Assignment-2

Submitted by:
Erum Meraj

26.01.2024

1 Introduction

The game *Diamonds* offers an engaging test of skill for three players. Initially introduced during a bootcamp led by our program dean, Asokan Pichai, this report outlines the journey of leveraging Gen AI to craft effective strategies, incorporating it as a third participant, and devising a systematic approach to test the implemented code.

2 Rules of the Game: Diamonds

Setup:

- 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.
- Point hierarchy: $2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < J < Q < K < A$
- 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 Training

Initiating dialogue with Gemini, I proposed the idea of engaging in a game based on provided rules. Gemini responded by seeking clarification on the total number of players involved. Upon confirming three players, I received the following response: “Great! Let’s play a round of this diamond bidding game with 3 players. I’ll be one of the players. We’ll need a way to simulate dealing the cards and bidding anonymously. Here’s an option:

- Dealing the cards: You can tell me which suit (Spades, Hearts, or Clubs) each player gets. I can’t see your cards, so you’ll need to keep track of them yourself.
- Bidding: For each diamond card being auctioned, tell me the value of the card you want to bid (e.g., ”7 of Clubs”). I will do the same, keeping my bid hidden from you.
- Resolving the bid: You can then tell me the highest bid value and how many players tied for the highest bid. Based on that, I can tell you who wins the diamond card and any points awarded.

Does this sound like a workable way to play?” However since the task was primarily to get GenAI to build winning strategies for the game I proceeded to do that. When I sought guidance from GenAI on strategies for the diamond bidding game, it presented a comprehensive understanding of the gameplay. Here’s a summary of its insightful suggestions:
GenAI outlined several strategic approaches:

- **Balancing Point Preservation and Diamond Acquisition:** At the outset, prioritize retaining high-value cards within your suit. These can prove pivotal in securing valuable diamonds later. However, be willing to sacrifice lower-value cards to secure desirable diamonds, especially if others may do the same.
- **Understanding Opponents:**
  - **Observing Bidding Patterns:** Pay close attention to how opponents bid on different diamond values. Recognize whether they consistently pursue high-value diamonds aggressively or adopt a more reserved approach. This observation can inform your predictions and adjustments.
  - **Identifying Bluffs:** If a player consistently bids high on low-value diamonds, it could indicate a bluff intended to deter bidding on more valuable ones later.
- **Bidding Strategies:**
  - **Incremental Bidding:** Avoid revealing your hand prematurely. Begin with lower value bids and incrementally increase them if necessary. This tactic maintains ambiguity and keeps opponents guessing.

  - **Strategic Sacrifices:** Consider sacrificing a high-value card from your suit to secure a particularly valuable diamond, especially if significant competition is expected.
  - **Adaptation:** Recognize that there's no one-size-fits-all strategy. Stay flexible and adapt your approach based on the game's dynamics, revealed diamonds, and your remaining cards.
- **Considering Overall Point Spread:** Instead of obsessing over winning every diamond, monitor opponents' point totals. Prioritize bids that can substantially enhance your lead or hinder opponents from catching up.

These strategies serve as a foundation for your gameplay, offering a framework that can be refined through experience and personal play style.
In attempting to translate these strategies into code, I prompted GenAI to detail the implementation process. It responded with a well-structured strategy encompassing various components and tactics. I saved this comprehensive strategy in a text file, intending to consult Gemini by breaking down its elements to obtain the corresponding code for each strategy.

4 Code

```
from collections import Counter

class Hand:
    """
    This class represents a player’s hand in the diamond bidding game.
    """
    def __init__(self, suit):
        """
        Initializes the hand with a specific suit using a while loop.
        """
        self.suit = suit
        self.cards = Counter()
        rank = 2
        while rank <= 14:
            self.cards[str(rank)] = 1  # Add each card value (2-A)
            rank += 1
        self.revealed_diamonds = []  # Track revealed diamond values
        self.opponent_history = {}  # Optional: Track opponent bids (dictionary)

    def remove_card(self, card_value):
        """
        Removes a card from the hand if present.
        """
        if str(card_value) in self.cards:
            self.cards[str(card_value)] -= 1
            if self.cards[str(card_value)] == 0:
                del self.cards[str(card_value)]  # Remove card if count reaches 0

    def get_remaining_cards(self):
        """
        Returns a list of remaining card values in the hand.
        """
        return list(self.cards.keys())

    def has_card(self, card_value):
        """
        Checks if the player has a specific card value in their hand.
        """
        return str(card_value) in self.cards and self.cards[str(card_value)] > 0

    def update_revealed_diamond(self, diamond_value):
        """
        Updates the list of revealed diamond values.
        """
        self.revealed_diamonds.append(diamond_value)

    def update_opponent_bid(self, opponent_name, diamond_value, bid_value):
        """
        Updates the opponent’s bidding history (optional).
        """
        if opponent_name not in self.opponent_history:
            self.opponent_history[opponent_name] = []
        self.opponent_history[opponent_name].append((diamond_value, bid_value))

    def place_bid(self, diamond_value):
        """
        Places a bid for a diamond based on a bidding strategy.
        """
        min_bid = diamond_value // 2

        if len(self.get_remaining_cards()) > 5 and diamond_value < 8:
            return min_bid

        if diamond_value >= 10:
            min_bid += 1

        high_cards_remaining = sum(self.cards[card_value] for card_value in [’J’, ’Q’, 
        if high_cards_remaining > 2 and diamond_value >= 8:
            min_bid += 1

        if self.opponent_history:
            for opponent, bids in self.opponent_history.items():
                recent_bids = bids[-2:]
                if any(value >= diamond_value // 2 for value, _ in recent_bids):
                    min_bid += 1

        return min_bid

# Example usage
hand = Hand("Spades")
print(hand.get_remaining_cards())
hand.remove_card(7)
print(hand.get_remaining_cards())
print(hand.has_card(7))
hand = Hand("Spades")
hand.update_revealed_diamond(8)
hand.update_revealed_diamond(10)
print(hand.revealed_diamonds)
hand = Hand("Spades")
hand.update_revealed_diamond(8)
print(hand.place_bid(5))
print(hand.place_bid(10))
hand = Hand("Spades")
hand.update_revealed_diamond(8)
print(hand.place_bid(5))
print(hand.place_bid(10))
print(hand.place_bid(8))
hand.remove_card('K')
print(hand.place_bid(8))
hand = Hand("Spades")
hand.update_revealed_diamond(8)
print(hand.place_bid(5))
hand.update_opponent_bid("Player1", 8, 4)
hand.update_opponent_bid("Player2", 5, 1)
print(hand.place_bid(8))
```

5 Conclusion

In a recent session, I discussed the challenge of explaining a card game, noting its complexity. I was particularly impressed by Gemini's response, which provided a comprehensive solution. However, upon review, certain portions of the generated code required adjustments to meet expectations. Despite this, the collaboration was a positive and successful experience overall.