Sahej Bhatia A00766737
COMP 8085

In the main function, a few segments are commented. Every segment is for playing with self. There are various segments that would allow for this to happen. If we were to uncomment the second last segment for example, the MCTSAI starts playing with self.

There is a model saved - mcts.model – you should be able to call it

I have added features at the end of each algo -that should save the moves made to a textfile.

Min-Max Algo

Pseduocode

```
Func minmax()
    If depth is 0 or node is lead then
        Return eval of node
    If isMaxNode then
        Max_val = - infinity
        For each child of node
            Node_val = minmax(child, depth-1, false, )
            Max_val = max(max_val, node_val)
            Undo the last move
        Return max_val
    else
        min_val = nfinity
        for each child of node do
            node_val = minmax(child, depth-1, false)
            min_val = max(min_val, node_val)
            undo the last move
        return min_val
```

Heuristic

I used the piecewise evaluation for this heuristic

- Count All black pieces and multiply them with their weights.
- piece_values = {'p': 1, 'b': 3, 'n': 3, 'r': 5, 'q': 9, 'k': 0} # piece values
- Count all white pieces.
- If current player is “White”, evaluate Count_White_Pieces - Count_Black_Pieces.
- Multiply that with 100 for better clarity

Check/CheckMate Evaluation

- If current player (AI) has put a Check, multiply the evaluation points by 100

A few thoughts

- Each leaf node's value is determined using a static evaluation function
- By removing a few branches from the minmax method, it can be made more efficient - Improved time complexity
- Time constraint – needs to search the whole space
-

MINMAX AI vs GENERIC AI [AI PLAYS AS WHITE]

- {'White': {'Wins': 1, 'Loses': 0, 'Draws': 0}, 'Black': {'Wins': 0, 'Loses': 1, 'Draws': 0}}
- {'White': {'Wins': 2, 'Loses': 0, 'Draws': 0}, 'Black': {'Wins': 0, 'Loses': 2, 'Draws': 0}}
- {'White': {'Wins': 3, 'Loses': 0, 'Draws': 0}, 'Black': {'Wins': 0, 'Loses': 3, 'Draws': 0}}
- {'White': {'Wins': 3, 'Loses': 0, 'Draws': 1}, 'Black': {'Wins': 0, 'Loses': 3, 'Draws': 1}}
- {'White': {'Wins': 3, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 3, 'Draws': 2}}
- {'White': {'Wins': 4, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 4, 'Draws': 2}}
- {'White': {'Wins': 5, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 5, 'Draws': 2}}
- {'White': {'Wins': 6, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 6, 'Draws': 2}}
- {'White': {'Wins': 7, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 7, 'Draws': 2}}
- {'White': {'Wins': 8, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 8, 'Draws': 2}}
- {'White': {'Wins': 9, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 9, 'Draws': 2}}
- {'White': {'Wins': 10, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 10, 'Draws': 2}}
- {'White': {'Wins': 11, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 11, 'Draws': 2}}
- {'White': {'Wins': 12, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 12, 'Draws': 2}}
- {'White': {'Wins': 13, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 13, 'Draws': 2}}
- {'White': {'Wins': 14, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 14, 'Draws': 2}}
- {'White': {'Wins': 15, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 15, 'Draws': 2}}
- {'White': {'Wins': 16, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 16, 'Draws': 2}}
- {'White': {'Wins': 17, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 17, 'Draws': 2}}
- {'White': {'Wins': 18, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 18, 'Draws': 2}}

The AI pulls a Win 18 times out of 20, and draws 2 times, with no losses.

MINMAX AI Self play

- {'White': {'Wins': 0, 'Loses': 0, 'Draws': 1}, 'Black': {'Wins': 0, 'Loses': 0, 'Draws': 1}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 1}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 1}}

- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 2}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 2}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 3}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 3}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 4}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 4}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 5}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 5}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 6}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 6}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 7}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 7}}
- {'White': {'Wins': 0, 'Loses': 2, 'Draws': 7}, 'Black': {'Wins': 2, 'Loses': 0, 'Draws': 7}}
- {'White': {'Wins': 0, 'Loses': 2, 'Draws': 8}, 'Black': {'Wins': 2, 'Loses': 0, 'Draws': 8}}
- {'White': {'Wins': 0, 'Loses': 2, 'Draws': 9}, 'Black': {'Wins': 2, 'Loses': 0, 'Draws': 9}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 9}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 9}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 10}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 10}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 11}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 11}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 12}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 12}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 13}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 13}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 14}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 14}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 15}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 15}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 16}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 16}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 17}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 17}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 18}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 18}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 19}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 19}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 20}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 20}}
- {'White': {'Wins': 0, 'Loses': 3, 'Draws': 21}, 'Black': {'Wins': 3, 'Loses': 0, 'Draws': 21}}
- {'White': {'Wins': 0, 'Loses': 4, 'Draws': 21}, 'Black': {'Wins': 4, 'Loses': 0, 'Draws': 21}}
- {'White': {'Wins': 0, 'Loses': 5, 'Draws': 21}, 'Black': {'Wins': 5, 'Loses': 0, 'Draws': 21}}
- {'White': {'Wins': 0, 'Loses': 5, 'Draws': 22}, 'Black': {'Wins': 5, 'Loses': 0, 'Draws': 22}}
- {'White': {'Wins': 0, 'Loses': 5, 'Draws': 23}, 'Black': {'Wins': 5, 'Loses': 0, 'Draws': 23}}
- {'White': {'Wins': 0, 'Loses': 5, 'Draws': 24}, 'Black': {'Wins': 5, 'Loses': 0, 'Draws': 24}}
- {'White': {'Wins': 0, 'Loses': 5, 'Draws': 25}, 'Black': {'Wins': 5, 'Loses': 0, 'Draws': 25}}

The AI shows a high count of draws. Which is in line, as 2 identical AI is playing against each other.

Iterative Deepening Search

Both the DFS and the BFS algorithms are incorporated into this iterative deepening technique. With this search technique, you may determine the optimal depth by increasing the limit incrementally until you reach your objective. Deep-first search up to a specific "depth limit" is performed, and the depth limit is increased after every iteration until the goal node is discovered. Breadth-first search's rapid search & depth-first search's storage efficiency are combined in this algorithm. When the search space is huge and

the depth of the goal node is unknown, the iterative search method is an effective uninformed search tool.

The branching factor is 4 on average. Approximately 50 moves are required to complete an average game.

Pseudocode

There are five steps in iterative deepening technique.

- Create the entire game's tree.
- All terminal states will be affected by the utility function.
- Depending on depth, the utility of a terminal states can be used to compute a real utility for their parents.
- Continue to the root node.
- Choose the move that has the highest value from the root node.

Iterative deepening or setting a preset depth is the quickest way to put an end to the search.

Thoughts

- Time take is exponential to reach goal node
- Very repetitive
- Time consuming
- Machine timed out multiple times
-

ITERATIVE DEEPENING AI Self play results

- {'White': {'Wins': 0, 'Loses': 0, 'Draws': 1}, 'Black': {'Wins': 0, 'Loses': 0, 'Draws': 1}}
- {'White': {'Wins': 0, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 0, 'Draws': 2}}
- {'White': {'Wins': 1, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 1, 'Draws': 2}}
- {'White': {'Wins': 1, 'Loses': 0, 'Draws': 3}, 'Black': {'Wins': 0, 'Loses': 1, 'Draws': 3}}
- {'White': {'Wins': 1, 'Loses': 0, 'Draws': 4}, 'Black': {'Wins': 0, 'Loses': 1, 'Draws': 4}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 4}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 4}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 5}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 5}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 6}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 6}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 7}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 7}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 8}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 8}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 9}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 9}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 10}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 10}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 11}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 11}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 12}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 12}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 13}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 13}}

- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 14}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 14}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 15}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 15}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 16}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 16}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 17}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 17}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 18}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 18}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 19}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 19}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 20}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 20}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 21}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 21}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 22}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 22}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 23}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 23}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 24}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 24}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 25}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 25}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 26}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 26}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 27}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 27}}
- {'White': {'Wins': 1, 'Loses': 1, 'Draws': 28}, 'Black': {'Wins': 1, 'Loses': 1, 'Draws': 28}}

There are 28 draws with only one decisive result. This is expected.

ITERATIVE DEEPENING AI vs MiniMax AI results

- {'White': {'Wins': 1, 'Loses': 0, 'Draws': 0}, 'Black': {'Wins': 0, 'Loses': 1, 'Draws': 0}}
- {'White': {'Wins': 2, 'Loses': 0, 'Draws': 0}, 'Black': {'Wins': 0, 'Loses': 2, 'Draws': 0}}
- {'White': {'Wins': 3, 'Loses': 0, 'Draws': 0}, 'Black': {'Wins': 0, 'Loses': 3, 'Draws': 0}}
- {'White': {'Wins': 3, 'Loses': 0, 'Draws': 1}, 'Black': {'Wins': 0, 'Loses': 3, 'Draws': 1}}
- {'White': {'Wins': 3, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 3, 'Draws': 2}}
- {'White': {'Wins': 4, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 4, 'Draws': 2}}
- {'White': {'Wins': 5, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 5, 'Draws': 2}}
- {'White': {'Wins': 6, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 6, 'Draws': 2}}
- {'White': {'Wins': 7, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 7, 'Draws': 2}}
- {'White': {'Wins': 8, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 8, 'Draws': 2}}
- {'White': {'Wins': 9, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 9, 'Draws': 2}}
- {'White': {'Wins': 10, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 10, 'Draws': 2}}
- {'White': {'Wins': 11, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 11, 'Draws': 2}}
- {'White': {'Wins': 12, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 12, 'Draws': 2}}
- {'White': {'Wins': 13, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 13, 'Draws': 2}}
- {'White': {'Wins': 14, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 14, 'Draws': 2}}
- {'White': {'Wins': 15, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 15, 'Draws': 2}}
- {'White': {'Wins': 16, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 16, 'Draws': 2}}
- {'White': {'Wins': 17, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 17, 'Draws': 2}}
- {'White': {'Wins': 18, 'Loses': 0, 'Draws': 2}, 'Black': {'Wins': 0, 'Loses': 18, 'Draws': 2}}

Almost all games are won by IDS. Minimax did manage 2 draws.

Sahej Bhatia A00766737
COMP 8085

MONTE CARLO SEARCH

MCTS is an algorithm that selects, expands, simulates, and updates the nodes in a tree to identify the best move from a collection of moves. Repetitive use of this strategy leads to the solution and understanding of the game's rules.

Selection :

Nodes on the tree with the best chance of winning are chosen by this method. Consider, for example, the moves that have a good chance of winning. There are three possible outcomes after move 4/6: node 2/3 is most likely to win, followed by 0/1 and half of the way through the game by node 2.

Searching through the tree's current state, we find the targeted node at the branch's terminus. Given that the selected node has the greatest chance of success, the path leading to the solution via that node will be the quickest.

Expanding:

Once we've found the proper node. A node's offspring are created by expanding a selected node and creating many new nodes as a result. In this example, we are only using one child node. Future moves in the game can be made using these child nodes.

Simulating:

Because no one knows which of the nodes in the group has the finest children/leaves, we must simulate. The move that is most likely to succeed and lead to the right answer down a tree.

Back Propagating:

Because of the new nodes as well as the good or negative scores they have in the ecosystem. It is necessary to go back and update the scores of each of the parent nodes one by one. It is possible that the tree's state will alter as a result of the new updated scores.

A total of 20 games were played in the simulation stage. The results are stored in “mcts.model” file.

Note – you can play with the trained model by calling the function – “play_with_mcts”

Thoughts

- Memory requirements were huge
- Needed to run this on a desktop – laptop couldn’t handle it
- Takes 30 seconds or more to respond on each turn
- Im not entirely sure if the implementation was right – this is a trial and error approach

-

Pseudocode

Create root node n and add it to the tree

Make n~ actual node n®

N*= null

F=0

T=0

While(moves available(n))

    While (!isnotleaf(n))

        N=best_child(n)

    N=add_random_chodenode(n)

    Rl = rand(pd, n)

    F,n~ = roll_out(n,rl)

    T++

    backpropagate(n,f)

    if(f>fg)

        n* = n~

        break

    n=n®

return n*, f, t

MONTE CARLO TREE SEARCH AI Self play results [2 hrs]

- {'White': {'Wins': 0, 'Loses': 0, 'Draws': 1}, 'Black': {'Wins': 0, 'Loses': 0, 'Draws': 1}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 1}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 1}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 2}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 2}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 3}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 3}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 4}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 4}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 5}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 5}}
- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 6}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 6}}

- {'White': {'Wins': 0, 'Loses': 1, 'Draws': 7}, 'Black': {'Wins': 1, 'Loses': 0, 'Draws': 7}}
- {'White': {'Wins': 0, 'Loses': 2, 'Draws': 7}, 'Black': {'Wins': 2, 'Loses': 0, 'Draws': 7}}
- {'White': {'Wins': 0, 'Loses': 2, 'Draws': 8}, 'Black': {'Wins': 2, 'Loses': 0, 'Draws': 8}}
- {'White': {'Wins': 0, 'Loses': 2, 'Draws': 9}, 'Black': {'Wins': 2, 'Loses': 0, 'Draws': 9}}

Unlike the previous AI, in MCTS, we see that Black has won 3 times out of 10 times. This can be attributed to the fact that, when most pieces are captured, the search space increases for one player, slightly tilting the bias of winning towards it.