AI Lab Task: The Nim Game AI with Minimax

Take or Lose: Solving the Nim Game with Minimax Strategy

Game Description

The Nim Game is a 2-player mathematical game of strategy.

- You are given 3 piles of sticks, e.g., [3, 4, 5].
- Players take turns. On a turn, a player chooses one pile and removes one or more sticks.
- The player who removes the last stick wins. That means the player who makes the piles all empty during their turn is the winner.

Detailed Task Requirements

1. **Game Setup:**
   - Three piles initialized as [3, 4, 5].
   - Human plays first by default.
2. **Valid Moves:**
   - Choose which pile (1, 2, or 3) and how many sticks to remove (minimum 1).
3. **Game Loop:**
   - Alternate turns between human and AI.
   - After each move, show updated pile states.
   - If all piles are empty → last player to move wins.
4. **Minimax AI:**
   - Evaluate all valid moves.
   - Simulate the opponent's best possible responses.
   - Pick the move that forces a win or avoids loss.
5. **Terminal Check:**
   - If no sticks remain → current player wins (+1), opponent loses (-1).

Sample Console Output

```
Piles: 1:[3], 2:[4], 3:[5]
Your move - Choose a pile (1-3): 2
How many sticks to remove from pile 2? 3

Piles after your move: 1:[3], 2:[1], 3:[5]
AI is thinking...

AI removes 2 sticks from pile 3
Piles after AI move: 1:[3], 2:[1], 3:[3]

Your move: _
```

Functions to Implement

- is_terminal(piles)
- get_valid_moves(piles)
- apply_move(piles, pile_index, num_to_remove)
- minimax(piles, is_maximizing)
- find_best_move(piles)

```python
def initialize_game():
    """Initialize the piles with 3, 4, and 5 sticks."""
    return [3, 4, 5]

def is_terminal(piles):
    """Check if the game has ended (no sticks left)."""
    # TODO: Return True if all piles are empty
    pass

def get_valid_moves(piles):
    """Return all valid moves as (pile_index, num_to_remove) pairs."""
    # TODO: Generate a list of valid moves
    pass

def apply_move(piles, pile_index, num_to_remove):
    """Return a new list of piles after applying a move."""
    # TODO: Copy piles and apply the move
    pass

def minimax(piles, is_maximizing):
    """Minimax recursive algorithm to determine best score."""
    # TODO: Base case for terminal state
    # TODO: Recursive call for maximizing and minimizing player
    pass
```

```python
def find_best_move(piles):
    """Return the best move for the AI using Minimax."""
    # TODO: Evaluate all valid moves using Minimax
    pass

def get_human_move(piles):
    """Get a valid move from the human player."""
    # TODO: Prompt user, validate input, and return move
    pass

def game_loop():
    """Main game loop where human and AI take turns."""
    piles = initialize_game()
    current_player = "HUMAN"

    while not is_terminal(piles):
        print(f"Piles: {piles}")
        if current_player == "HUMAN":
            pile, amount = get_human_move(piles)
        else:
            print("AI is thinking...")
            pile, amount = find_best_move(piles)
            print(f"AI removes {amount} from pile {pile+1}")

        piles = apply_move(piles, pile, amount)
        current_player = "AI" if current_player == "HUMAN" else "HUMAN"

    print(f"Game over! {current_player} loses.")
```