**Minimax Algorithm – Step-by-Step Explanation**

**1. Purpose of the Algorithm**

* The minimax algorithm is used in two-player games where one player tries to **maximize** the score and the other tries to **minimize** it.
* It helps in making the **best possible decision** assuming the opponent also plays perfectly.

**2. Game Tree Concept**

* The algorithm works like a **tree structure** where each node represents a game state or move.
* Each level in the tree represents a **turn**:
  + One for the **Maximizer (player who wants the highest score)**
  + The next for the **Minimizer (player who wants the lowest score)**

**3. Leaf Nodes (End of the Game)**

* The bottom level of the tree contains the **final scores** of all possible game outcomes.
* These are called **leaf nodes** because they have no further moves.

**4. Traversing the Tree**

* The algorithm starts from the **top** of the tree (the first move) and goes down to the **bottom** level (the final outcomes).
* It checks **every possible move** and outcome recursively.

**5. Decision-Making Process**

* When the algorithm reaches the bottom, it returns the score for that path — this means no more moves can be made.
* As it moves back up the tree:
  + The **Maximizer** chooses the **higher** value between two moves.
  + The **Minimizer** chooses the **lower** value between two moves.

**6. Moving Between Nodes**

* Each node in the tree has two possible next moves — a **left child** and a **right child**.
* This helps the algorithm check both possible directions or choices from every game state.

**7. Role of Depth**

* The **depth** of the tree shows how many rounds or levels the game has before reaching the end.
* Each level represents a turn by one of the players.

**8. The Backtracking Step**

* After reaching the end (leaf nodes), the algorithm moves back up, comparing values.
* It applies the “max” or “min” rule at each level depending on whose turn it is.
* This process continues until the topmost node (the root) gets its final value.

**9. Finding the Optimal Value**

* The value obtained at the top of the tree is called the **optimal value**.
* It represents the **best guaranteed score** for the first player if both play their best moves.

**10. Final Result**

* In this example, the algorithm finds the **optimal value as 9**.
* This means that if both players play perfectly, the Maximizer can always end up with a score of **9**, no matter what moves the Minimizer makes.