Simple Poker Game Design, Simulation, and Probability

Nanxiang Wang
Foothill High School
Pleasanton, CA 94588
nanxiang.wang309@gmail.com

Mason Chen
Stanford Online High School
Stanford, CA, 94301, USA
mason05@ohs.stanford.edu

Abstract

This project simulates poker probability. The game selects three cards from four shared cards with the two player cards to form the best hand to determine is the winner. In order to simplify the probability scenario, partial deck (9, 10, J, Q, K, and A) is used to increase the probability of higher ranked hands such as Four of a Kind and Full House. I used a JAVA program to run simulations so I could calculate each player's winning probability. The program is programmed to generate random cards for both players. I compared the simulated results with the expected probability derived based on calculations and confirmed that the JAVA simulated probability could match the expected probability. This project successfully integrated JAVA computer science and statistics/probability with the poker application.

Keywords
Java, Statistics, Probability, Randomized

1. Introduction

Most poker players lose money in poker since they play without applying the probability and assessing their risk on each play. The objective of this paper is to use JAVA to simulate poker probability and study Sample Size effect on Statistics and decision making. The project scope is for learning purpose, not for gambling purpose. Authors used partial deck (9, 10, J, Q, K, A) of 24 cards to simplify JAVA poker simulation. Figure 1 has listed the rankings of different matched patterns for the full deck (52 cards) scenario. The full deck poker for 6 to 7 random cards is very popular in most Poker tournament[]. There is also an US Patent [3] that studied the partial deck on Royal Flush probability. In this paper, the authors will study the Poker Probability on the 24-cards Partial Deck and use JAVA programming to verify the winning probability between two players. The ranking of Partial Deck may be different from the Full Deck.

2. Study Partial Deck Probability

I used partial deck (9, 10, J, Q, K, A) of 24 cards to simplify JAVA poker simulation. Partial Deck can increase the matching probability on higher ranked patterns such as Four of a Kind, and Full House. Partial Deck Poker may also simplify JAVA simulation process concentrated on higher ranked patterns which may be critical for real time decision making on each betting move.

Figure 1. Ranking of Matched Patterns for Full Deck.

2.1 Poker Partial Deck Case Study

In order to demonstrate Poker probability and simulation simply, a special case study has been created in Figure 2.

Figure 2. Case Study

Four cards are shown in the shared dealer field; two players have one card shown and one card hidden as in Figure 4. Each player will calculate the winning probability by guessing the other unknown card of the opponent's hand. To keep in simple, the authors will consider "tie" if the matching pattern is the same and the card numbers are the same even the card category is different (for example, Spade A will be treated the same as Heart A as tie).

3. JAVA Program

The JAVA program has seven different stages. First, it creates the deck, then it shuffles the deck. Next, it draws cards for the board and then for the players. It then evaluates both players' hands and compares the scores. Lastly, it prints out who wins.

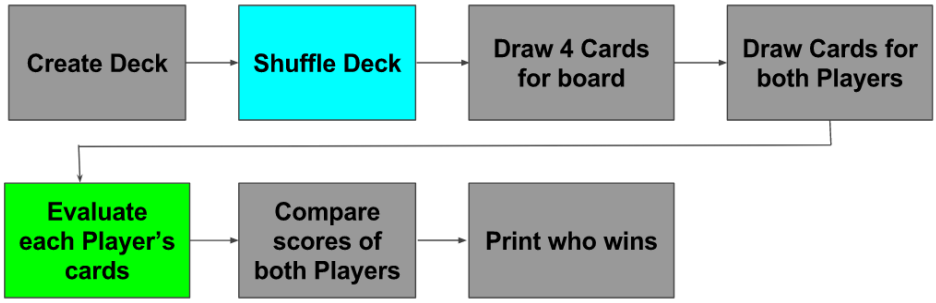

Figure 3. Java Flowchart

3.1 Shuffle

I used random sequence instead of random number generator to shuffle the deck. This makes sure that there will be no repeating cards. To shuffle the deck, the program generates a random index. It then swaps the first card with the card at that index. It then moves on to the second card, and repeats. This continues until all 24 cards have been swapped.

Shuffle Deck

```
int length = decksize;

Random random = new
Random();

for (int i=0;i<length;i++){

    int j = i +
random.nextInt(length-i);

    swapCards(i, j, cards);

}
```

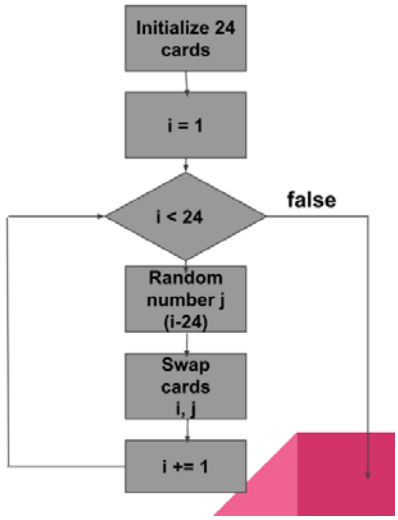

Figure 4. Shuffle Function

3.2 Scoring

To score each player's hand, I set specific values to the various hands. The program then goes through all possible combinations of the cards on the table and in the player's hand. As it runs, it stores the highest value, which it uses at the end to find out who wins.

1.	Royal Flush/Straight Flush/Flush	1.	900/800/500
2.	Four of a Kind	2.	700
3.	Full House/Three of a Kind	3.	600/300
4.	Two Pairs/One Pair	4.	200/100
5.	Straight	5.	400
6.	Single	6.	0

Figure 6. Scoring Hand Values

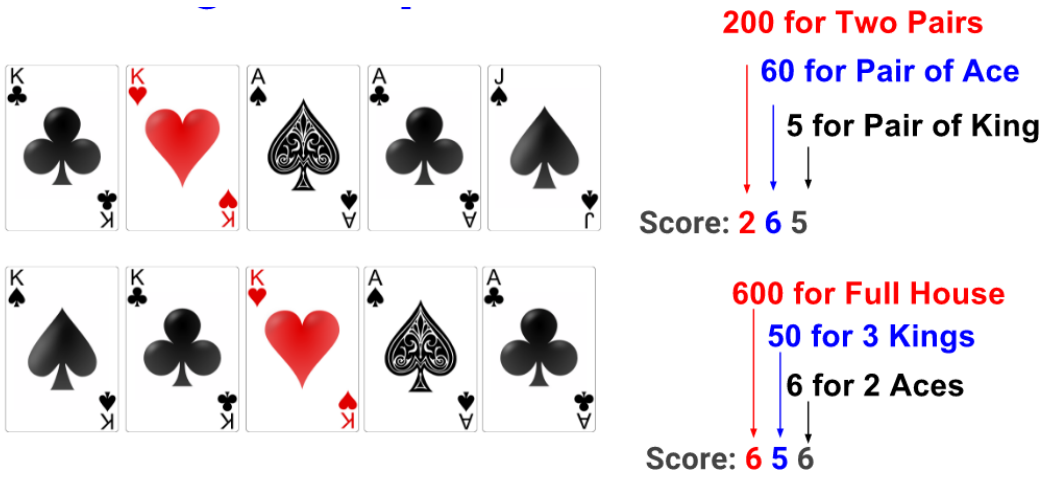

Figure 7. Scoring Example

3. 3 Evaluate the cards

To evaluate the cards, the Java program sets a variable i at 0. The program then swaps the card at i with the card at 5. It then scores the cards, stores the score, and increases the index by one. This continues for 5 more runs. If a score is higher than the score stored by the function, the function then replaces the stored value with the new highest score. At the end of the function, the program returns the stored value.

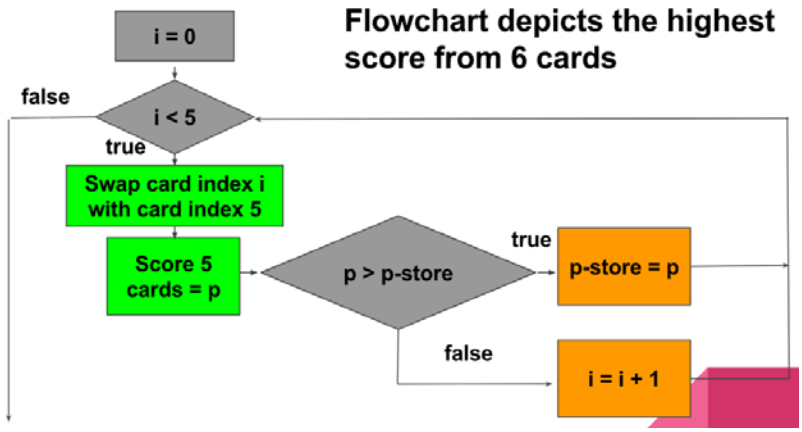

Figure 5. Evaluation for 6 Cards

4. Results

At the end of 20 tests, I found that the simulated percentage of Player One winning over Player Two was 75%. I also found that the simulated percentage of any player getting a full house was 1.563%, and that the simulated probability of both players getting a full house was 0.0244%. Compared with the expected probabilities, the probabilities I simulated are very close to the expected.

	Player One Full House	Player Two Full House	Both Player Full House
Expected	1.3%	1.3%	0.03%

Simulated	1.563%	1.563%	0.0244%

Figure 6. Table of Probabilities

5. Conclusion

Based on my calculations and experiments, I conclude that the player with a higher starting hand is more likely to win.

In the future, I will try to find the probability of winning with one player only fully randomized, find the probability of winning with both players fully randomized, and find the probability of winning with board only randomized

References

Billings, D. , et al., The challenge of poker, Artif. Intell., vol. 134, pp. 201-240, 2002.

Li, W., and Shang, L., Estimating Winning Probability for Texas Hold'em Poker, *International Journal of Machine Learning and Computing*, Vol. 3, No. 1, 2013

Ward, B., and Cabot A., Partial-deck poker game with guaranteed royal flush opportunity, *US Patent* 20040212147 A1, 2004

D. Billigs, "Algorithms and Assessment in Computer Poker," Doctor PhD, *Department of Computing Science, University of Alberta*, 2006.

D. Billings, et al., "The challenge of poker," *Artif. Intell.*, vol. 134, pp. 201-240, 2002.

D. Felix, et al., "An Experimental Approach to Online Opponent Modeling in Texas Hold'em Poker," presented at *the Proceedings of the 19th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, Savador, Brazil, 2008.