**EXPERIMENT 7**

|  |  |
| --- | --- |
| **Name: SOBAN WAJUDDIN MARUF** | **Roll No: 21CO58** |
| **Course: AI Lab** | **Sem: VI** |

**AIM :** Implementation of Alpa-Beta Pruning Algorithm

**Code:**

import math

def alphabeta(nodeIndex, depth, alpha, beta, isMaximizing, scores):

# Base case: if depth is 0 or there's only one node left, return its value

if depth == 0 or len(scores) == 1:

return scores[0]

if isMaximizing:

# If it's the maximizing player's turn, return the maximum value

bestValue = -math.inf

for i in range(len(scores) // 2):

# Recursively call the function for the left child and right child of the current

node

value = alphabeta(nodeIndex \* 2 + i, depth - 1, alpha, beta, False, scores)

# Update the bestValue with the maximum value found so far

bestValue = max(bestValue, value)

# Update alpha with the bestValue found so far

alpha = max(alpha, bestValue)

# If alpha >= beta, stop exploring this branch of the tree

if alpha >= beta:

break

# Display the final value of alpha

print("Value of alpha at node", nodeIndex, ":", alpha)

# Return the bestValue found for this level of the tree

return bestValue

else:

# If it's the minimizing player's turn, return the minimum value

bestValue = math.inf

for i in range(len(scores) // 2, len(scores)):

# Recursively call the function for the left child and right child of the current

node

value = alphabeta(nodeIndex \* 2 + i, depth - 1, alpha, beta, True, scores)

# Update the bestValue with the minimum value found so far

bestValue = min(bestValue, value)

# Update beta with the bestValue found so far

beta = min(beta, bestValue)

# If alpha >= beta, stop exploring this branch of the tree

if alpha >= beta:

break

# Display the final value of beta

print("Value of beta at node", nodeIndex, ":", beta)

# Return the bestValue found for this level of the tree

return bestValue

scores = [9, 5, 2, 4, 12, 3, 23, 25]

print("The optimal value is:", alphabeta(0, 3, -math.inf, math.inf, True, scores))

**---------------------------------------------------------------**

**Output:**

Value of alpha at node 4 : 9

Value of alpha at node 5 : 9

Value of alpha at node 6 : 9

Value of alpha at node 7 : 9

Value of beta at node 0 : 9

Value of alpha at node 6 : 9

Value of beta at node 1 : 9

Value of alpha at node 8 : 9

Value of beta at node 2 : 9

Value of alpha at node 10 : 9

Value of beta at node 3 : 9

Value of alpha at node 0 : 9

The optimal value is: 9

**---------------------------------------------------------------**