# **Minimax algorithm**

### **Aim:**

To implement the **Minimax algorithm** using Python to determine the optimal value for a player in a decision-making game tree, where two players play optimally in a turn-based environment.

### **Procedure:**

1. Take space-separated integer input as leaf node scores and validate it's a power of 2.
2. Calculate the tree depth using log2(length of scores).
3. Define and use the recursive minimax function with base and recursive cases for both players.
4. Call minimax from the root and print the optimal value.

### **Program:**

import math

def minimax(curDepth, nodeIndex, maxTurn, scores, targetDepth):

if curDepth == targetDepth:

return scores[nodeIndex]

if maxTurn:

return max(

minimax(curDepth + 1, nodeIndex \* 2, False, scores, targetDepth),

minimax(curDepth + 1, nodeIndex \* 2 + 1, False, scores, targetDepth)

)

else:

return min(

minimax(curDepth + 1, nodeIndex \* 2, True, scores, targetDepth),

minimax(curDepth + 1, nodeIndex \* 2 + 1, True, scores, targetDepth)

)

# Take input from user

input\_str = input("Enter the scores (space-separated, length must be a power of 2): ")

scores = list(map(int, input\_str.strip().split()))

# Validate if the number of scores is a power of 2

if (len(scores) & (len(scores) - 1)) != 0:

print("Error: Number of scores must be a power of 2.")

else:

treeDepth = int(math.log(len(scores), 2))

print("The optimal value is:", minimax(0, 0, True, scores, treeDepth))

### **Input:**

Enter the scores (space-separated, length must be a power of 2): 3 5 2 9 12 5 23 23

### **Output:**

The optimal value is: 12

**Result:**

The program runs successfully and outputs the correct result based on the Minimax algorithm.