Semester -I A.Y.2025-26 Sub.: - Artificial Intelligence Lab Class: SE

---

1. Aim

Understand and implement the basic Minimax algorithm for two-player deterministic, zero-sum games and apply it to a simple game (Tic-Tac-Toe). Evaluate the algorithm's behavior and discuss limitations and improvements.

2. Learning Objectives

By the end of the lab the student should be able to:

- Explain the minimax decision rule and game trees.
- Implement minimax using recursion to choose optimal moves for perfect-play agents.
- Apply minimax to Tic-Tac-Toe and verify correct play.
- Analyze complexity and discuss pruning (alpha-beta) and depth-limiting.

3. Background / Theory

Two-player, deterministic games with perfect information (e.g., Tic-Tac-Toe, Chess at a conceptual level) can be modeled as a game tree. Each node represents a game state and edges represent legal moves. Players alternate turns; one is called MAX (tries to maximize utility) and the other MIN (tries to minimize utility). In a zero-sum game, one player's gain is the other's loss.

Minimax idea: Starting from the current state, explore possible moves to terminal states and evaluate each terminal state with a utility function (win = +1, draw = 0, loss = -1 for MAX). Propagate utilities upward: at MAX nodes choose the child with maximum utility; at MIN nodes choose the child with minimum utility. The root decision yields the best move assuming perfect play by both.

Complexity: Time complexity is O(b^d) where b = branching factor, d = search depth. Tic-Tac-Toe is small enough to be solved fully. Larger games require depth-limiting and heuristics.

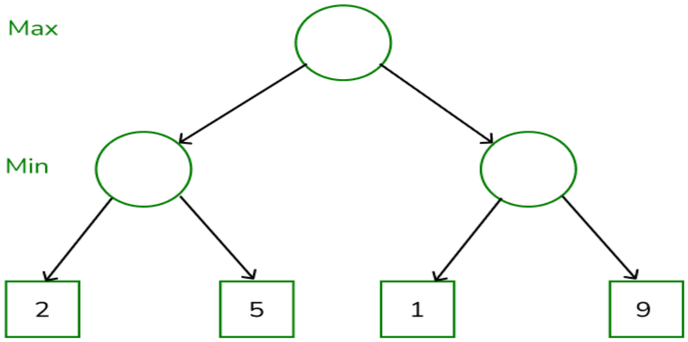
---

4. Algorithm

```
minimax (node n, depth d, player p)
1. If depth = 0 then
        return value(node)
2. If player = “MAX”
        set α = - ∞
        for every child of node
                value = minimax (child,depth-1,’MIN’)
                α = max(α, value)
        return (α )
else
        set α = +∞
        for every child of node
                value = minimax (child,depth-1,’MAX’)
                α = min(α, value)
        return (α )
```

5. Python Implementation

```
# Example graph
```



6. Sample Output

7. Observations

8. Conclusion