Untitled0.ipynb - Colaboratory

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | import math  class Node: def \_\_init\_\_(self, value=None):  self.value = value self.children = []  def minimax(node, depth, maximizing\_player): if depth == 0 or not node.children: return node.value  if maximizing\_player: max\_eval = -math.inf for child in node.children:  eval = minimax(child, depth - 1, False) max\_eval = max(max\_eval, eval) return max\_eval else:  min\_eval = math.inf for child in node.children:  eval = minimax(child, depth - 1, True) min\_eval = min(min\_eval, eval) return min\_eval  def alpha\_beta\_pruning(node, depth, alpha, beta, maximizing\_player): if depth == 0 or not node.children: return node.value  if maximizing\_player: max\_eval = -math.inf for child in node.children:  eval = alpha\_beta\_pruning(child, depth - 1, alpha, beta, False) max\_eval = max(max\_eval, eval) alpha = max(alpha, eval) if beta <= alpha:  break return max\_eval else:  min\_eval = math.inf for child in node.children:  eval = alpha\_beta\_pruning(child, depth - 1, alpha, beta, True) min\_eval = min(min\_eval, eval) beta = min(beta, eval) if beta <= alpha:  break return min\_eval  # Example usage if \_\_name\_\_ == "\_\_main\_\_":  root = Node() root.children = [Node(3), Node(6), Node(8)] root.children[0].children = [Node(4), Node(2)] root.children[1].children = [Node(9), Node(1)] root.children[2].children = [Node(5), Node(7)]  print("Minimax result:", minimax(root, 2, True)) print("Alpha-Beta Pruning result:", alpha\_beta\_pruning(root, 2, -math.inf, math.inf, True)) |

 Minimax result: 5

Alpha-Beta Pruning result: 5

2/2