AI Mini Project

Report - 2

Student 1
Name : G.Sachin Sai Reddy
Roll no : CS20B1088

Student 2
Name : K.Nishank
Roll no : CS20B1095

Project Title : AI Chess game with min - max Algorithm, Heuristics for reduced state space

These the are the following steps which we have done for solving problem

Step 1 :

Initially, we start by defining the initial heuristic values for each piece.
Example : heuristic value of pawn :1,
Heuristic value of Knight : 3,..etc.
We initialize the state of the chess board.

Step 2 :

Instead of directly going for any pruning technique . we use a position score

```
position_score = sum_of_pieces + king_castled + pawn_islands + free_bishops + forward_knights
```

This pieces are defined like this ;

forward pawn (more likely to castle)
bishops with open diagonals

Pawn_islands
Forward knights

We will develop pieces in a similar manner and we iterate the entire chessboard to get the optimum value .

Iterate through every row on the chessboard and calculate heuristics adjustment. For the first iteration of this, I will look for developed pieces.
We use This position score in the further step of tree generation which is the most important step in the game process.

Step 3 :

Tree generation algorithm by using brute force we generate entire state space
For instance , Assume we have 15 possible moves for each turn. After 5 moves our tree size would be (15)^5 ,Which is around 0.75 million .
One way to reduce the size of the search tree is by limiting the number of moves per tree to around 5. This could lead to a significant reduction in the size of the tree, allowing me to potentially calculate up to 10 moves into the future without using too much memory or CPU power.
As the game progresses, it may not be necessary to store each position in the tree beyond the second move, as only the score may be needed.
For the initial iteration of the algorithm, I will calculate three moves ahead to start building the search tree.

```
self.current_game_state = TreeNode([copy.deepcopy(self.chessboard),0])
        current_positions = [self.current_game_state]

        #track the number of moves into the future you are calculating.
        current_depth = 1
        target_depth = depth_override or self.depth
        current_turn = copy.copy(self.current_turn)

        #keep searching until the desired AI depth has been reached.
        while current_depth <= target_depth:
            for position in current_positions:
                #returns a dictionary of possible chess moves
                pos_moves =
RulesEnforcer.all_possible_moves(position.data[0], current_turn)
```

```
                for start, moves in pos_moves.items():
                    for move in moves:
                        current_pos = position.data[0]
                        new_pos = ChessAi.make_hypothetical_move(start,
move, current_pos)

                        if current_turn == 'w':
                            score = ChessAi.position_evaluator(new_pos)
                        else:
                            #if black, store the negative score because
black wants to play the best move
                            score = -ChessAi.position_evaluator(new_pos)

                        if current_depth > 1:
                            position.add_child([new_pos, score])
                        else:
                            position.add_child([new_pos, score, start,
move])

            current_depth += 1
```

This is the code which we use as a tree generator .

Step 4 :

The Minimax algorithm is employed to determine the optimal moves at each level of the tree. It accepts a tree of moves and applies the minimax approach to select the best option within that tree. The heuristic algorithm is utilized to assess the chess moves. Recursion is used to traverse down the tree to the leaf node, after which the minimax algorithm is applied to work backward and compute the original value.

input: Starting node of the possible move tree (created by the tree generator function)
output: The best move to make at the current state (str)

Step 5 :

We will the define the rules for each piece for Rook ,knight queen , bishop
By following the standard rules which are present in chess . we will have a
Collision detection function which will be used in this state of rules to avoid collision of pieces

Finally we will have a move state function in order to make a move and state at every move is printed as follows below :

Outputs So far :

Board -

```
Lets play chess!!! Here is the board:

['b-r', 'b-n', 'b-b', 'b-q', 'b-k', 'b-b', 'b-n', 'b-r']
['b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['w-p', 'w-p', 'w-p', 'w-p', 'w-p', 'w-p', 'w-p', 'w-p']
['w-r', 'w-n', 'w-b', 'w-q', 'w-k', 'w-b', 'w-n', 'w-r']
```

White - Player Black - AI

Player Move -

```
Your turn:
Enter starting point coordinate: e2
Enter ending point coordinate: e4
You have made a move!

['b-r', 'b-n', 'b-b', 'b-q', 'b-k', 'b-b', 'b-n', 'b-r']
['b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', 'w-p', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['w-p', 'w-p', 'w-p', 'w-p', '0-0', 'w-p', 'w-p', 'w-p']
['w-r', 'w-n', 'w-b', 'w-q', 'w-k', 'w-b', 'w-n', 'w-r']
```

AI Move -

```
AI is thinking...
g8
['f', 6]
AI has made a move!


['b-r', 'b-n', 'b-b', 'b-q', 'b-k', 'b-b', '0-0', 'b-r']
['b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p']
['0-0', '0-0', '0-0', '0-0', '0-0', 'b-n', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', 'w-p', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['w-p', 'w-p', 'w-p', 'w-p', '0-0', 'w-p', 'w-p', 'w-p']
['w-r', 'w-n', 'w-b', 'w-q', 'w-k', 'w-b', 'w-n', 'w-r']
```

AI killing Player's Pawn -

```
Your turn:
Enter starting point coordinate: e5
Enter ending point coordinate: e6
You have made a move!

['b-r', 'b-n', 'b-b', 'b-q', 'b-k', 'b-b', '0-0', 'b-r']
['b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p', 'b-p']
['0-0', '0-0', '0-0', '0-0', 'w-p', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', 'b-n', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['w-p', 'w-p', 'w-p', 'w-p', '0-0', 'w-p', 'w-p', 'w-p']
['w-r', 'w-n', 'w-b', 'w-q', 'w-k', 'w-b', 'w-n', 'w-r']


AI is thinking...
d7
['e', 6]
AI has made a move!


['b-r', 'b-n', 'b-b', 'b-q', 'b-k', 'b-b', '0-0', 'b-r']
['b-p', 'b-p', 'b-p', '0-0', 'b-p', 'b-p', 'b-p', 'b-p']
['0-0', '0-0', '0-0', '0-0', 'b-p', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', 'b-n', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['w-p', 'w-p', 'w-p', 'w-p', '0-0', 'w-p', 'w-p', 'w-p']
['w-r', 'w-n', 'w-b', 'w-q', 'w-k', 'w-b', 'w-n', 'w-r']
```

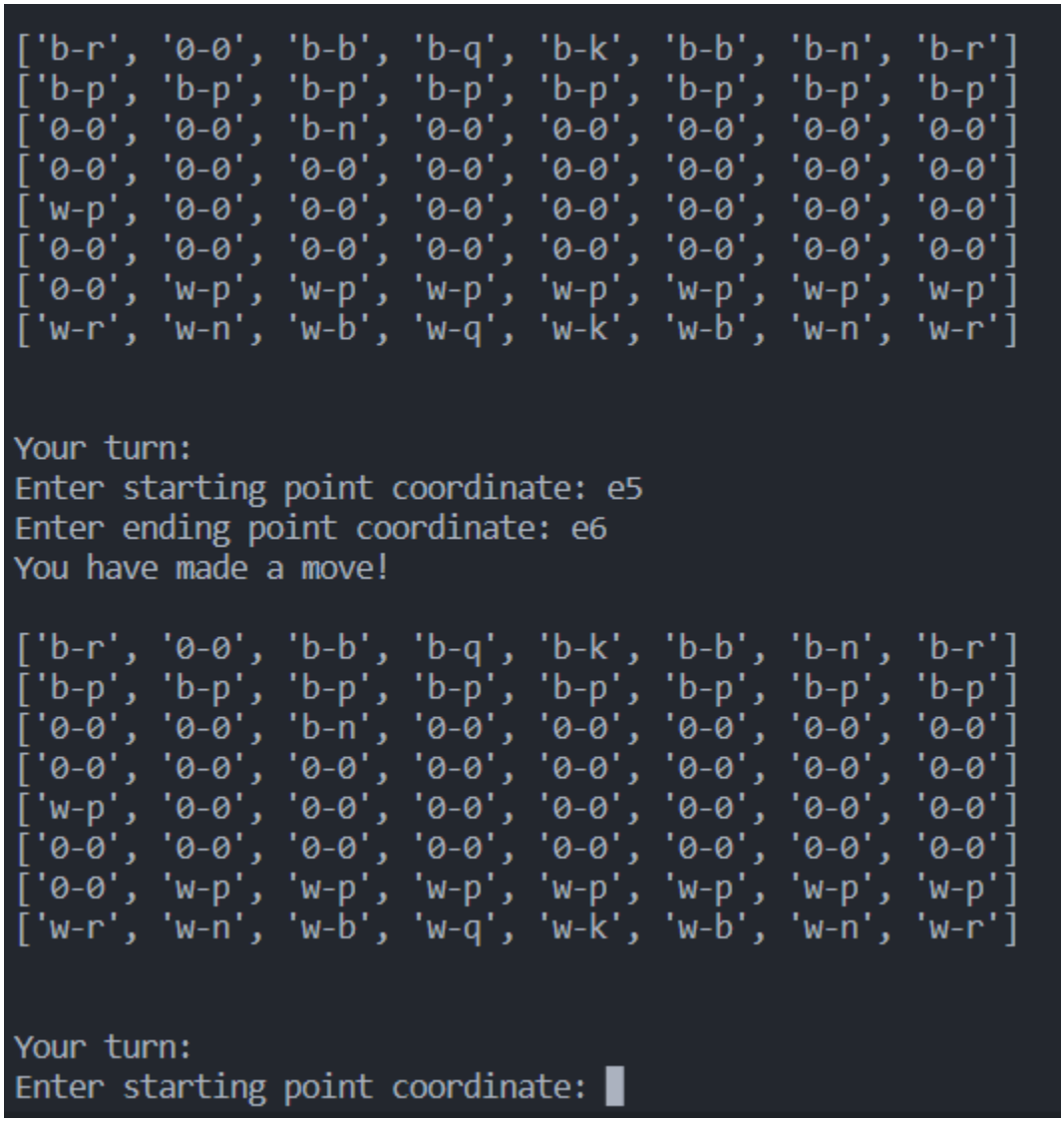
Player Checkmate’s AI king -

```
Your turn:
Enter starting point coordinate: f1
Enter ending point coordinate: b5
You have made a move!

['b-r', 'b-n', 'b-b', 'b-q', 'b-k', 'b-b', '0-0', 'b-r']
['b-p', 'b-p', 'b-p', '0-0', 'b-p', 'b-p', 'b-p', 'b-p']
['0-0', '0-0', '0-0', '0-0', 'b-p', '0-0', '0-0', '0-0']
['0-0', 'w-b', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', 'b-n', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['w-p', 'w-p', 'w-p', 'w-p', '0-0', 'w-p', 'w-p', 'w-p']
['w-r', 'w-n', 'w-b', 'w-q', 'w-k', '0-0', 'w-n', 'w-r']
```

AI saving king from Check -

```
AI is thinking...
c7
['c', 6]
AI has made a move!


['b-r', 'b-n', 'b-b', 'b-q', 'b-k', 'b-b', '0-0', 'b-r']
['b-p', 'b-p', '0-0', '0-0', 'b-p', 'b-p', 'b-p', 'b-p']
['0-0', '0-0', 'b-p', '0-0', 'b-p', '0-0', '0-0', '0-0']
['0-0', 'w-b', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', 'b-n', '0-0', '0-0', '0-0']
['0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0', '0-0']
['w-p', 'w-p', 'w-p', 'w-p', '0-0', 'w-p', 'w-p', 'w-p']
['w-r', 'w-n', 'w-b', 'w-q', 'w-k', '0-0', 'w-n', 'w-r']
```

Invalid Move by Player -



If the player plays a invalid move as we can see in this case, there is no piece in e5 yet the player gives those coordinates and tries to make a move, The move thus results as void and the game asks to make a move again.