Developing strategies for the bidding card game "Diamonds"with GenAI

JAGROTU SHREYA, WE COHORT-5

1 Introduction

This assignment aims to explore the potential of GenAI in developing and analyzing bidding strategies for the card game Diamonds. Through a structured approach, the focus is on utilizing Gemini, the GenAI tool, to:
Comprehend the Dynamics of the Diamonds Card Game Formulate Effective Bidding Strategies Refine and Improve Strategy Outputs The report outlines the following structured process for developing and analyzing bidding strategies for Diamonds:

1)Problem Statement
2)Teaching Gemini the Game Dynamics
3)Iterative Strategy Development
4)Analysis of Strategy Outputs
5)Conclusion

2 Problem Statement

Game of diamonds Each player gets a suit of cards other than the diamond suit. The diamond cards are then shuffled and put on auction one by one. All the players must bid with one of their own cards face down. The banker gives the diamond card to the highest bid, i.e. the bid with the most points.
Card Ranking 2<3<4<5<6<7<8<9<T<J<Q<K<A
The winning player gets the points of the diamond card to their column in the table. If there are multiple players that have the highest bid with the same card, the points from the diamond card are divided equally among them. The player with the most points wins at the end of the game.

3 Teaching GPT-3 the Game

Guided by the game's rules and mechanics, we imparted the nuances of the Game of Diamonds to our AI counterpart. Through a step-by-step breakdown, we elucidated the intricacies of card manipulation, bid assessment, and score allocation. By translating these concepts into executable Python code, we equipped GPT-3 with the foundational knowledge necessary to understand and engage in gameplay simulations

4 Strategy Integration

Strategies are woven into the fabric of the game, influencing both player deci sions and algorithmic logic. Key strategies such as risk management, observa tion, and adaptability are pivotal in maximizing one's chances of victory. In 1 our Python implementation, these strategies are reflected in the bid assessment, with players dynamically adjusting their bids based on card values and opponent behavior.

article enumitem

4.1 Player Class

[style=nextline]Initializes a player with a name, a list of cards, and points.

4.2 Card Class

[style=nextline]Initializes a card with a rank, a suit, and calculates its points based on the rank.

4.3 DiamondsGame Class

[style=nextline]Initializes the game with a list of players and creates a shuffled deck of diamond cards. Executes the auction phase of the game. It iterates through each diamond card, allows players to bid with one of their cards face down, determines the highest bid(s), and allocates points to the winning player(s). Displays the final results of the game by printing each player's name and points, as well as announcing the winner.

These functions are crucial for running the game and managing its flow. They encapsulate key functionalities such as player initialization, card handling, auction phase, and result display.

Listing 1: Example Python code

```
import random

class Player:
    def __init__(self, name, cards):
        self.name = name
        self.cards = cards
        self.points = 0

class Card:
    def __init__(self, rank, suit):
        self.rank = rank
        self.suit = suit
        self.points = self.calculate_points()

    def calculate_points(self):
        # Define the points for each rank
        points_dict = {'2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '
        return points_dict[self.rank]

class DiamondsGame:
    def __init__(self, players):
        self.players = players
        self.diamonds = [Card(rank, 'diamonds') for rank in ['2', '3', '4', '5',
        random.shuffle(self.diamonds)

    def auction_phase(self):
        for diamond in self.diamonds:
            bids = {}
            for player in self.players:
                # Player makes a bid with one of their cards face down
                bid_card = random.choice(player.cards)
                bids[player] = bid_card

            # Determine the highest bid(s)
            highest_bid = max(bids.values(), key=lambda card: card.points)
            winners = [player for player, bid in bids.items() if bid == highest_

            # Allocate points to the winners
            points_to_allocate = diamond.points / len(winners)
            for winner in winners:
                winner.points += points_to_allocate

    def display_results(self):
        for player in self.players:
            print(f"{player.name}: {player.points} points")
        winner = max(self.players, key=lambda player: player.points)
        print(f"The winner is {winner.name} with {winner.points} points!")
```

```
# Example usage:
player1 = Player("Alice", [Card(rank, 'hearts') for rank in ['2', '3', '4', '5',
player2 = Player("Bob", [Card(rank, 'clubs') for rank in ['2', '3', '4', '5', '6
players = [player1, player2]

game = DiamondsGame(players)
game.auction_phase()
game.display_results()
```

5 Iteration and Analysis

Throughout the development process, the implementation of the Game of Diamonds undergoes iterative refinement and rigorous analysis to ensure its robustness, fairness, and alignment with the intended gameplay experience. Code Refinement involves multiple iterations to enhance clarity, efficiency, and functionality, addressing bugs and logical errors systematically through debugging and testing. Strategic Integration incorporates risk management, observation, and adaptability into gameplay logic, fine-tuning strategies for effectiveness. Simulation and Testing involve extensive testing under various scenarios to evaluate performance, identify vulnerabilities, and ensure fairness and stability. Player Feedback is solicited and integrated, with iterative playtesting sessions providing insights into mechanics and balance. Statistical Analysis is conducted to identify patterns and inform decision-making, ensuring competitiveness. Fairness and Balance mechanisms are implemented, such as tiebreakers, to foster balanced gameplay. User Experience is prioritized, with intuitive interfaces and feedback mechanisms to enhance player engagement and clarity on rules and actions.

6 Conclusion

The Python implementation of the Game of Diamonds emerges as a harmonious blend of strategic gameplay and algorithmic logic. Through this journey, we unravel the strategic interplay between players and algorithms, exploring the dynamic landscape of competitive gaming. By integrating strategies into the codebase and iteratively refining our approach, we pave the way for an immersive and intellectually stimulating gaming experience

7 Reference

Converstion with chatGpt