**Manual of code**

**Lab 08**

**The minimax Function:**

This is the core function that implements the Minimax algorithm. It works recursively to evaluate all possible paths in the tree.

****

* cur\_depth: This keeps track of how deep we are in the tree. It starts at 0 (the root) and increases as we go deeper.
* node\_index: This is the index of the current node in the scores array. It helps us navigate the tree.
* max\_turn: A boolean value (True or False) that tells us whether it’s the maximizer’s turn (True) or the minimizer’s turn (False).
* scores: This is the list of values at the leaf nodes of the tree. These are the final scores we’re trying to evaluate.
* target\_depth: This is the maximum depth of the tree. When cur\_depth equals target\_depth, we stop recursing.

**2. Base Case:**

****

If we’ve reached the maximum depth of the tree (i.e., we’re at a leaf node), we simply return the value of the current node from the scores array.

**3. Maximizer’s Turn:**

****

If it’s the maximizer’s turn (max\_turn == True), the function recursively calls itself for the left and right children of the current node.

The maximizer chooses the maximum of the two values returned by the recursive calls.

**4. Minimizer’s Turn:**

****

If it’s the minimizer’s turn (max\_turn == False), the function recursively calls itself for the left and right children of the current node.

The minimizer chooses the minimum of the two values returned by therecursive calls.

**5. Input Scores:**

****

This is the list of values at the leaf nodes of the binary tree. The tree is assumed to be a perfect binary tree, meaning every non-leaf node has exactly two children.

**6. Calculate Tree Depth:**

**python**

**Copy**

**import math**

****

* **The depth of the tree is calculated using math.log2(len(scores)). Since the tree is a perfect binary tree, the depth is the logarithm (base 2) of the number of leaf nodes.**
* **For example, if there are 8 leaf nodes, the depth is log2(8) = 3.**

**7. Run the Minimax Algorithm:**

****

* **The minimax function is called with the following arguments:**
  + **cur\_depth = 0: Start at the root of the tree.**
  + **node\_index = 0: Start at the first node in the scores array.**
  + **max\_turn = True: The maximizer starts first.**
  + **scores: The list of leaf node values.**
  + **tree\_depth: The calculated depth of the tree.**

**8. Print the Result:**

****

* **The program prints the optimal value computed by the Minimax algorithm.**

**Example Walkthrough:**

**Input:**

**scores = [8, 3, 6, 1, 9, 2, 5, 7]**

**Tree Structure:**

**Max**

**/ \**

**Min Min**

**/ \ / \**

**8 3 6 1**

**/ \ / \ / \ / \**

**9 2 5 7 9 2 5 7**

**How It Works:**

1. **The maximizer starts at the root and tries to maximize the value.**
2. **The minimizer responds by trying to minimize the value.**
3. **The algorithm evaluates all possible paths from the root to the leaf nodes.**
4. **The final result is the optimal value for the maximizer, assuming the minimizer plays optimally.**

**Output:**

**Copy**

**The optimal value is: 7**