B A Saran
22011102012
B.Tech CSE(IoT)-A

ARTIFICIAL INTELLIGENCE LAB

Alpha-Beta Pruning ALGORITHM

AIM: To implement the Alpha Beta Pruning Algorithm.

ALGORITHM:

1. **Define Function**: Create a function `minimax(curDepth, nodeIndex, maxTurn, scores, targetDepth, alpha, beta)` to evaluate the game tree recursively with alpha and beta values for pruning.

2. **Base Case**: If `curDepth` equals `targetDepth`, return the score at `scores[nodeIndex]` (the leaf node value).

3. **Maximizing Player**: If `maxTurn` is `True`, initialize `maxEval` to negative infinity and recursively calculate the maximum value between the left (`nodeIndex * 2`) and right (`nodeIndex * 2 + 1`) child nodes. Update `alpha` and check for pruning conditions.

4. **Minimizing Player**: If `maxTurn` is `False`, initialize `minEval` to positive infinity and recursively calculate the minimum value between the left and right child nodes. Update `beta` and check for pruning conditions.

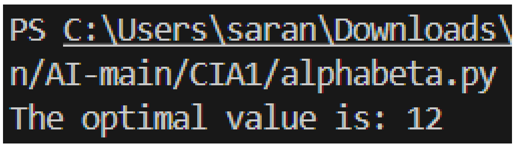
5. **Execute**: Call `minimax(0, 0, True, scores, targetDepth, alpha, beta)` to start from the root node and print the optimal value.

CODE:

```
import math


def minimax(curDepth, nodeIndex, maxTurn, scores, targetDepth, alpha, beta):
    if curDepth == targetDepth:
        return scores[nodeIndex]
    if maxTurn:
        maxEval = -math.inf
        left_eval = minimax(curDepth + 1, nodeIndex * 2, False, scores, targetDepth, alpha, beta)
        maxEval = max(maxEval, left_eval)
        alpha = max(alpha, maxEval)

        if beta <= alpha:
            return maxEval

        right_eval = minimax(curDepth + 1, nodeIndex * 2 + 1, False, scores, targetDepth, alpha, beta)
        maxEval = max(maxEval, right_eval)
        alpha = max(alpha, maxEval)

        return maxEval
    else:
        minEval = math.inf
        left_eval = minimax(curDepth + 1, nodeIndex * 2, True, scores, targetDepth, alpha, beta)
        minEval = min(minEval, left_eval)
        beta = min(beta, minEval)

        if beta <= alpha:
            return minEval

        right_eval = minimax(curDepth + 1, nodeIndex * 2 + 1, True, scores, targetDepth, alpha, beta)
        minEval = min(minEval, right_eval)
        beta = min(beta, minEval)

        return minEval

# Different scores array
scores = [8, 7, 6, 5, 12, 10, 14, 3]
treeDepth = int(math.log(len(scores), 2))
alpha = -math.inf
```

```
beta = math.inf

print("The optimal value is: ", end="")
print(minimax(0, 0, True, scores, treeDepth, alpha, beta))
```

OUTPUT:



```
PS C:\Users\saran\Downloads\
n/AI-main/CIA1/alphabeta.py
The optimal value is: 12
```

RESULT: The Alpha Beta Pruning algorithm is implemented.