UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG
School of Computer Science & Applied Mathematics

Advanced Analysis of Algorithms
2021 Assignment: Overview Document

1 Introduction

The most famous AI challenge of the 20th century was to defeat humans in the game of chess. Beginning with Claude Shannon's paper in 1950,[1] this goal was finally achieved in 1997, when IBM's *Deep Blue* defeated world champion Garry Kasparov. To honour this accomplishment, we will be developing an AI agent that plays the board game known as *Congo*. This document will outline exactly how the game works, what the assignment will consist of, and how marks will be allocated. More documents will be sent out at a later stage to expand on the requirements of Section 3.

2 The Game of *Congo*

Congo is a variant of chess that was invented in the 80s. It contains elements of chess, checkers and shogi (also known as Japanese chess), but the main philosophy is the same: to capture your opponent's most valuable piece. You can try out the game yourself here `https://mindsports.nl/index.php/dagaz/882-congo` and read more about it at `https://mindsports.nl/index.php/the-pit/523-congo` and `https://en.wikipedia.org/wiki/Congo_(chess_variant)`. For ease of reference, we describe the rules below.

The game is played on a 7×7 board. In the descriptions below, rows are called *ranks* and columns are called *files*. The start state looks as follows:

[1] `https://bit.ly/3tThS31`

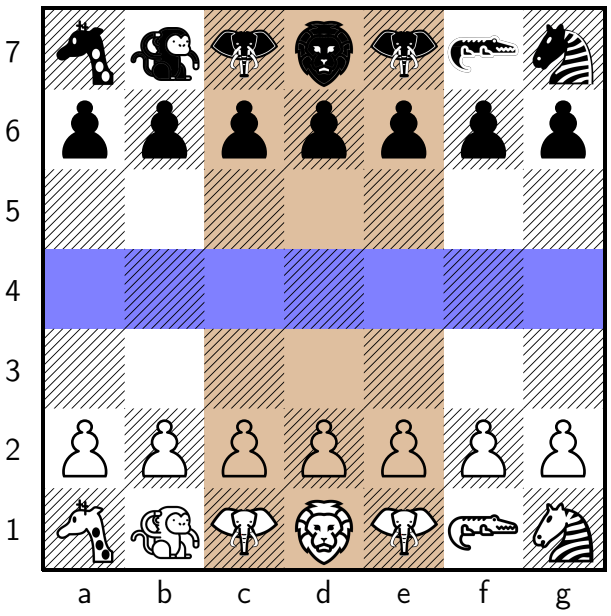


Figure 1: The starting position for White and Black. Initial positions: giraffes (a-file), monkeys (b-file), elephants (c- and e-files), lions (d-file), crocodiles (f-file), and zebras (g-file). Pawns fill each player's second rank. Each player's castles are highlighted in brown, and the river along the fourth rank is in blue.

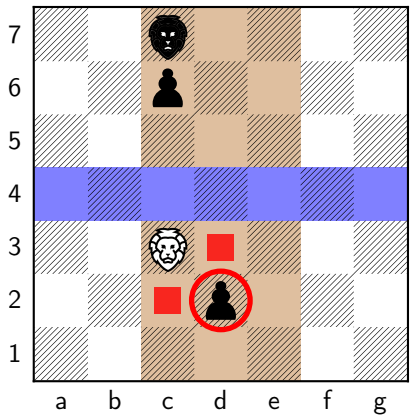
Game dynamics

Congo consists of two players, Black and White, who take turns moving (no skipping of turns is allowed). White starts first. The aim of the game is to capture the opponent's lion—the game is immediately over when one player's lion has been captured. Players take turns making moves and potentially capturing their opponent's pieces. If a player moves to a square that is occupied by an opposing piece, that piece is said to be *captured* and is removed from the board. In *Congo*, as in chess, a player is not allowed to capture their own pieces, so moving to a square occupied by one's own piece is prohibited.

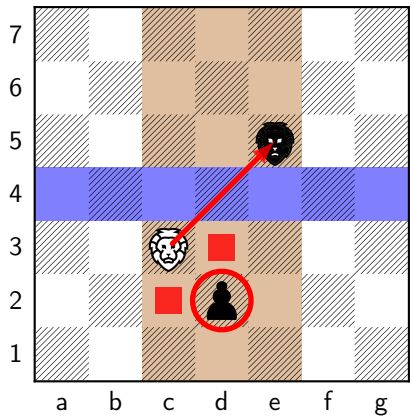
Pieces

Each side begins with a number of pieces of various types. These pieces are:

Lion: The lion moves and captures one square in any direction (including diagonally), but it may not leave its 3×3 castle (highlighted in brown in the diagrams). However, there is one exception to this rule: a lion is allowed to move in a straight or diagonal line across the river if it is able to immediately **capture the opposing lion**. If a player's lion is captured, that player immediately loses the game. The diagram below shows the moves the white lion is able to make.

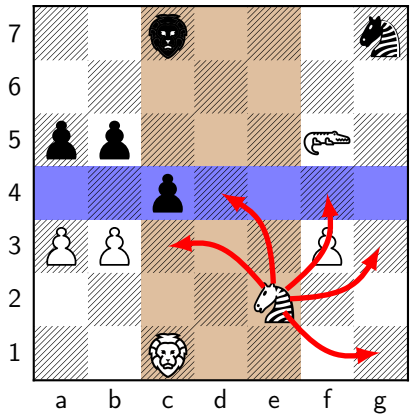


(a) In the current position, White's lion on c3 can move to the squares marked by a red square or can capture Black's pawn on d2 (red circle).

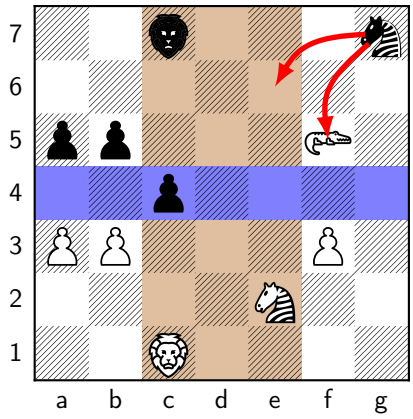


(b) Compared to the position on the left, the White lion on c3 can now also move across the river to capture the Black lion on e5 and win the game (red arrow)! The White lion could also potentially capture Black's lion anywhere along the c-file.

Zebra: If you are familiar with chess, the zebra moves exactly as the chess knight. That is, it moves two squares vertically and one square horizontally, or two squares horizontally and one square vertically (with both forming the shape of an L). The zebra can jump over pieces to reach its destination.

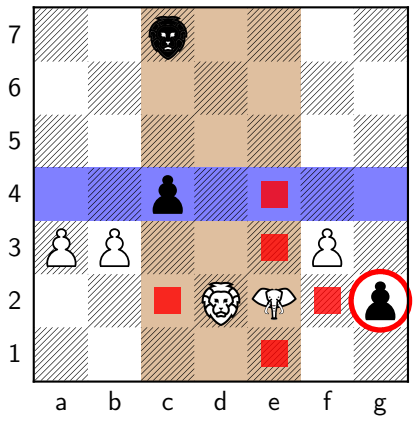


(a) The White zebra on e2 can move to any square indicated by the arrows. It cannot move to c1, since that square is occupied by another White piece. Note that the pawn on f3 does not prevent the zebra from jumping to f4 or g3.

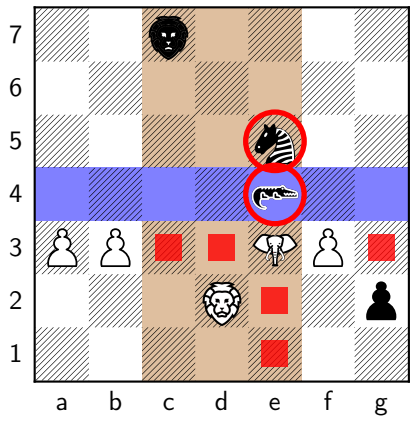


(b) The Black zebra can move to any square indicated by the arrows and can thus choose to capture the White piece on f5.

Elephant: The elephant can move or capture one or two squares **horizontally or vertically** in any direction. If it moves two squares, this move is a jump. Therefore, if there is any piece in the way, it does not block the elephant from moving two squares.

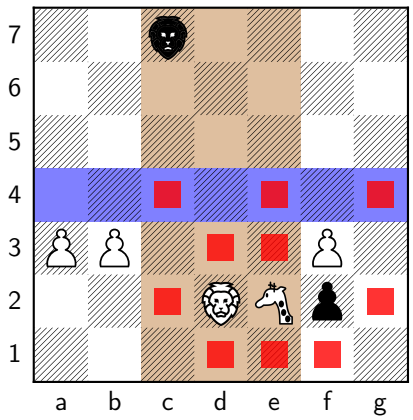


(a) The White elephant on e2 can move to any of the squares marked by a red square, and it can capture the Black piece in g2 (circled in red). Note that it cannot move to d2 since another White piece occupies that square, but it can jump to c2 (and is not blocked by the piece on d2).

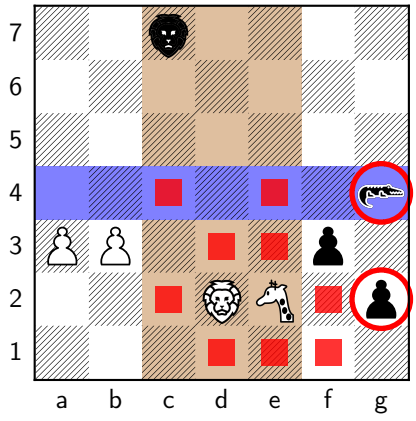


(b) The White elephant on e3 can move to any of the squares marked by a red square. In addition, it can move one square forward to capture the Black piece on e4, (circled in red), or jump two steps to capture the piece on e5 (in this case, the piece on e4 would remain unaffected).

Giraffe: The giraffe can move one or two steps in any direction (orthogonally or diagonally), but **can only make captures two squares away**. If the giraffe moves two squares, the intermediate square is jumped.

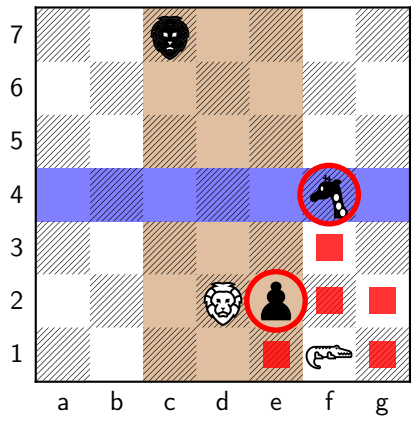


(a) The White giraffe on e2 can move to any of the squares marked by a red square. Note that it can move to c2, g2 or g4 by jumping over the intermediate pieces. However, it cannot move to d2 (which is occupied by a White piece) and cannot capture the Black piece on f2, since it is only one square away.

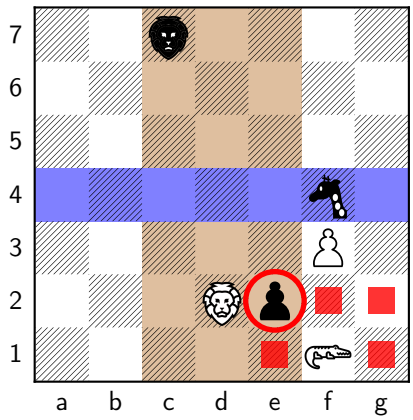


(b) The White giraffe on e2 can move to any of the squares marked by a red square. It cannot move to or capture f3 since it is only one square away, but it can jump two squares to capture the Black piece on g4 or g2 (red circles).

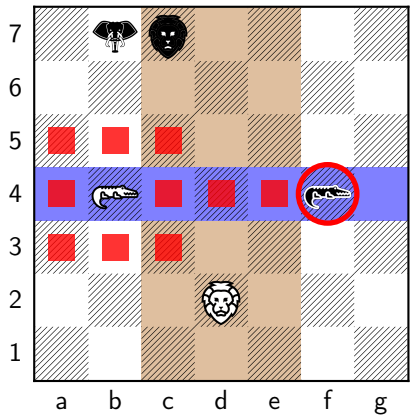
Crocodile: The crocodile moves and captures one step in any direction (orthogonally or diagonally). It can also move and capture any number of steps in a **straight line towards the river** (including entering the river). Once in the river, the crocodile can additionally move and capture any number of steps **along** the river.



(a) The White crocodile on f1 can move to any square marked by a red square. This includes any square adjacent to f1, and any square along the file between the crocodile and the river, including entering the river. The crocodile can therefore choose to capture the Black piece at e2 or f4 (red circle).

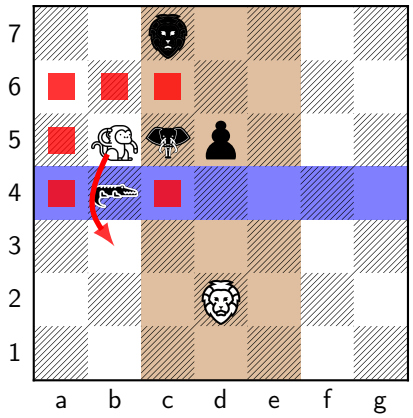


(b) A similar state to the one on the left. However, the White pawn on f3 now blocks the crocodile on f1 from entering the river (the crocodile cannot jump over it). It therefore cannot capture the black piece at f4.

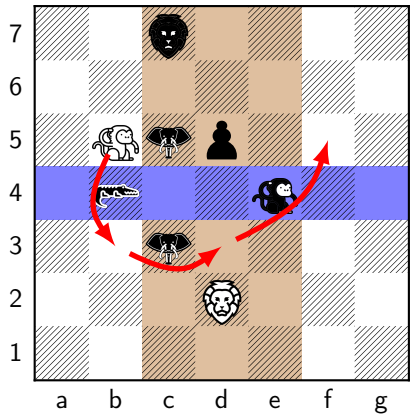


(c) Once in the river, the crocodile can move any number of squares along it. The White crocodile on b4 can move to any of the squares marked by a red square and can also capture the black piece at f4 (circled in red).

Monkey: The monkey can move one step in any direction (orthogonally or diagonally), **but cannot make a capture this way**. It can also capture an opposing piece on an adjacent square (orthogonally or diagonally) by jumping over it to the **vacant** square immediately beyond (if the square is not empty, the capture cannot be made). If a capture is made, then **multiple captures** are permitted (but are not compulsory) with the same piece in the same turn! In a multi-capture move: a) successive jumps can be in different directions; b) a given piece may be jumped only once; c) a given square may be visited more than once; d) all pieces jumped are removed from play only after the entire multi-capture move has been completed.



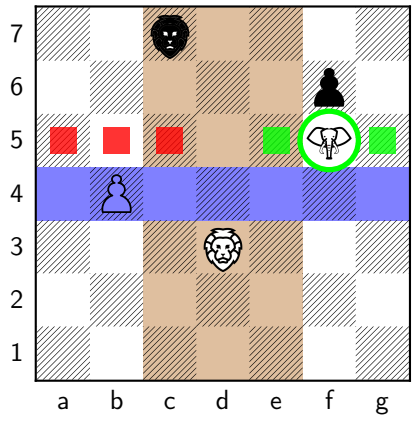
(a) The White monkey on b5 can move to any of the squares marked by a red square. It cannot capture the Black piece on c5 since it must jump over it to do so, but the square it would jump to (d5) is occupied. It can, however, capture the Black piece on b4 by jumping over it to b3 (red arrow). At the end of the turn, the piece on b4 would be removed.



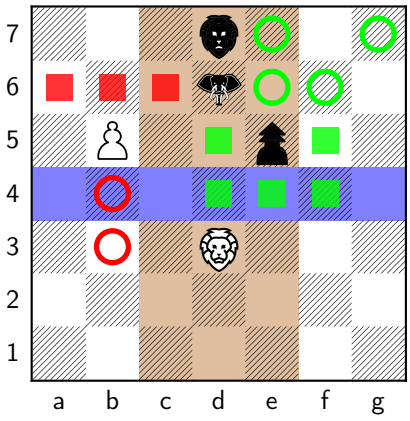
(b) An example of a multi-capture move. The White monkey on b5 first jumps to b3, capturing the Black piece on b4. It then continues by jumping to d3, capturing the piece on c3. Finally, it jumps to f5, capturing the piece on e4. This ends the turn, at which point all the pieces are removed from the board. All jumps are indicated by red arrows.

Pawn: A pawn can move and capture one step straight or diagonally forward. (Black pawns move forward towards rank 1, while White pawns move forward towards the rank 7.) When past the river, it can move (**but not capture**) one or two steps straight backwards. However, this is not a jumping move, and so the pawn may not move backwards if there is a piece in its path.

If a pawn reaches the opposite side of the board, it is *promoted* to a *superpawn*. A superpawn has the abilities of a regular pawn, but can additionally move and capture one square sideways. It may now retreat one or two squares **straight or diagonally backwards**, and this ability no longer depends on its position with regard to the river. However, just like the regular pawn, it may not capture or jump over a piece when retreating.



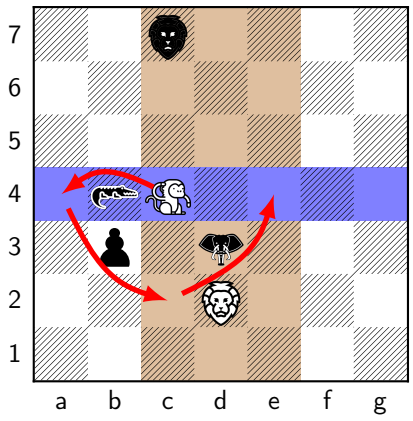
(a) In this position, neither pawn has managed to cross the river completely yet. The White pawn on b4 can therefore only advance onto any square indicated by a red square, while the Black pawn on f6 can only advance to any square indicated by a green square or capture the White piece on f5 (green circle).



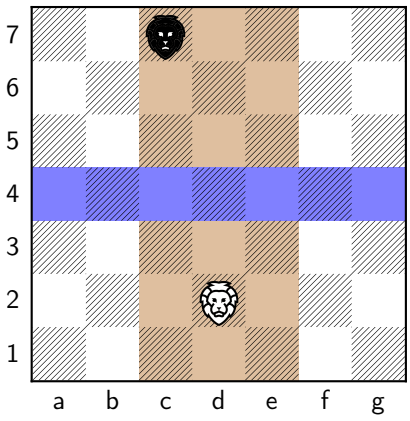
(b) The White pawn on b5 has crossed the river and so now can retreat one or two squares straight backwards (indicated by red circles) in addition to advancing forward (red squares). The Black superpawn on e5 can move and capture forwards or sideways (green squares) and can retreat one or two squares straight or diagonally backwards (green circles). In the case of the superpawn, retreating can happen anywhere on the board. The superpawn cannot retreat to c7, since there is a piece on d6 blocking its path.

Other Game Features

The River: As you saw in the initial setup, the board has a river running across the middle of it! Any piece may move into the river, but if that piece does not leave the river on its next turn, it is said to have drowned. The piece may not have moved at all, or may have moved within the river, or even, in the monkey's case, out of and back into the river. In all cases, the piece is removed from the board. **The only piece this does not apply to is the crocodile, which cannot drown and can stay in the river forever!**



(a) An example of drowning. The White monkey begins in the river on c4 and executes a multi-capture move, jumping to a4, then c2 and finally e4, which leads it back into the river.



(b) The state of the board after the move ends. The three Black pieces that were jumped over are removed from the board, but as the monkey landed back in the river, it too is removed.

End of game: The game ends when one of the lions is captured. If only two lions remain, the game is a draw. However, a lion and any piece will win against just a lion.

3 Groups and Grading

You may complete this assignment either individually or in groups of **two**. This assignment will consist of a number of components, some of which contain a competitive element.

As we have seen in the previous labs on search, in order to create the AI agent we must develop two components: the problem dynamics and the search algorithm itself. The roadmap for how this assignment will be graded is as follows:

3.1 Successor Function (30/100)

The first part of the assignment will require you to implement the successor function (and hence the rules of the game). The successor function will inform the agent at a given state: a) what actions are available (i.e. the legal moves), and b) the next state that occurs when a move is executed. We will also need to implement functions that convert the current state to a string

representation, and also accept a string representation and set up the board accordingly. For this we will be using Forsyth–Edwards Notation (https://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation). Successfully creating a working implementation of the game will constitute 30% of the overall mark.

3.2 Minimax (20/100)

Having implemented the game dynamics, the next step will be to implement the adversarial search procedure. In particular, we will need to implement Minimax search. This will form a reasonably strong AI capable of playing the game! Successfully creating a working implementation of a Minimax agent will constitute 20% of the overall mark.

3.3 Improvements (30/100)

Given the base Minimax algorithm, the next step is to add a number of improvements. If you are tackling this assignment individually, you will need to implement at least **two** improvements. However, if you have partnered up, then you must implement at least **four**. A list of potential improvements that may be included is provided below. You are welcome to implement other changes or ideas outside of these, but please contact myself or a tutor before doing so. We have not covered most of these, so implementing them will require some research on your part.

- Alpha-beta pruning (https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning)
- Improved evaluation function (https://en.wikipedia.org/wiki/Evaluation_function)
- Move ordering (https://www.chessprogramming.org/Move_Ordering)
- Aspiration windows (https://www.chessprogramming.org/Aspiration_Windows)
- Quiescence search (https://en.wikipedia.org/wiki/Quiescence_search)
- Principal variation search/Scout/Negascout (https://en.wikipedia.org/wiki/Principal_variation_search)
- Transposition tables (https://en.wikipedia.org/wiki/Transposition_table)
- Killer heuristic (https://en.wikipedia.org/wiki/Killer_heuristic)
- Pre-generated opening book moves (https://www.chessprogramming.org/Opening_Book)
- Pre-generated endgame tablebases (https://en.wikipedia.org/wiki/Endgame_tablebase)

In order to determine the effect of these improvements, you will need to submit a mini research report that describes the techniques that were used, and run a number of experiments to quantify how much each change improved the AI over and above the baseline minimax algorithm. These improvements could be in terms of performance (i.e. the agent plays better) or in terms of speed (e.g. given a fixed time budget, the agent is able to search deeper). A rubric detailing how this report is to be compiled will be made available later. The report will constitute 30% of the overall mark.

3.4 Competitive Elements (20/100)

One important component of a strong game-playing agent is the ability to generate the game tree as fast as possible. 10% of the mark will be based on how many nodes your agent can generate per second. An exact breakdown of this portion will be provided in due course.

The final 10% of the mark will depend on how strong your program is compared to other submissions from the class. Everyone's agent will play against one another in a round-robin tournament, and marks will be allocated based on overall final rankings.