Date:22.08.25

TASK:4

Implementation of Mini-Max algorithm using recursion to search through the Game - tree

AIM

To implement the Minimax algorithm using Python for a two-player turn-based game in order to determine the optimal move for the maximizing player.

ALGORITHM

1.Start at the Root Node of the game tree (representing the current state of the game).

2.Define Depth of the tree (how many moves ahead to look) and whose turn it is:

- MAX tries to maximize the score.
- MIN tries to minimize the score.

3. If current node is a terminal (leaf) node or depth limit is reached:

- Return the score (evaluation of that state).

4. If it's MAX's turn:

- Initialize best to -∞.
- For each child node:
    - Recursively call minimax() for the child node (next depth, MIN's turn).
    - Update best = max(best, value returned).
- Return best.

5. If it's MIN's turn:

- Initialize best to +∞.
- For each child node:
    - Recursively call minimax() for the child node (next depth, MAX's turn).
    - Update best = min(best, value returned).
- Return best.

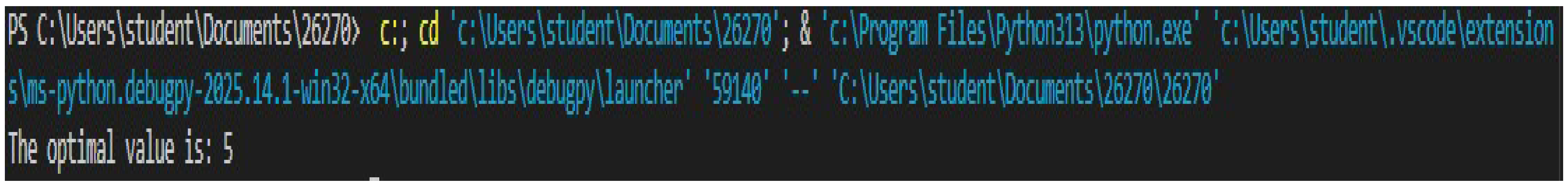
6. Continue recursively until the root node receives the optimal value, representing the best move the MAX player can make.

PROGRAM
Minimax Tree Game AI

```
def minimax(depth, node_index, is_max_player, scores, max_depth):
    # Base case: if we've reached the leaf node
    if depth == max_depth:
        return scores[node_index]

    if is_max_player:
        # MAX player's turn: choose the maximum value
        left = minimax(depth + 1, node_index * 2, False, scores, max_depth)
        right = minimax(depth + 1, node_index * 2 + 1, False, scores, max_depth)
        return max(left, right)
    else:
        # MIN player's turn: choose the minimum value
        left = minimax(depth + 1, node_index * 2, True, scores, max_depth)
        right = minimax(depth + 1, node_index * 2 + 1, True, scores, max_depth)
        return min(left, right)

# Example: Terminal scores of leaf nodes
scores = [3, 5, 6, 9, 1, 2, 0, -1]  # 8 leaf nodes (2^3)

# Tree depth (3 levels: root -> depth 0 to leaf -> depth 3)
```

```
max_depth = 3

# Start minimax from root (depth 0, index 0, MAX player's turn)
optimal_value = minimax(0, 0, True, scores, max_depth)

# Output the result
print("The optimal value is:", optimal_value)
```

OUTPUT



```
PS C:\Users\student\Documents\26270>  c:; cd 'c:\Users\student\Documents\26270'; & 'c:\Program Files\Python313\python.exe' 'c:\Users\student\.vscode\extension
s\ms-python.debugpy-2025.14.1-win32-x64\bundled\libs\debugpy\launcher' '59140' '--' 'C:\Users\student\Documents\26270\26270'
The optimal value is: 5
```

RESULT

Thus, the Minimax algorithm using Python for a two-player turn-based game in order to determine the optimal move for the maximizing player was successfully executed and output was verified.