1. Write a program to implement a Minimax decision-making algorithm, typically used in a turn-based, two player games. The goal of the algorithm is to find the optimal next move.
   USN - 1NT18CS133

Algorithm: • Construct the complete game tree
• Evaluate scores for leaves using the evaluation function
• Back-up scores from leaves to root, considering the player type:
o For max player, select the child with the maximum score
o For min player, select the child with the minimum score
• At the root node, choose the node with max value and perform the corresponding move

In [1]:
```python
import math
import random
#minimax class
def minimax (currentDepth, nodeIndex, maxTurn, score,  treeDepth):
    # base case : treeDepth reached
    if (currentDepth == treeDepth):
        return score[nodeIndex]

    if (maxTurn):
        return max(minimax(currentDepth + 1, nodeIndex * 2, False, score, treeDepth),
        minimax(currentDepth + 1, nodeIndex * 2 + 1,False, score, treeDepth))

    else:
        return min(minimax(currentDepth + 1, nodeIndex * 2, True, score, treeDepth),
        minimax(currentDepth + 1, nodeIndex * 2 + 1,True, score, treeDepth))

# Driver code
score = random.sample(range(1, 50), 4)
print(str(score))
treeDepth = math.log(len(score), 2)

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

```
[42, 29, 39, 15]
The optimal value is : 29
```

In [ ]: