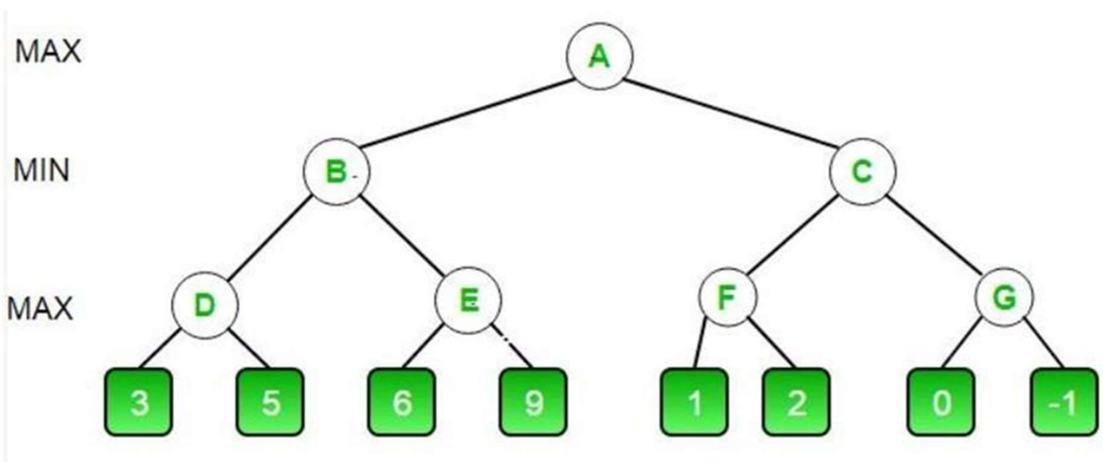
Ex No.6: *Implementation of Mini-Max Algorithm for an Application*

Mini-max Algorithm

The min max algorithm in AI, popularly known as the minimax, is a backtracking algorithm used in decision making, game theory and artificial intelligence (AI). It is used to find the optimal move for a player, assuming that the opponent is also playing optimally. Popular two-player computer or online games like Chess, Tic-Tac-Toe, Checkers, Go, etc. use this algorithm.
A backtracking algorithm is used to find a solution to computational problems in such a way that a candidate is incrementally built towards a solution, one step at a time. And the candidate that fails to complete a solution is immediately abandoned.

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

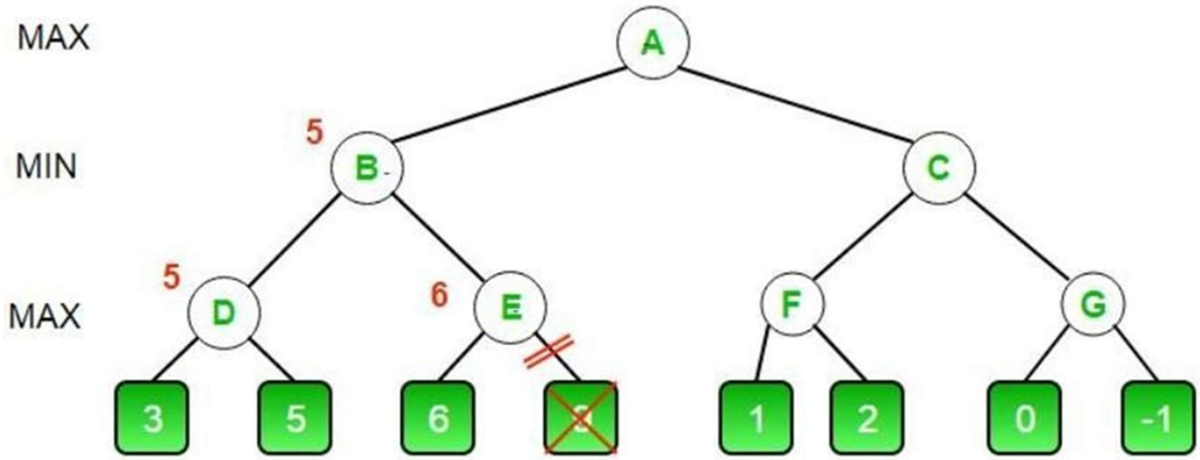
Optimization Technique:

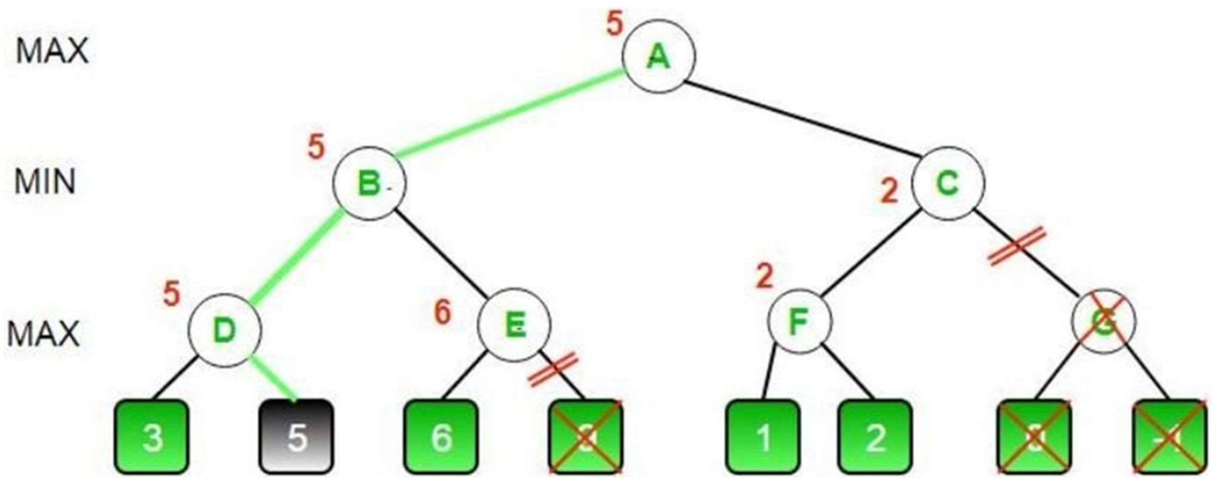
Alpha-Beta pruning is not actually a new algorithm, rather an optimization technique for minimax algorithms. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be 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.





Tool: AWS Cloud9 ide and Python 3.9.0

Code :

```
MAX, MIN = 1000, -1000
def minimax(depth, nodeIndex, maximizingPlayer,
                              values, alpha, beta):

          if depth == 3:
                    return values[nodeIndex]
```

```
        if maximizingPlayer:

            best = MIN

            for i in range(0, 2):

                val = minimax(depth + 1, nodeIndex * 2 + i,
                                        False, values, alpha, beta)
                best = max(best, val)
                alpha = max(alpha, best)

                if beta <= alpha:
                    break

            return best




        else:
            best = MAX
            for i in range(0, 2):

                val = minimax(depth + 1, nodeIndex * 2 + i,
                                            True, values, alpha, beta)
                best = min(best, val)
                beta = min(beta, best)
                if beta <= alpha:
                    break

            return best

    if __name__ == "__main__":

        values = []
        for i in range(0, 8):

            x = int(input(f"Enter Value {i}  : "))
            values.append(x)

        print ("The optimal value is :", minimax(0, 0, True, values, MIN, MAX))
```

Output:

Exp6.py

```python
MAX, MIN = 1000, -1000
def minimax(depth, nodeIndex, maximizingPlayer,
            values, alpha, beta):

    if depth == 3:
        return values[nodeIndex]

    if maximizingPlayer:

        best = MIN

        for i in range(0, 2):

            val = minimax(depth + 1, nodeIndex * 2 + i,
                          False, values, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)

            if beta <= alpha:
                break

        return best

    else:
        best = MAX
        for i in range(0, 2):
```

```
Enter Value 0  : 6
Enter Value 1  : 8
Enter Value 2  : 2
Enter Value 3  : 3
Enter Value 4  : 1
Enter Value 5  : 0
Enter Value 6  : -1
Enter Value 7  : 4
The optimal value is : 3


Process exited with code: 0
```

Result:

The Optimal value of the given tree successfully found using Minimax Algorithm with Alpha Beta Pruning in time complexity O(B^D).