AI EX 6

MINMAX ALGORITHM

AIM: Developing a mini max algorithm for real world problems.

PROBLEM STATEMENT: Find the optimal value in the given tree of integer values in the most optimal way possible under the time complexity O(B^D).

ALGORITHM:

1. Start traversing the given tree in top to bottom manner.
2. If node is a leaf node then return the value of the node.
3. If isMaximizingPlayer exist then bestVal = -INFINITY
4. For each child node, value = minimax(node, depth+1, false, alpha, beta)
5. bestVal = max(bestVal, value) and alpha = max(alpha, bestVal)
6. If beta <= alpha then stop traversing and return bestVal
7. Else, bestVal = +INFINITY
8. For each child node, value = minimax(node, depth+1, true, alpha, beta)
9. bestVal = min(bestVal, value) and beta = min(beta, bestVal)
10. If beta <= alpha the stop traversing and return bestVal

<u>OPTIMIZATION TECHNIQUE:</u>

Alpha-Beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithms. It reduces the computation time by a huge factor.

This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available.

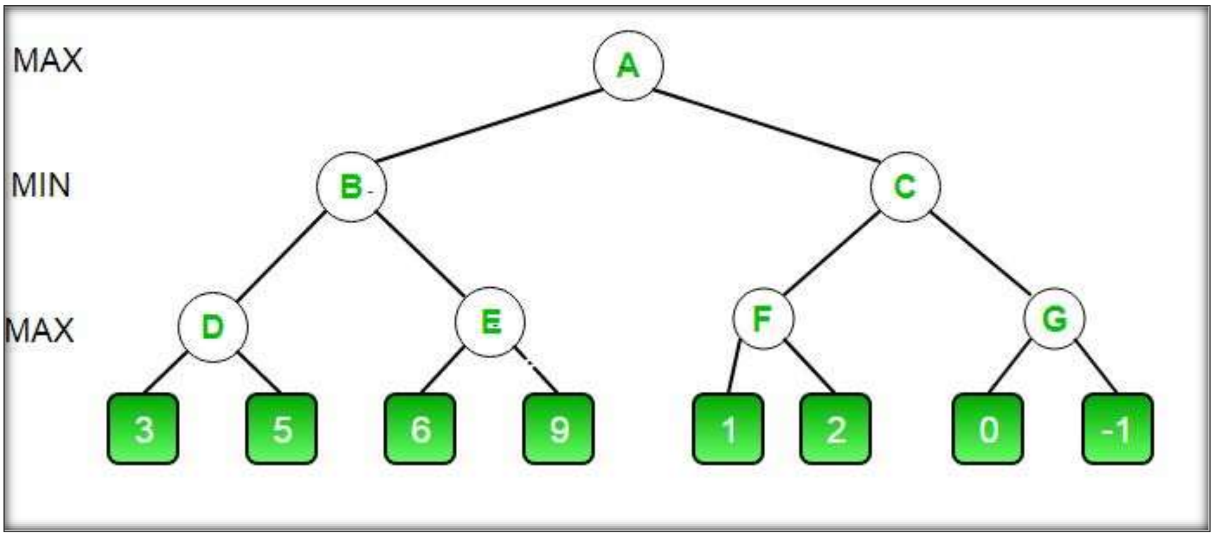
It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

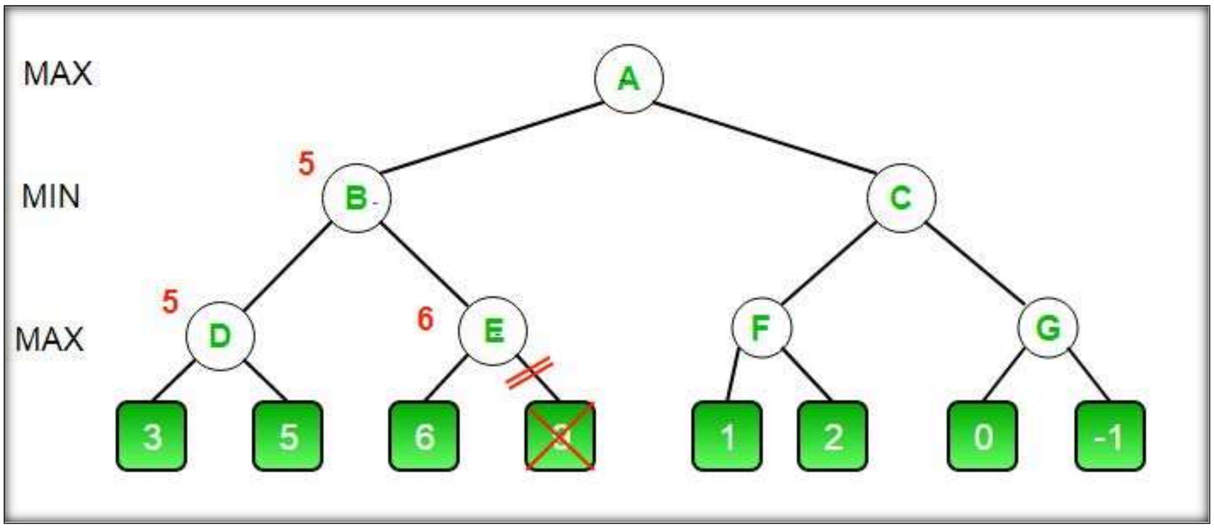
Let’s define the parameters alpha and beta.

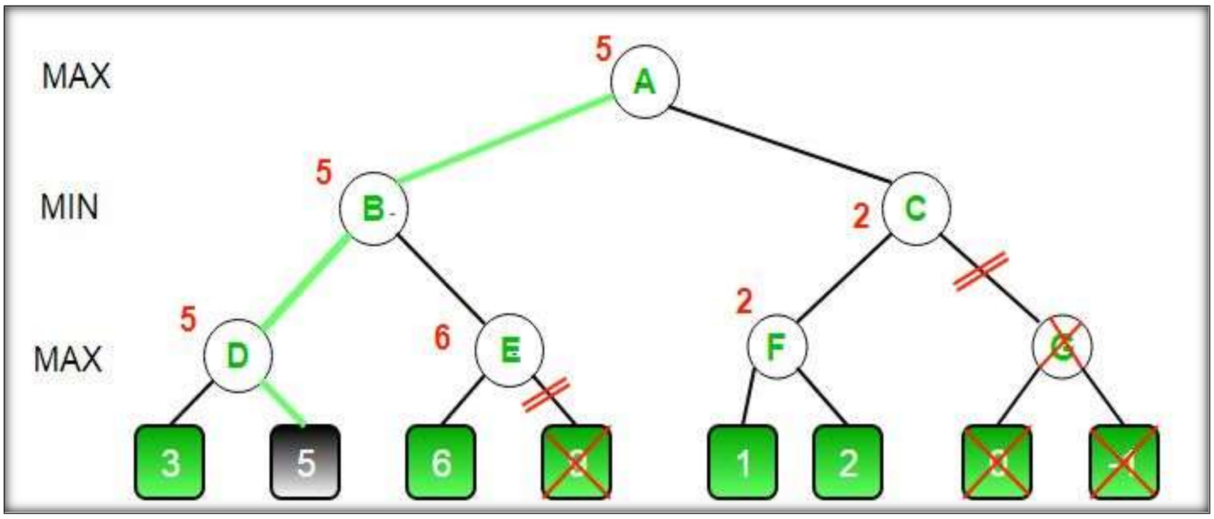
Alpha is the best value that the maximizer currently can guarantee at that level or above.

Beta is the best value that the minimizer currently can guarantee at that level or above.

<u>GIVEN PROBLEM:</u>





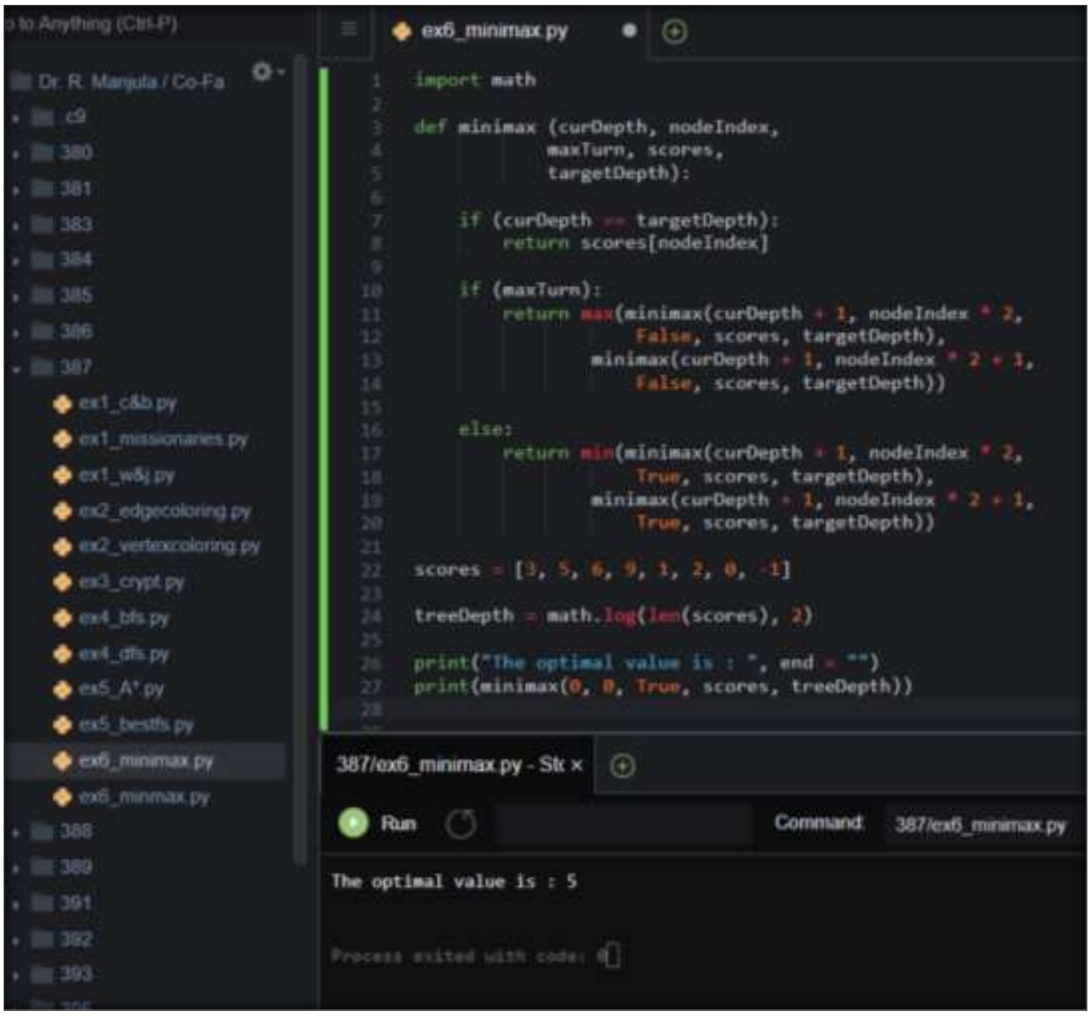


CODE:

```
import math

def minimax (curDepth, nodeIndex,
                    maxTurn, scores,
                    targetDepth):
```

```
    if (curDepth == targetDepth):
        return scores[nodeIndex]

    if (maxTurn):
        return max(minimax(curDepth + 1, nodeIndex * 2,
                        False, scores, targetDepth),
                    minimax(curDepth + 1, nodeIndex * 2 + 1,
                        False, scores, targetDepth))

    else:
        return min(minimax(curDepth + 1, nodeIndex * 2,
                        True, scores, targetDepth),
                    minimax(curDepth + 1, nodeIndex * 2 + 1,
                        True, scores, targetDepth))

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

treeDepth = math.log(len(scores), 2)

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

OUTPUT:



```python
import math

def minimax (curDepth, nodeIndex,
             maxTurn, scores,
             targetDepth):

    if (curDepth == targetDepth):
        return scores[nodeIndex]

    if (maxTurn):
        return max(minimax(curDepth + 1, nodeIndex * 2,
                   False, scores, targetDepth),
                minimax(curDepth + 1, nodeIndex * 2 + 1,
                   False, scores, targetDepth))

    else:
        return min(minimax(curDepth + 1, nodeIndex * 2,
                   True, scores, targetDepth),
                minimax(curDepth + 1, nodeIndex * 2 + 1,
                   True, scores, targetDepth))

scores = [3, 5, 6, 9, 1, 2, 0, -1]

treeDepth = math.log(len(scores), 2)

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

```
The optimal value is : 5

Process exited with code: 0
```

RESULT: The Optimal value of the given tree successfully found using Minimax Algorithm.

CHITRALEKHA.CH

RA1911003010387