

```

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

```

Minimax result: 5

Alpha-Beta Pruning result: 5 2/2