Game Theory

Project: Custom Game

Team Members:

- **Shimaa Said Hosny - 2205187**
- **Habiba Tarek Nassar - 2205188**
- **Nour Sherif Mohamed - 2205061**
- **A’laa Omar Ali - 2205013**
- **Mohamed Gamal Dowek - 2205131**

GameTheoryAnalyzer Code Report

1 Overview

The provided code is a Python implementation of a GameTheoryAnalyzer class that allows users to define a two-player strategic game, input strategies and payoffs, and perform various gametheoretic analyses. The code uses libraries such as numpy, matplotlib.pyplot, networkx, and itertools to handle game data, visualize game trees, and compute solutions. It is designed to run in a Jupyter Notebook environment, as indicated by the .ipynb file format.

The program supports:

- Defining a game with user-specified players, strategies, and payoffs.
- Displaying the game in normal form (payoff matrix).
- Visualizing the extensive form as a game tree (for two-player simultaneous games).
- Identifying dominated strategies (with optional iterative elimination).
- Finding best responses for each player.
- Identifying pure strategy Nash equilibria.
- Calculating mixed strategy Nash equilibria for 2x2 games.
- An interactive menu to navigate analysis options.

2 Code Structure

The code is organized as a single class, GameTheoryAnalyzer, with the following key components:

1. **Initialization (__init__)**:
   - Initializes an empty game dictionary and prompts the user to define the game via _get_user_game_input().
2. **Game Definition (_get_user_game_input)**:
   - Collects the number of players (with a focus on two players), player names, strategies, and payoffs.
   - Stores game data in a dictionary containing:
     - players: List of player names.
     - strategies: List of strategy lists for each player.
     - payoffs: Nested list representing payoffs for each strategy profile.
     - description: A string describing the game.

3. **Normal Form Display (display_normal_form)**:
    - Prints the payoff matrix for two-player games, with strategies as rows (Player 1) and columns (Player 2).
    - Includes a warning for games with more than two players, as the display is simplified.
4. **Extensive Form Display (display_extensive_form)**:
    - Uses networkx and matplotlib to generate a game tree for two-player simultaneous games.
    - Saves the visualization as extensive_form_game.png.
    - Represents simultaneous moves with an information set for Player 2.
5. **Dominated Strategies (find_dominated_strategies)**:
    - Identifies strictly dominated strategies, with an option for iterative elimination to find rationalizable strategies.
    - Returns the reduced strategy set after elimination.
6. **Best Responses (find_best_responses)**:
    - Computes the best response strategies for each player given the opponent's strategy.
    - Focused on two-player games for simplicity.
7. **Pure Strategy Nash Equilibria (find_nash_equilibriam)**:
    - Identifies strategy profiles where no player can improve their payoff by unilateral deviation.
8. **Mixed Strategy Nash Equilibrium (calculate_mixed_strategy_equilibrium)**:
    - Calculates mixed strategy probabilities for 2x2 games using the indifference principle.
    - Computes expected payoffs at the equilibrium.
9. **Main Loop (run_analysis)**:
    - Provides an interactive menu for users to select analysis options or redefine the game.
10. **Helper Function (_get_payoff_for_profile)**:
    - Retrieves the payoff for a specific player given a strategy profile.

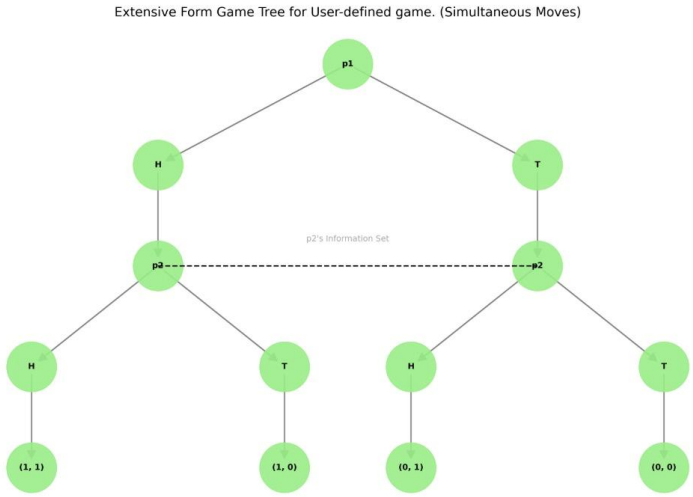
3 Strengths

- **User-Friendly Interface**: The interactive menu and clear prompts make it easy for users to define games and select analyses.
- **Comprehensive Analysis**: Covers key game theory concepts, including normal and extensive forms, dominated strategies, best responses, and Nash equilibria (pure and mixed).
- **Visualization**: The extensive form visualization using networkx and matplotlib is a strong feature for understanding simultaneous games.

- **Error Handling**: Includes input validation for payoffs and warnings for unsupported cases (e.g., non-2x2 games for mixed strategies).
- **Modularity**: The class-based structure with separate methods for each analysis enhances maintainability.

4 Limitations

1. **Limited Support for N-Player Games**:
    - While the code accepts more than two players, many features (e.g., normal form display, extensive form visualization, mixed strategy calculations) are tailored for two-player games. For N>2 players, the normal form display simplifies by fixing other players' strategies, which is not ideal for general N-player games.
    - Payoff handling for N-player games is incomplete, as the nested list structure assumes two-dimensional matrices, which doesn't scale well for higher dimensions.
2. **Extensive Form Limitation**:
    - The extensive form is designed only for simultaneous two-player games, limiting its applicability to sequential games or games with more players.



3. **Mixed Strategy Restriction**:
    - Mixed strategy Nash equilibrium calculations are restricted to 2x2 games, using an analytical approach. Larger games would require numerical methods (e.g., linear programming), which are not implemented.
4. **Payoff Matrix Indexing**:
    - The _get_payoff_for_profile function and related methods have indexing issues for N-player games, as they assume a fixed structure. This can lead to errors or incorrect results for games with more than two players.

5. **Visualization Dependency**:
    - The extensive form visualization requires matplotlib and networkx, and the output is saved as a file rather than displayed inline in the notebook (though this may be intentional for Colab compatibility).
6. **No Support for Sequential Games**:
    - The code assumes simultaneous moves, lacking support for sequential games, which are common in extensive form representations.

5 Suggestions for Improvement

1. **Enhance N-Player Support**:
    - Modify the payoff structure to use a more flexible data structure (e.g., a dictionary keyed by strategy profile tuples) to handle N-player games accurately.
    - Extend the normal form display to support user-specified slices of the payoff matrix for N>2 players, allowing users to fix strategies for all but two players.
2. **Support Sequential Games**:
    - Add functionality to define sequential games, allowing users to specify the order of moves and information sets for extensive form visualization.
3. **Expand Mixed Strategy Calculations**:
    - Implement numerical methods (e.g., using scipy.optimize for linear programming) to compute mixed strategy Nash equilibria for games larger than 2x2.
4. **Improve Visualization**:
    - Enable inline display of the extensive form game tree in Jupyter Notebooks using %matplotlib inline.
    - Add customization options for visualization (e.g., node colors, edge labels).
5. **Robust Payoff Handling**:
    - Refactor _get_payoff_for_profile to handle N-dimensional payoff matrices more robustly, possibly using recursive indexing or a flatter data structure.
6. **Add Input Validation**:
    - Strengthen input validation for strategies and payoffs to prevent edge cases (e.g., empty strategy lists, non-numeric payoffs).
7. **Documentation and Comments**:
    - Add docstrings to all methods and include more detailed comments explaining the game theory concepts and mathematical formulas used (e.g., for mixed strategy calculations).
8. **Unit Tests**:

- Include unit tests to verify the correctness of game-theoretic calculations, especially for edge cases like degenerate games or zero denominators in mixed strategy calculations.

6 Example Output Analysis

The provided output corresponds to a 2x2 game with players p1 and p2, each with strategies H and T, and the following payoff matrix:

p1 \ p2	H	T
H	(1, 1)	(1, 0)
T	(0, 1)	(0, 0)

- **Normal Form**: The matrix is displayed clearly, with proper alignment and formatting.
- **Extensive Form**: A game tree is generated, showing simultaneous moves with an information set for p2, saved as extensive_form_game.png.
- **User Interaction**: The menu-driven interface allows users to explore different analyses, with the sample run showing normal form display, extensive form visualization, and program exit.

7 Conclusion

The **GameTheoryAnalyzer** is a well-structured tool for analyzing two-player strategic games, offering a range of game-theoretic analyses with a focus on 2x2 simultaneous games. While it excels in user interaction and visualization for two-player games, it has limitations in handling N-player games and sequential games. With enhancements to support more general game structures, robust payoff handling, and numerical methods for mixed strategies, it could become a more versatile tool for game theory education and analysi