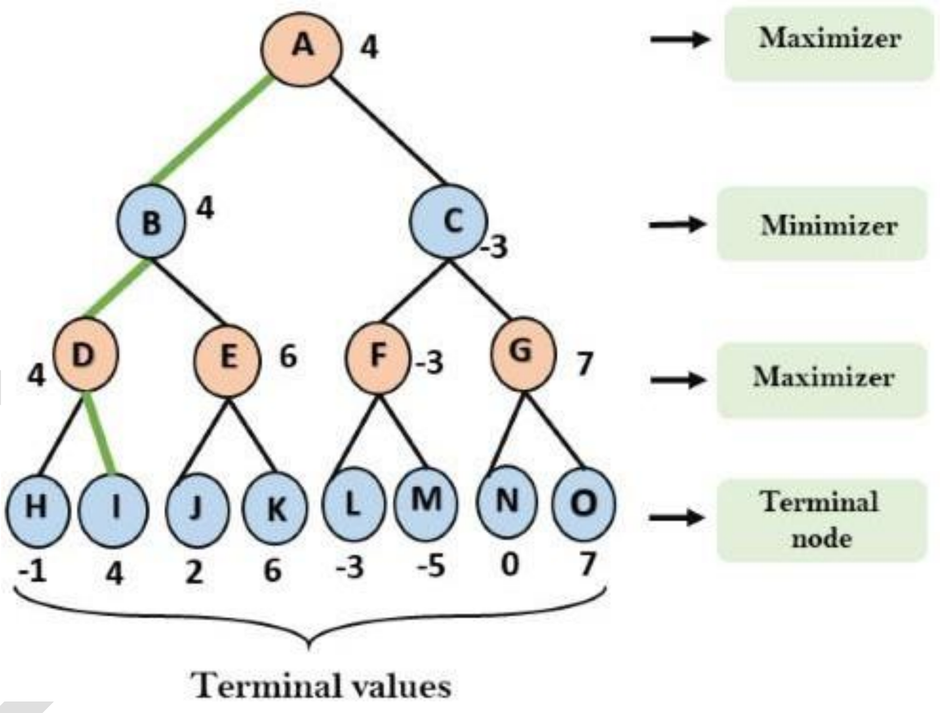
EX.NO: 4 **DATE: 30 - 08 - 2024**

MINIMAX ALGORITHM

AIM:

To implement the Minimax Algorithm for a two-player game, with Maximizer maximizing the score and Minimizer minimizing it through DFS evaluation.

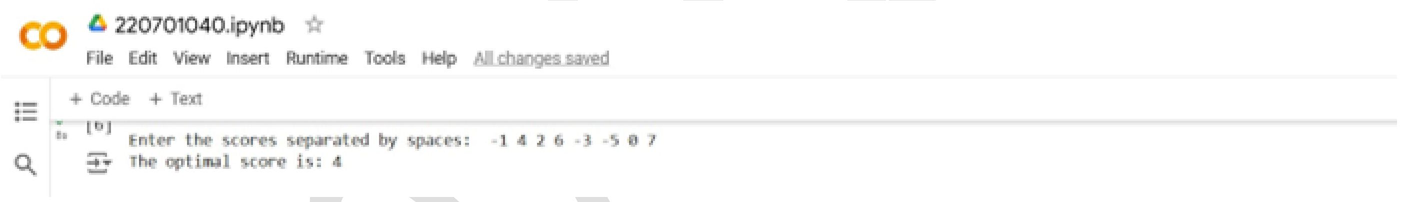
- A simple example can be used to explain how the minimax algorithm works. We've included an example of a game-tree below, which represents a two-player game.
- There are two players in this scenario, one named Maximizer and the other named Minimizer.
- Maximizer will strive for the highest possible score, while Minimizer will strive for the lowest possible score.
- Because this algorithm uses DFS, we must go all the way through the leaves to reach the terminal nodes in this game-tree.
- The terminal values are given at the terminal node, so we'll compare them and retrace the tree till we reach the original state.



PROGRAM:

```
import math
def minimax(depth, node_index, is_maximizer, scores, height):
    if depth == height:
        return scores[node_index]
    if is_maximizer:
        return max(minimax(depth + 1, node_index * 2, False, scores, height),
                   minimax(depth + 1, node_index * 2 + 1, False, scores, height))
    else:
        return min(minimax(depth + 1, node_index * 2, True, scores, height),
                   minimax(depth + 1, node_index * 2 + 1, True, scores, height))
def calculate_tree_height(num_leaves):
    return math.ceil(math.log2(num_leaves))
scores = list(map(int, input("Enter the scores separated by spaces: ").split()))
tree_height = calculate_tree_height(len(scores))
optimal_score = minimax(0, 0, True, scores, tree_height)
print(f"The optimal score is: {optimal_score}")
```

OUTPUT:



```
Enter the scores separated by spaces:  -1 4 2 6 -3 -5 0 7
The optimal score is: 4
```

RESULT:

Thus ,the Minimax Algorithm successfully determines the optimal moves for both players by evaluating the game-tree and selecting the best possible scores for Maximizer and Minimizer.