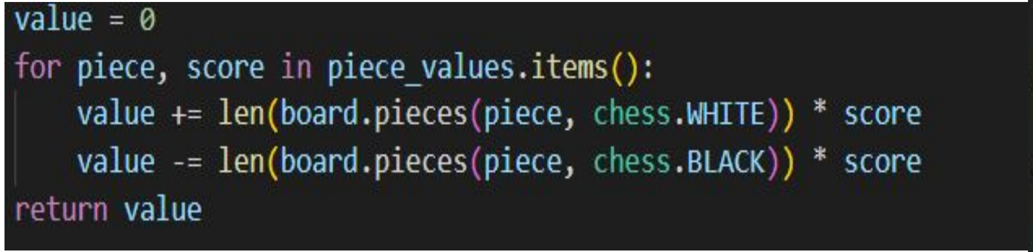
AI Assignment 3

Game Playing

CS22B021 - Nishchith
CS22B020 - Jaswanth

Algorithm - Minimax
Environment - Chess

Function Used:

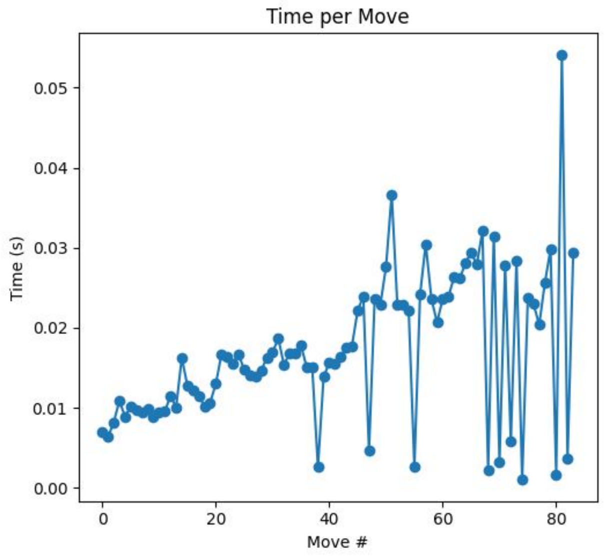


```
value = 0
for piece, score in piece_values.items():
    value += len(board.pieces(piece, chess.WHITE)) * score
    value -= len(board.pieces(piece, chess.BLACK)) * score
return value
```

Search Algo :

```
minimax(self, board, depth, maximizing):
```

Output & Observations :



Total Moves: 84
Total Time: 6.50s
Average Time/Move: 0.08s

Performance Insights:

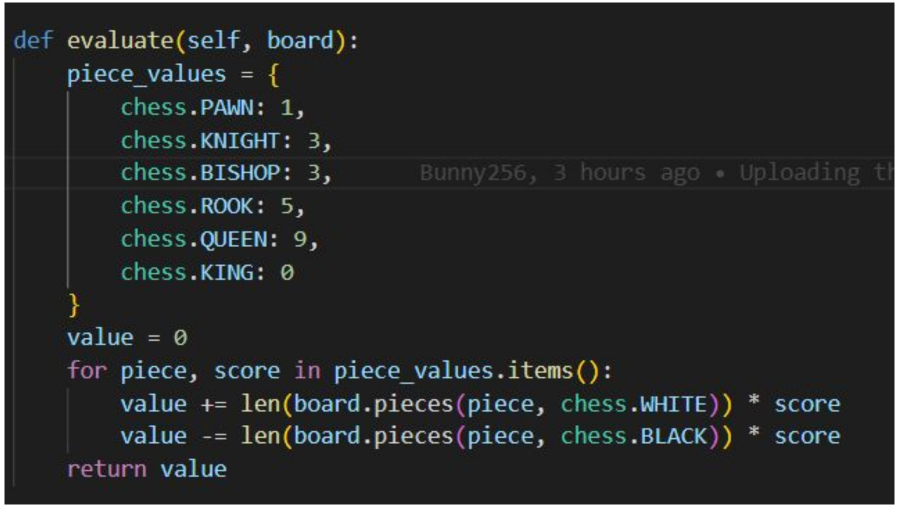
- **Slower** than Alpha-Beta due to **no pruning**.
- Time per move **gradually increases**, showing
- Spikes in time suggest some moves required deeper or broader evaluations.

Algorithm Efficiency:

- **Explores all nodes**, making it **computationally heavier**.
- Same evaluation function as Alpha-Beta, so **higher time is due purely to lack of pruning**.

Algorithm - Alpha
Environment - Chess

Function used:

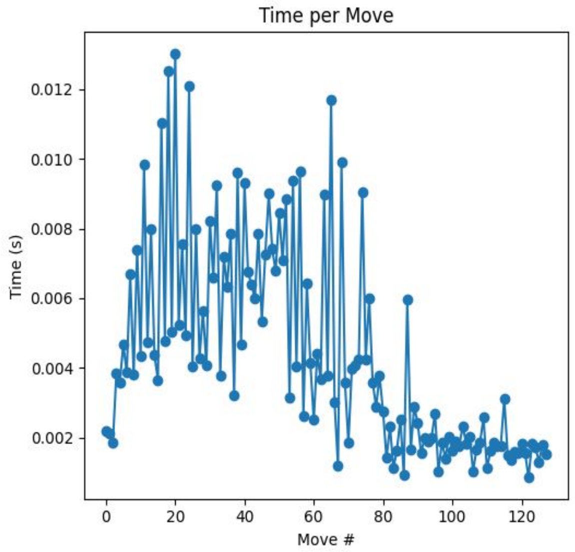


```
def evaluate(self, board):
    piece_values = {
        chess.PAWN: 1,
        chess.KNIGHT: 3,
        chess.BISHOP: 3,       Bunny256, 3 hours ago • Uploading th
        chess.ROOK: 5,
        chess.QUEEN: 9,
        chess.KING: 0
    }
    value = 0
    for piece, score in piece_values.items():
        value += len(board.pieces(piece, chess.WHITE)) * score
        value -= len(board.pieces(piece, chess.BLACK)) * score
    return value
```

Output :

This indicates that:

- The agent is fast (avg. 0.05s/move),
- But likely not very **strategic** due to shallow search depth and simple evaluation.



```
Total Moves: 128
Total Time: 6.66s
Average Time/Move: 0.05s
```

Observations:

- **Shallow Depth (2)** – Limits the ability to plan long-term or avoid traps.
- **Simplistic Evaluation** – Focuses only on material without understanding *position*.
- **Random Tie-Breaking** – When multiple moves have same score, choice is random, which may result in suboptimal paths.