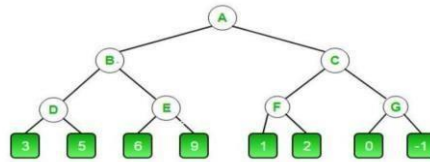


TASK:4

Implementation of **Mini-Max algorithm** using recursion to search through the Game - tree using python by applying following constraints.

Aim: To create a program for searching problem using Mini-max algorithm with Alpha-Beta pruning approach.



Algorithm:

Step 1: At the first step the, Max player will start first move from node A where $\alpha = -\text{infinity}$ and $\beta = +\text{infinity}$, these values of alpha and beta passed down to node B. Node B transmitting the identical value to its off spring D.

Step2: As Max's turn at Node D approaches, the value of α will be decided. when the value of α is compared to 3 then 5 the value at node D is $\max(3,5) = 5$. Hence the node value is also 5

Step 3: The algorithm returns to node B, where the value of beta will change since this a turn of min **Step**

4: Max will take over at node E and change alpha's value.

Step 5: We know traverse the tree backward, from node B to node A

Step 6: As a result, in this case, the ideal value for the maximizer is 5.

Program:

```
# Initial values of Alpha and Beta
```

```
MAX, MIN = 1000, -1000
```

```
# Minimax function with alpha-beta pruning
```

```
def minimax(depth, nodeIndex, maximizingPlayer, values, alpha, beta):
```

```
    # Terminating condition: leaf node is reached
```

```
    if depth == 3:
```

```
        return values[nodeIndex]
```

```
    if maximizingPlayer:
```

```
        best = MIN
```

```
        # Recur for left and right children
```

```
        for i in range(2):
```

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

```
            best = max(best, val)
```

```
            alpha = max(alpha, best)
```

```
        # Alpha Beta Pruning
```

```
        if beta <= alpha:
```

```
            break
```

```
        return best
```

```
    else:
```

```
        best = MAX
```

```

# Recur for left and right children
for i in range(2):
    val = minimax(depth + 1, nodeIndex * 2 + i, True, values, alpha, beta)
    best = min(best, val)
    beta = min(beta, best)
# Alpha Beta Pruning
if beta <= alpha:
    break
return best

# Driver Code
if __name__ == "__main__":
    values = [5, 7, 3, 2, 1, 2, 0, -3]
    print("The optimal value is:", minimax(0, 0, True, values, MIN, MAX))

```

Sample output :

```

===== RESTART: C:/Users/mahes/VTU26520.py
The optimal value is: 3
>>> |

```

Result:

Thus creating a program for searching problem using Mini-max algorithm with Alpha-Beta pruning approach was successfully executed and output was verified.