Hossam Nabil Elsabbagh 202202228

Phase II: Application of Adversarial Search Algorithms to a Game Environment

1. Introduction

This phase focuses on applying adversarial search algorithms—**Minimax** and **Alpha-Beta Pruning**—to a suitable two-player deterministic game environment. The objective is to model the game, implement both algorithms, design an evaluation function, and compare their performance in terms of efficiency and optimality.

The selected game for this phase is the **Nim Game**, a classical combinatorial game that is widely used to demonstrate adversarial search techniques.

---

2. Game Selection & Modeling

The Nim Game satisfies all the required criteria:

- **Two-player and turn-based:** Players alternate turns.
- **Zero-sum:** One player's gain is the other player's loss.
- **Multiple possible moves per turn:** Each player can remove different numbers of stones from different piles, resulting in a branching factor greater than 2.
- **Manageable but non-trivial game tree depth:** The game tree is deep enough to demonstrate search complexity while remaining computationally feasible.
- **Not Tic-Tac-Toe:** A different adversarial game is used as required.

---

3. Game Description

a) Game State Representation

The game state is represented using:

- **Piles:** A list of integers, where each integer represents the number of stones in a pile (e.g., [3, 4, 5]).
- **Player Turn:** An indicator specifying whose turn it is:
    - 1 for the Max player (AI)
    - -1 for the Min player (opponent)

No additional features such as the last move are stored, since the pile configuration fully represents the game state.

---

b) Initial State

The initial state of the game is defined as:

- Piles: [3, 4, 5]
- Starting player: Max player (AI)

This configuration provides a balanced and non-trivial starting point.

---

c) Actions / Moves

At any state, a player can:

- Select any pile that contains at least one stone
- Remove one or more stones from that pile

This results in multiple valid actions at each turn, ensuring a high branching factor.

---

d) Transition Function

The transition function generates a new state by:

1. Copying the current pile configuration
2. Applying the selected move (removing stones from a chosen pile)
3. Switching the turn to the opposing player

---

e) Terminal States

A terminal state occurs when all piles contain zero stones.

- **Win condition:** The player who makes the last valid move wins.
- **Loss condition:** The player who cannot make a move loses.
- **Draw condition:** Draws are not possible in the Nim Game.

---

f) Game Tree Complexity Estimate

- **Approximate branching factor:** Between 3 and 10, depending on pile sizes
- **Estimated depth to terminal state:** Typically, between 6 and 12 moves
- **Suitability for Minimax & Alpha-Beta:**
    - Deterministic
    - Zero-sum
    - Alternating turns
    - Clear terminal conditions
    - Allows effective pruning

---

4. Modeling Assumptions

To simplify the implementation and analysis, the following assumptions were made:

- A fixed **maximum search depth** of 6 is used
- Players strictly alternate turns
- No symmetry pruning is applied to keep the implementation simple
- Initial pile sizes are restricted to avoid an excessively large state space

---

5. Algorithms Implementation

Minimax Algorithm

The Minimax algorithm explores the game tree by assuming that:

- The Max player attempts to maximize the evaluation score
- The Min player attempts to minimize the evaluation score

All possible moves are explored up to the maximum depth or until a terminal state is reached.

Alpha-Beta Pruning

Alpha-Beta Pruning is an optimization of Minimax that eliminates branches that cannot influence the final decision. It uses two bounds:

- **Alpha:** Best value found so far for the Max player
- **Beta:** Best value found so far for the Min player

When alpha ≥ beta, further exploration of that branch is stopped.

---

6. Evaluation Function

The evaluation function works as follows:

- If the state is terminal, it returns:
    - +1 for a win
    - -1 for a loss
- If the state is non-terminal, it returns the negative sum of all stones in the piles

These heuristic favors state that they are closer to the end of the game with fewer stones remaining.

---

7. Performance Comparison

Both algorithms were executed using the same maximum depth limit.

Criterion	**Minimax**	**Alpha-Beta**
Time taken	Higher	Lower
Nodes expanded	Large	Significantly fewer
Depth reached	Same (depth = 6)	Same (depth = 6)
Optimality of moves	Optimal	Optimal
Efficiency	Low	High

---

8. Discussion

Effect of Pruning

Alpha-Beta pruning significantly improves performance by avoiding the exploration of branches that cannot affect the final decision, reducing computation time and node expansion.

When Pruning is Most Effective

Pruning is most effective when:

- Good moves are explored early
- The branching factor is high
- The search depth is large

Role of Heuristics

When the search is depth-limited, heuristic evaluation functions allow the algorithm to estimate the desirability of non-terminal states, enabling strong decision-making even without reaching terminal states.

---

9. Visualization

- **Partial Game Tree:** A subset of the game tree is displayed to illustrate explored nodes.
- **Pruned Branches:** Branches removed by Alpha-Beta pruning are highlighted conceptually where alpha ≥ beta.
- **Board State Snapshots:** Sequential pile configurations show how the AI progresses through the game.
- **Decision Comparison:** Diagrams demonstrate that Minimax explores all branches, while Alpha-Beta reaches the same decision more efficiently.

---

10. Conclusion

This phase demonstrated the application of adversarial search algorithms to a deterministic two-player game. While Minimax guarantees optimal decisions, Alpha-Beta pruning significantly improves efficiency without sacrificing optimality. The Nim Game proved to be an effective environment for analyzing and comparing these algorithms.