Diamonds Game

Akshada Kulkarni

1 Introduction

The game of Diamonds is a two-player strategic card game where players bid over diamond cards to score points. The bidding process involves players selecting a card from their hand to represent their bid, and the player with the highest bid wins the diamond card.

2 Problem Statement

Create a report on the teaching and generation of a solution by genAI of the Diamonds card game.

3 Initial Prompt

Here is the diamonds game: 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. 2-3-4-5-6-7-8-9-J-Q-K-A, A being the highest and 2 the lowest. 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. if all the players come up with the same card value, divide equally between them The player with the most points wins at the end of the game. Did you understand how to play?

4 Teaching the AI

To help teach the AI the game, I asked it to play with me. I played a few rounds with the AI, going through all possible situations, then I asked it for strategies to write code to play this game.

5 Iterating upon strategy

After a few iterations and clarifications, such as ensuring the diamonds deck was shuffled randomly, error handling etc., the strategy we ended on was as such:

5.1 Setup

Define the deck of cards, including suits (hearts, spades, clubs), card values (2 through Ace), and their corresponding points. Initialize players' decks, assigning one suit to each player (excluding diamonds). Shuffle the deck of diamond cards.

5.2 Main Game Loop

Iterate through each diamond card in the shuffled deck. For each diamond card: Prompt each player to bid with one of their cards face down, reveal bids simultaneously, and determine the highest bidder: If only one player bids the highest value card, they win the diamond card. If multiple players bid the same highest value card, the diamond card is divided equally among them. Update the points for the winning bidder(s).

5.3 End Game

After all diamond cards have been auctioned, calculate the total points for each player. Determine the winner based on the player with the highest total points.

5.4 Strategies to play the game

The AI also offered strategies for playing the game in real life, which included bluffing, risk management(bid conservatively or aggressively depending on their current points and the remaining diamond cards), and observation(players can keep track of which cards have been played and adjust their bidding strategies accordingly).

5.5 User Interface (Optional)

Implement a user interface for players to interact with the game, make bids, and view the current state of the game. Display the diamond cards being auctioned, the bids made by each player, and the current points for each player.

5.6 Error Handling and Validation

Implement error handling to ensure valid inputs from players. Validate bids to ensure they are within the allowed range and meet any other game rules.

5.7 Testing

Test the game thoroughly to ensure that it functions correctly and handles various scenarios, including edge cases. By breaking down the game into these steps, you can systematically implement each part of the game logic and create a functional Diamonds game.

6 Code

I then asked the AI for the code. After a few adjustments, such as taking care of decimals, revealing the diamond card of each round and displaying each player's updated points after each round, the code we finished with is below.

```
import random

# Define card values and points
card_values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']
points = {'2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9, '10': 10, 'J': 11, 'Q': 12, 'K': 13, 'A': 14}

# Initialize players' decks
player_decks = {'Player 1': card_values.copy(), 'Player 2': card_values.copy()}

# Assign suits to players
player_suits = {'Player 1': 'Hearts', 'Player 2': random.choice(['Spades', 'Clubs'])}

# Initialize points for each player
player_points = {'Player 1': 0, 'Player 2': 0}

# Shuffle diamond cards
diamond_cards = card_values[0:13]  # Selecting cards 2 through Ace
random.shuffle(diamond_cards)

# Main game loop
for diamond_card in diamond_cards:
    print(f"\nAuctioning {diamond_card} of Diamonds")
    bids = {}

    # Prompt each player to bid
    for player, suit in player_suits.items():
        print(f"{player}, your suit is {suit}.")
        print(f"Your current points: {player_points[player]}")

        # Ask player to choose a card to bid with
        while True:
            bid_card = input("Choose a card to bid with (2-A): ").upper()

            # Check if the chosen card is valid and not already played
            if bid_card not in card_values:
```

```
                print("Invalid input. Choose a card to bid with (2-A): ")
                continue
            elif bid_card not in player_decks[player]:
                print("You've already played this card. Choose a different card.")
                continue
            else:
                bids[player] = bid_card
                player_decks[player].remove(bid_card)
                break

    # Reveal bids
    print("\nRevealing bids:")
    for player, bid_card in bids.items():
        print(f"{player} bids {bid_card}")

    # Determine the highest bidder
    highest_bid = max(bids.values(), key=lambda x: points[x])
    winners = [player for player, bid_card in bids.items() if bid_card == highest_bid]

    # Divide points equally among winners
    points_to_distribute = points[diamond_card]
    points_per_winner = points_to_distribute / len(winners)

    # Update points for winners
    for winner in winners:
        player_points[winner] += points_per_winner

    # Reveal the diamond card and updated points after each round
    print(f"\nThe diamond card was {diamond_card}")
    print("Updated points:")
    for player, p in player_points.items():
        print(f"{player}: {p}")

# Game over, display final points
max_points = max(player_points.values())
winning_players = [player for player, points in player_points.items() if points == max_points]
print("\nGame Over!")
for player, points in player_points.items():
    print(f"{player} earned {points} points.")
if len(winning_players) == 1:
    print(f"\n{winning_players[0]} wins with {max_points} points!")
else:
    print("\nIt's a draw! The following players tied with the highest score:")
    for player in winning_players:
        print(f"- {player}")
```

7 Analysis and Conclusion

The AI tool was quick to grasp the rules of the language, however it was susceptible to many small misunderstandings and errors. It easily wrote mostly-

functional, well-written code, however again with a few errors when it came to decimal handling, etc.

However, it quickly improved its understanding, displaying a great learning capacity. We could achieve significant improvements in performance, proving the effectiveness of the iterative teaching process.

In conclusion, AI proved to be a valuable tool for developing strategies to play a game and code it, with its quick understanding showing promise for future enhancements of the game as well.