import math

# Define a simple game tree for demonstration

# Example of a tree node structure:

# Node format: (is\_maximizing, value, children\_nodes)

def minimax(node, depth, is\_maximizing):

if not node['children']:

return node['value']

if is\_maximizing:

best\_score = -math.inf

for child in node['children']:

score = minimax(child, depth + 1, False)

best\_score = max(score, best\_score)

return best\_score

else:

best\_score = math.inf

for child in node['children']:

score = minimax(child, depth + 1, True)

best\_score = min(score, best\_score)

return best\_score

# Example game tree

# A node has children if it is not a leaf node

game\_tree = {

'is\_maximizing': True, # Maximizing player starts

'value': None, # Not used in non-leaf nodes

'children': [

{'is\_maximizing': False, 'value': None, 'children': [

{'is\_maximizing': True, 'value': 1, 'children': []},

{'is\_maximizing': True, 'value': 2, 'children': []}

]},

{'is\_maximizing': False, 'value': None, 'children': [

{'is\_maximizing': True, 'value': 3, 'children': []},

{'is\_maximizing': True, 'value': 4, 'children': []}

]}

]

}

# Calculate the best score for the root node

best\_score = minimax(game\_tree, 0, True)

print("Best score for the maximizing player:", best\_score)

Output:

