## **Experiment: Min-Max Algorithm in Python (Game Tree)**

### **AIM**

To implement the **Minimax algorithm** in Python for a two-player turn-based game scenario (zero-sum game) and find the optimal value that a player can secure assuming the opponent also plays optimally.

### **REQUIREMENTS**

* Python 3.x
* Basic understanding of recursion and tree structures
* Familiarity with game theory concepts like maximizing and minimizing players

### **PROCEDURE**

1. Create a recursive minimax function that:  
   * Returns the value of the node if it's a leaf.
   * If it's the maximizing player's turn, returns the **max** value from its children.
   * If it's the minimizing player's turn, returns the **min** value from its children.
2. Use a tree of predefined values at the leaf nodes (like in Tic Tac Toe or any 2-player game).
3. Simulate a game tree using a list and apply Minimax on it.

CODE:

import math

# Minimax algorithm

def minimax(depth, node\_index, is\_maximizing\_player, values, height):

# Base case: leaf node is reached

if depth == height:

return values[node\_index]

if is\_maximizing\_player:

return max(

minimax(depth + 1, node\_index \* 2, False, values, height),

minimax(depth + 1, node\_index \* 2 + 1, False, values, height)

)

else:

return min(

minimax(depth + 1, node\_index \* 2, True, values, height),

minimax(depth + 1, node\_index \* 2 + 1, True, values, height)

)

# Tree height

height = 3

# Terminal leaf nodes

values = [3, 5, 6, 9, 1, 2, 0, -1]

# Start Minimax algorithm

optimal\_value = minimax(0, 0, True, values, height)

print("Optimal value for the maximizer is:", optimal\_value)

OUTPUT:

Optimal value for the maximizer is: 5