name and entry number of teammates b) Names of all students you discussed/collaborated with (see guidelines on collaboration vs. cheating on the course home page). If you never discussed the assignment with anyone else say None. c.) A small writeup on the methods used (optional).
- You can use verification server to make sure your submission confirm with the evaluation protocol. Check piazza for updated details.
- Submission Deadline is **11:59pm, 21st September** on Moodle. Any updates will be posted on Piazza.

6 Phases

You will work on Rollerball Chess in two phases (assignments) in this course.

6.1 Phase 1

- Phase 1 (Assignment 2) will be worth 4 marks in course total.
- Your code will be evaluated against 3 TA bots, each of build with different algorithm and having different skill level. In total there will be 6 matches i.e once as white and once as black, against each of the 3 bots.
- Each match will be given a score, and your final score will be average of the 6 games.
- The performance in A2 will decide your seed for A5 Tournament 1.

6.2 Phase 2

- Phase 2 (Assignment 5) will be worth 12 marks in course total.
- Instead of evaluating against TA bots, A5 will be a tournament style competition against bots from the class.
- Bots will be evaluated based on their elo scores after the tournament.
- More details will be released at the time of Assignment.

7 General Guidelines

- You may work in teams of two or by yourself. If you work in a team of two then make sure you mention the team details in the write-up. Our recommendation – this will lead to a much more open ended final assignment (phase 2); hence best to work with a partner with whom you communicate well. You must team up with the same partner for the final assignment. The exception is if your partner withdraws the course, in which case, take explicit permission on Piazza for a change of partner.
- You are allowed to use any of python or C++. However, we **strongly recommend** C++. This is because your code will be much more time-efficient and hence given time limit, your bot will be able to perform better. Remember, marks will/may be awarded based on relative performance in class.
- Your code must be your own. You are not allowed to use **ChatGPT** or any **coding assistant** in general for this assignment. Note: It is not very difficult to catch such assistants :)
- Note that the GUI takes a server address to connect to, so it is possible to set up a hotspot and connect to your friends' bots to play against them. While we encourage collaboration in this manner, we strongly condone cheating. Please see the guidelines between collaboration and cheating at https://www.cse.iitd.ac.in/~mausam/courses/col333/autumn2023/.
- You must not discuss this assignment with anyone outside the class. Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team. Please read academic integrity guidelines on the course home page and follow them carefully.
- You get a zero if your player does not follow the submission guidelines in this document.
- We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.