Maverick Woodworth
Game Theory and Networks
Project Report

- **Problem Statement:**

This project centers around the game of chess in a modified end game scenario. While familiar to most, a quick recap of the game of chess will be beneficial moving forward. Chess consists of two opposing players (White and Black), each with their armies of assorted pieces. These pieces consist of pawns, rooks, knights, bishops, a queen, and most importantly the king. The overall goal of the game for each player is to checkmate the opponent's king, securing victory. A checkmate is when a king piece is in direct danger from an opposing player's piece, with no way to escape to safety.

Unless you're a grandmaster with years of experience or superhuman, it is probably difficult to envision the full scope of possibilities more than a few moves in advance. For many, the move they choose is often a greedy approach with only minimal foresight. So how can game theory be applied to chess? Instead of greedily choosing your next move based on the immediate layout of the board or only looking a move or two ahead at most, applying game theory allows us to take the full scope of the game into account with all its potentialities before evaluating our next move.

While daunting, the game of chess is nonetheless a 'perfect information' game, with complete knowledge of the game available at all times to all players. Chess is also a sequential game, with players taking turns making moves in response to one another. Because of this, we can abstract the game of chess as a decision tree. This decision tree is traversed from the root (beginning of the game) to a leaf node (end state) by a unique sequence dictated by the opposing players' chosen moves. With a formalized utility payoff function, backward induction can be applied to solve for subgame-perfect nash equilibria based on a given search depth through the tree.

The standard game of chess is prohibitively complex to analyze for this report. Some modifications have been made so that implementing this game theoretical approach is feasible as well as practical for testing and experimentation with different board layouts in an end game scenario:

- 4x4 layout as opposed to the standard 8x8 board
- Individual pieces prioritize capturing / placing opposing king in check to limit search space
  - Otherwise moves into empty spaces is allowed to progress game
- Special moves not considered:
  - Castling
  - En Passant, Pawn Promotion
- Max depth limit of ~12 moves used for analysis although most games were solved sooner
  - If no solution found, results in a tie

Remaining aspects of the game remain the same. There are two players, Black and White with their respective pieces. The strategies each player can employ is represented by the pieces at their disposal and the available moves they can make at any given time with those pieces. Pieces' movement is the same as in regular chess, barring special moves noted above. Utility calculation is another critical aspect to get right for this application of game theory to be viable. As mentioned, leaf nodes in this decision tree represent end states for the game. End states can include either Black or White winning, a stalemate, or a tie. A win results in a utility payoff of 1 for the winner, while a loss results in a utility payoff of -1 for the loser. Additionally, a stalemate or tie both result in a utility payoff of 0 for both players. The utility payoffs are detailed in the table below:

	White Utility	**Black Utility**
White Wins by Checkmate	1	-1
Black Wins by Checkmate	-1	1
Stalemate	0	0
Tie	0	0

- **Analysis:**

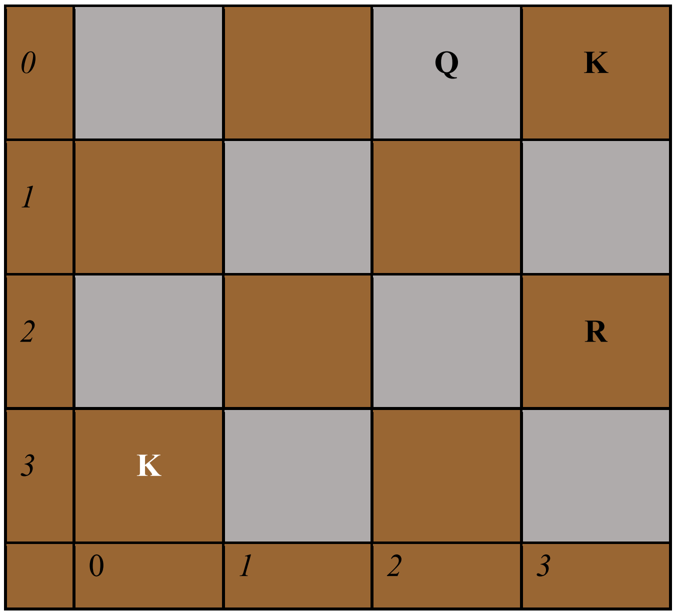
To implement backward induction, a min-max algorithmic approach was utilized from the perspective of Black. Evaluating the full scope of possibilities of the game, Black is trying to maximize their utility by winning while White is simultaneously trying to minimize this value, winning themselves.

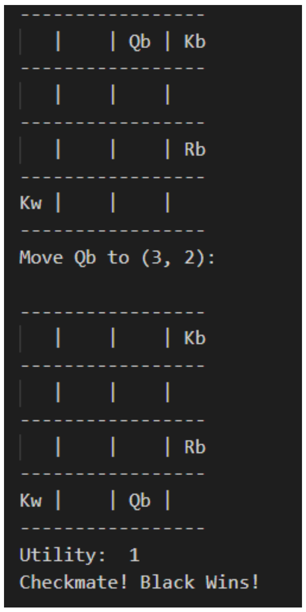
For my analysis, a series of scenarios were simulated with different board setups representing a particular end game scenario with varying difficulties. A maximum search depth of 12 is used for evaluation of the following games when necessary. If it is possible for Black to checkmate or if checkmate by White is inevitable, output will include the sequence of moves and resulting chessboards leading to that result along with the utility for Black. This sequential output of the players' moves represents an SPNE for that end game scenario. If no solution is found within the max search depth, the scenario is ruled a tie with a utility of 0 reported for Black and implied for White. What follows are a series of example games, beginning with easy situations, and progressing to more competitive games. In each game, Black is next to move.

Easy:

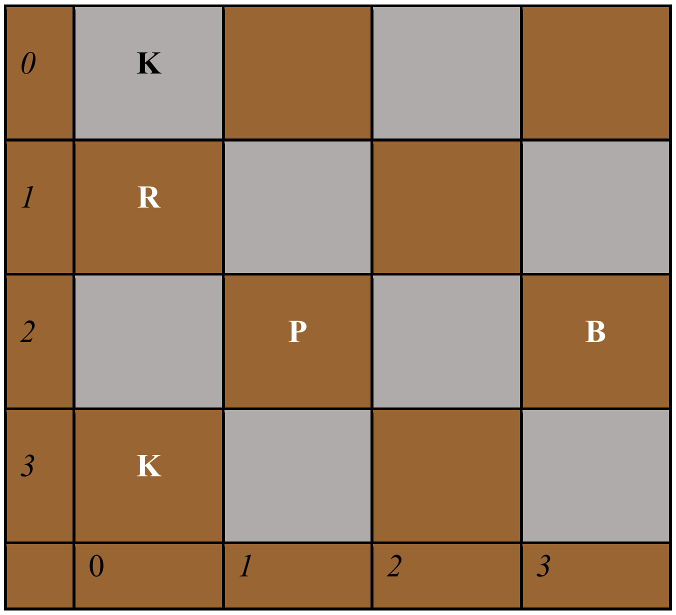
To begin and to ensure that the algorithm works as intended, simple layouts with obvious endings were tested. The first of which being an obvious 1-move checkmate by Black, followed by an inevitable checkmate by White.

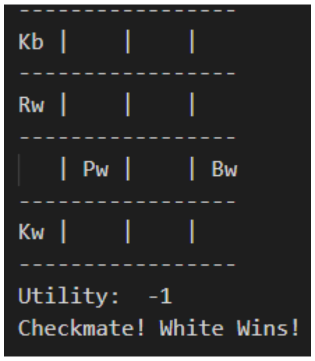
i. Black Checkmate (1 move; depth = 1; Utilities = [Black (1), White (-1)]):




```
----------------
   |    | Qb | Kb
----------------
   |    |    |
----------------
   |    |    | Rb
----------------
Kw |    |    |
----------------
Move Qb to (3, 2):

----------------
   |    |    | Kb
----------------
   |    |    |
----------------
   |    |    | Rb
----------------
Kw |    | Qb |
----------------
Utility:  1
Checkmate! Black Wins!
```

ii. Checkmate by White Inevitable (no moves available; Utilities = [Black (-1), White (1)]):

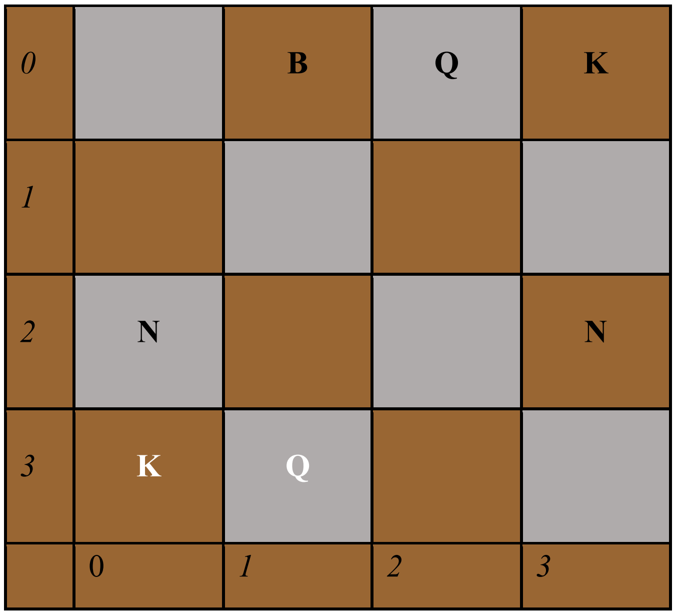


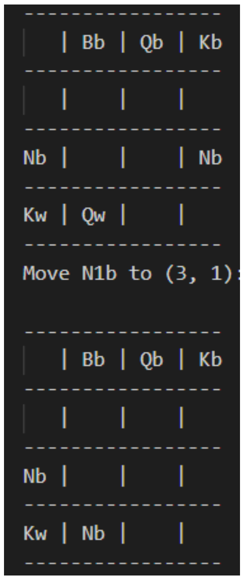

```
-----------------
Kb |    |    |
-----------------
Rw |    |    |
-----------------
   | Pw |    | Bw
-----------------
Kw |    |    |
-----------------
Utility:  -1
Checkmate! White Wins!
```

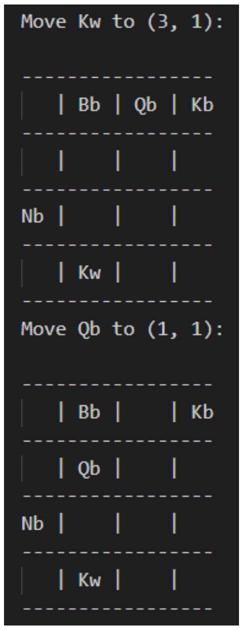
Medium:

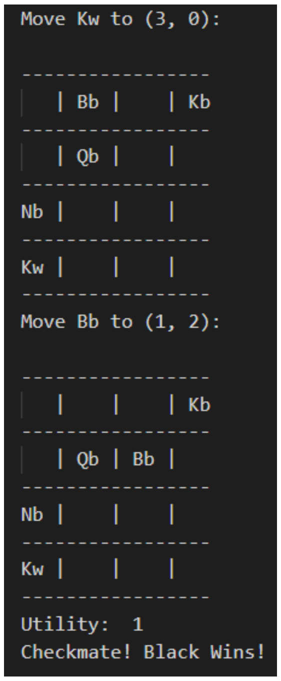
In this next example, a more competitive layout requires more moves for a winning solution to be found:

i. Black Checkmate (5 moves; depth = 5; Utilities = [Black (1), White (-1)]):




```
----------------
   | Bb | Qb | Kb
----------------
   |    |    |
----------------
Nb |    |    | Nb
----------------
Kw | Qw |    |
----------------
Move N1b to (3, 1):

----------------
   | Bb | Qb | Kb
----------------
   |    |    |
----------------
Nb |    |    |
----------------
Kw | Nb |    |
----------------
```

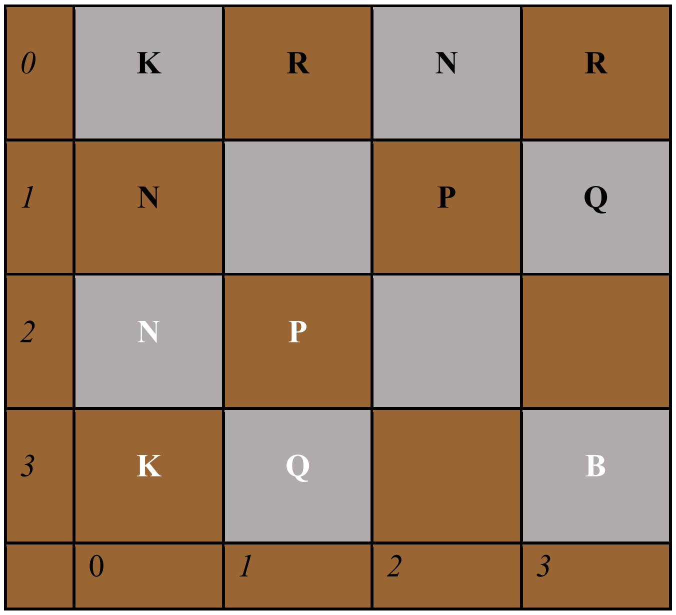

```
Move Kw to (3, 1):

----------------
   | Bb | Qb | Kb
----------------
   |    |    |
----------------
Nb |    |    |
----------------
   | Kw |    |
----------------
Move Qb to (1, 1):

----------------
   | Bb |    | Kb
----------------
   | Qb |    |
----------------
Nb |    |    |
----------------
   | Kw |    |
----------------
```

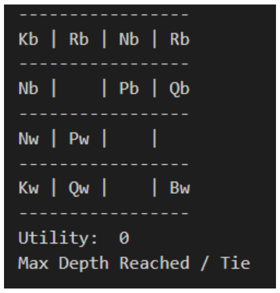

```
Move Kw to (3, 0):

----------------
   | Bb |    | Kb
----------------
   | Qb |    |
----------------
Nb |    |    |
----------------
Kw |    |    |
----------------
Move Bb to (1, 2):

----------------
   |    |    | Kb
----------------
   | Qb | Bb |
----------------
Nb |    |    |
----------------
Kw |    |    |
----------------
Utility:  1
Checkmate! Black Wins!
```

Hard:

More difficult scenarios that require an even greater search depth for a potential solution:

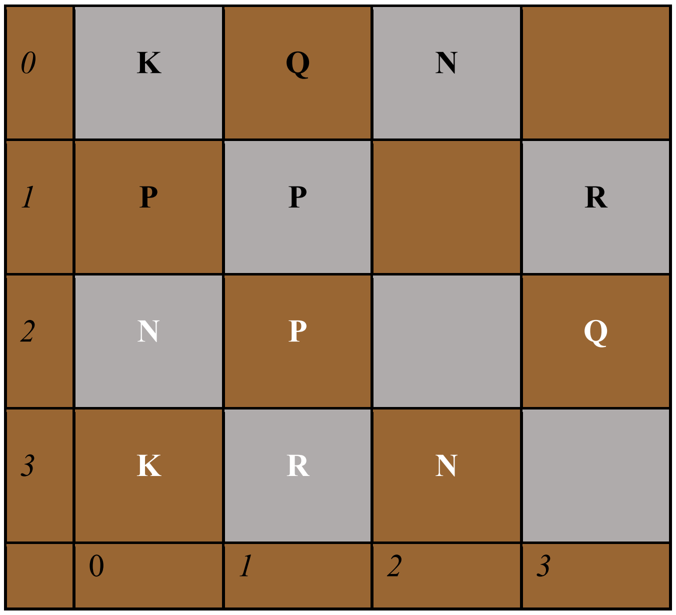
i. No Solution Found (depth = 12; Utilities = [Black (0), White (0)]):

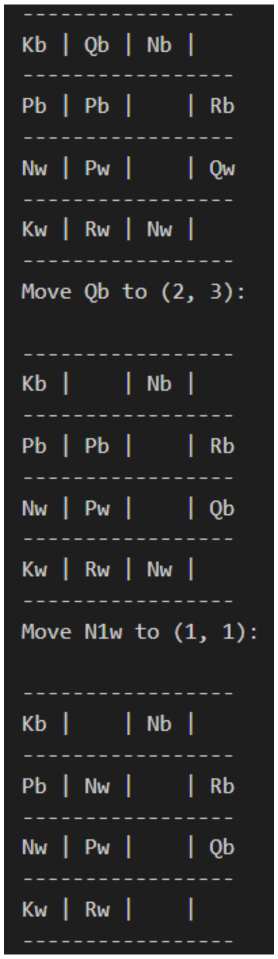




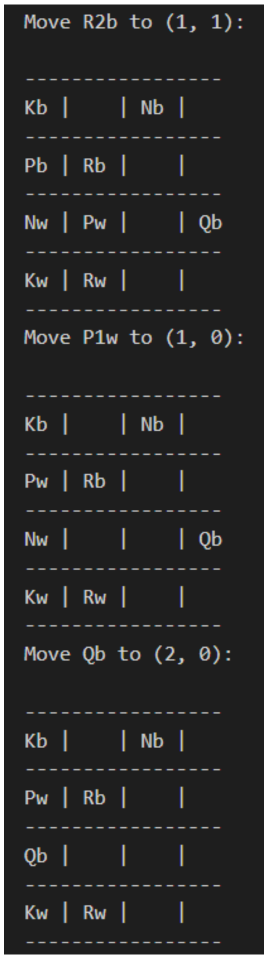
```
----------------
Kb | Rb | Nb | Rb
----------------
Nb |    | Pb | Qb
----------------
Nw | Pw |    |
----------------
Kw | Qw |    | Bw
----------------
Utility:  0
Max Depth Reached / Tie
```

ii. Black Checkmate (7 moves; depth = 7; Utilities = [Black (1), White (-1)])

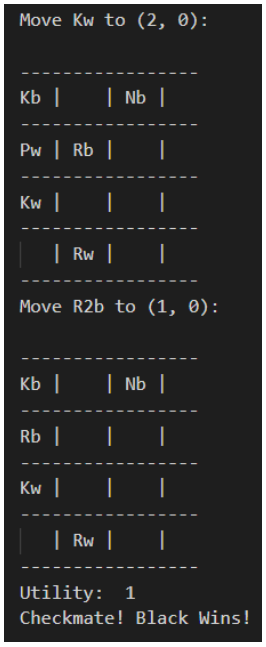




```
-----------------
Kb | Qb | Nb |
-----------------
Pb | Pb |    | Rb
-----------------
Nw | Pw |    | Qw
-----------------
Kw | Rw | Nw |
-----------------
Move Qb to (2, 3):

-----------------
Kb |    | Nb |
-----------------
Pb | Pb |    | Rb
-----------------
Nw | Pw |    | Qb
-----------------
Kw | Rw | Nw |
-----------------
Move N1w to (1, 1):

-----------------
Kb |    | Nb |
-----------------
Pb | Nw |    | Rb
-----------------
Nw | Pw |    | Qb
-----------------
Kw | Rw |    |
-----------------
```



```
Move R2b to (1, 1):

-----------------
Kb |    | Nb |
-----------------
Pb | Rb |    |
-----------------
Nw | Pw |    | Qb
-----------------
Kw | Rw |    |
-----------------
Move P1w to (1, 0):

-----------------
Kb |    | Nb |
-----------------
Pw | Rb |    |
-----------------
Nw |    |    | Qb
-----------------
Kw | Rw |    |
-----------------
Move Qb to (2, 0):

-----------------
Kb |    | Nb |
-----------------
Pw | Rb |    |
-----------------
Qb |    |    |
-----------------
Kw | Rw |    |
-----------------
```



```
Move Kw to (2, 0):

-----------------
Kb |    | Nb |
-----------------
Pw | Rb |    |
-----------------
Kw |    |    |
-----------------
   | Rw |    |
-----------------
Move R2b to (1, 0):

-----------------
Kb |    | Nb |
-----------------
Rb |    |    |
-----------------
Kw |    |    |
-----------------
   | Rw |    |
-----------------
Utility:  1
Checkmate! Black Wins!
```

- **Conclusion:**

These example scenarios show that using backward induction, game theory can be applied to sequential, turn-based games like chess. In this modified version, SPNE were able to be retrieved when the initial board layout proved solvable based on the depth of the search performed.