Solitario di Prina

Gallina Roberto

13/11/2022

Contents

1 Premesse

In tutto il documento ci si riferirà a il *mazzo (Deck)*, esso è da intendere come un mazzo di 40 carte, divise in 4 semi. Per convenzione durante il documento si userà il mazzo francese le cui carte sono:

- l'asso
- il due
- il tre
- il quattro
- il cinque
- il sei
- il sette
- il fante (J)
- la regina (Q)
- il re (K)

Mentre i semi sono:

- i cuori
- i quadri
- i fiori
- i picche

Il mazzo è quindi composta dalle carte sottostanti:

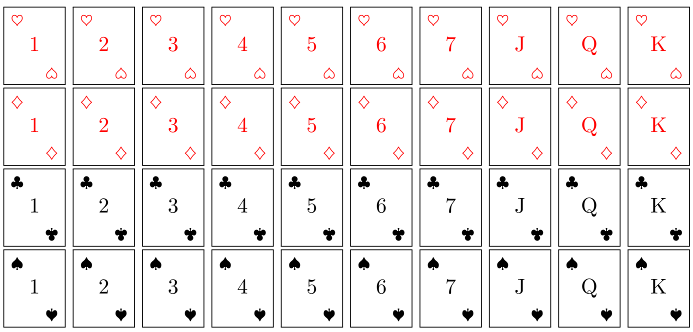


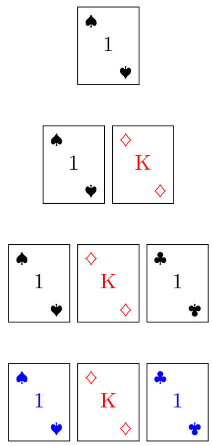
Figure 1: Mazzo

In tutto il documento ci si riferirà a la *sequenza*, essa rappresenta l'ordine delle carte nel mazzo mescolato.

2 Funzionamento del gioco

Il gioco è molto semplice, dato un mazzo da gioco mischiato, si tengono tutte le carte coperte in pila; si scopre le prime tre carte. Nel caso la prima e la terza carta hanno stesso valore o stesso seme, allora la seconda carta viene spostasta sopra la prima (avvicinando la terza). Successivamente si scopre un'altra carta di ricomincia a controllare dall'inizio.

Ecco un esempio del funzionamento



Avendo lo stesso valore (1), la carte centrale sale sopra la prima

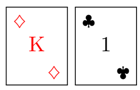


Figure 2: Esempio di funzionamento

2.1 Sequenza corretta

Viene definita *sequenza corretta*, una sequenza, in cui terminate le carte si hanno due esattamente due pile di carte.

Eccone un esempio

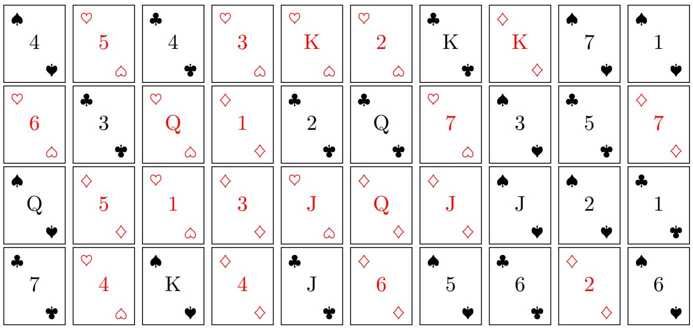


Figure 3: Sequenza corretta

2.2 Sequenza perfetta

Viene definita *sequenza perfetta*, una sequenza corretta, in cui è esattamente l'ultima carta a formare la seconda pila; per cui terminate le carte di hanno esattamente due pile di carte in cui la seconda pila ha esattamente una carta

Eccone un esempio

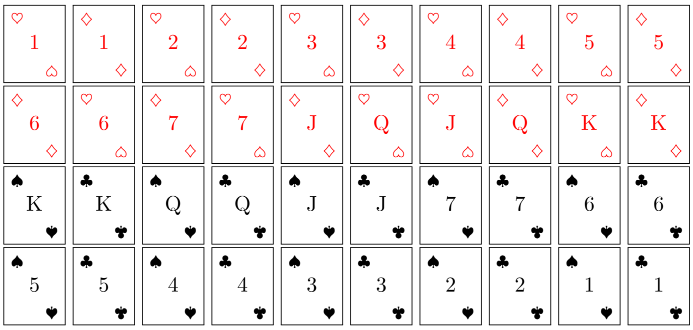


Figure 4: Sequenza perfetta

2.3 Sequenza n-perfect

Viene definita *sequenza n-perfect*, una sequenza, in cui terminate le carte non ci sono pile con più di un carta, ossia il numero delle pile è uguale al numero delle carte

Eccone un esempio

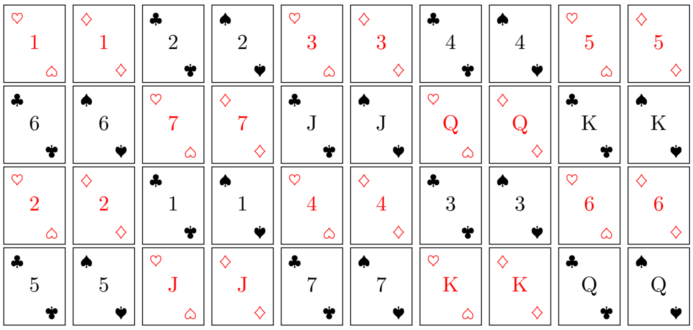


Figure 5: Esempio di sequeza n-perfect

3 Introduzione

Durante le innumerevoli partite giocate da me giocate, non è capito che la sequenza fosse corretta o ben che meno perfetta, e quindi nata in me l'idea si sapere quanto siano rare tali sequenze, ho quindi iniziato ad analizzare il gioco dal punto di vista statistico.

Poichè il mazzo è composto da 40 carte, possiamo calcolare il numero totale di sequenze, esso sarà pari a $40! \simeq 8.1591528 \cdot 10^{47}$; questo numero computazionalmente enorme, non è quindi possibile su un computer verificare tutte le sequenze possibili per avere dati esatti. Allo stesso modo anche calcolare quante siano le sequenze corrette, perfette o n-perfect non è semplice, avendo dati esempi di tutte e tre possiamo sicuramente affermare che ne esistano.

4 Analisi statistica

Definiamo P_C la probabilità che una sequenza sia corretta, P_P la probabilità che una sequenza sia perfetta e P_{N-P} la probabilità che una sequenza sia n-perfect; queste probabilità sono a noi sconusciute, ma sicuramente $P_C > 0$, $P_P > 0$ e $P_{N-P} > 0$ avendo dato un esempio per ognuna.

Ci aspettiamo che su 10 sequenze, le sequenze corrette S_C^{10} siano:

$$S_C^{10} \simeq 10 \cdot P_C$$

Possiamo quindi isolare P_C

$$P_C \simeq \frac{S_C^{10}}{10}$$

Ovviamente questa è un'approsimazione, ma aumentando il numero di sequenze otterremo un risultato sempre più vicino a P_C, possiamo quindi definire:

$$P_C = \lim_{n \to +\infty} \frac{S_C^n}{n}$$

Allo stesso modo definiamo:

$$P_P = \lim_{n \to +\infty} \frac{S_P^n}{n} \qquad P_{N-P} = \lim_{n \to +\infty} \frac{S_{N-P}^n}{n}$$

Possiamo quindi scrivere un programma per generare quante più sequenze possibili in modo approssimare queste probabilità.

5 Implementazione

Iniziamo col definire gli enumeratori per i semi e per i valori, in modo da poterli usare con semplicità dopo.

```
class Seme {
    public static HEART = "1";
    public static DIAMOND = "2";
    public static FLOWER = "3";
    public static CLUB = "4";
}
```

```
class Value {
    public static ONE = "1";
    public static TWO = "2";
    public static THREE = "3";
    public static FOUR = "4";
    public static FIVE = "5";
    public static SIX = "6";
    public static SEVEN = "7";
    public static JACK = "J";
    public static QUEEN = "Q";
    public static KING = "K";
}
```

Definiamo ora una carta come una classe che memorizza un seme e un valore.

```
class Card{
    private Seme seme;
    private Value value;

    constructor(seme, value) {
        this.seme = seme;
        this.value = value;
    }

    public Seme getSeme() {
        return this.seme
    }

    public Value getValue() {
        return this.value
    }
}
```

Definiamo poi il mazzo come un contenitore di 40 carte; definiamogli poi:

- un metodo $shuffle$ che randomicamente mischia le carte nel mazzo.
- un metodo $sameValueOrSeme$ che confronta due carte, ritornando $true$ se hanno lo stesso valore o lo stesso seme
- un metodo $check$ che data una sequenza di carte le confronta facendo salire quelle con lo stesso valore o seme, ritornando la sequenza ridotta
- un metodo $isCorrect$ che analizza la sequenza è dice se la sequenza ridotta è di 2 carte

- un metodo $isPerfect$ che analizza la sequenza è dice se la sequenza ridotta è di 2 carte (in cui l'ultima è esattamente l'ultima scesa)
- un metodo $isNPerfect$ che analizza la sequenza è dice se la sequenza ridotta è di 40 carte

La classe risultante sarà simile a questa

```
class Deck{
    private Card cards[40];

    constructor() {
        this.cards.add(new Card(Seme.HEART, Value.ONE));
        this.cards.add(new Card(Seme.DIAMOND, Value.ONE));
        .
        .
    }

    public void shaffle() {
        cards.randomSort();
    }

    public bool sameValueOrSeme(a, b) {
        return a.value == b.value || a.seme == b.seme;
    }

    public Card[] check(cards) {
        i = 0;
        j = 2;

        while (j < cards.length) {
            if (sameValueOrSeme(cards[i], cards[j])) {
                cards.removeByIndex(i);
                i = 0;
                j = 2;
            } else {
                i++;
                j++;
            }
        }
        return cards;
    }

    public bool isCorrect() {
        return check(cards).length == 2;
    }

    public bool isPerfect() {
        last = cards.pop();
        cards = check(cards);

        if (cards.length == 2) {
            return sameValueOrSeme(cards[0], last);
        }
        return false;
    }

    public bool isNPerfect() {
        return check(cards).length == 40;
    }
}
```

Possiamo quindi scrivere un programma per calcolare sequenze corrette, perfette o n-perfect.

```
do {
    d = new Deck();
    d.shaffle();
} while (d.isCorrect());
// or d.isPerfect() or d.isNPerfect()

d.print();  // is a method for print the sequence
```

Oppure, ben più interessante, possiamo generare diverse sequenze contare quante di esse siano corrette, prefette o n-perfect

```
const MAX_TENT = 1 * 1000 * 1000;
correct = 0;
perfect = 0;
n_perfect = 0;

for (i = 0; i < MAX_TENT; i++) {
    d = new Deck();
    d.shaffle();

    if (d.isCorrect()) {
        correct++;
        if (d.isPerfect())
            perfect++
    }
    if (d.isNPerfect())
        n_perfect++;
}
```

Possiamo ora lanciare il programma generando 10, 100, 1 000, 10 000, 1 000 000, 10 000 000, 100 000 000 e 1 000 000 000 sequenze; vediamo il risultato

Sequenze generate	Sequenze corrette	P_C	Sequenze perfette	P_P	Sequenze n-perfect	P_{NP}
10	1	0.1	0	0	0	0
10^2	2	2	0	0	0	0
10^3	5	0.5	1	0.1	0	0
10^4	42	0.42	6	0.6	0	0
10^5	456	0.4559	141	0.141	0	0
10^6	4607	0.4607	1431	0.1431	1	0.0001
10^7	46245	0.46245	13995	0.13995	9	0.00009
10^8	462679	0.462679	140521	0.140521	86	0.000086
10^9	4640873	0.464087	1406504	0.140650	929	0.000092

Table 1: Risultati delle sequenze, le probabilità sono approssimate a 6 cifre decimali

6 Conclusioni

Come era prevedibile la generazione di più sequenze ha portato a un'approsimazione sempre migliore, possiamo quindi presumere che:

$$P_C \simeq 0.46 \qquad P_P \simeq 0.14 \qquad P_{N-P} \simeq 0.000092$$

Inoltre possiamo anche stimare il numero totale di sequenze corrette S_C

$$S_C = 40! \cdot P_C \simeq 4.75 \cdot 10^{47}$$

il numero totale di sequenze perfette S_P

$$S_P = 40! \cdot P_P \simeq 1.14 \cdot 10^{47}$$

e il numero totale di sequenze n-perfect S_{N-P}

$$S_{N-P} = 40! \cdot P_{N-P} \simeq 7.5 \cdot 10^{43}$$

Inoltre come da definizione, ogni sequenza perfetta è una sequenza corretta, ma non vale il contrario, esistono quindi delle sequenze corrette che non sono perfette, definiamo quindi

7 Dati statistici

Possiamo poi modificare leggermente lo script, infatti usando il metodo check, possiamo valutare la lunghezza della sequenza ridotta e ottenere più dati riguardo la distribuzione delle sequenze.

Lungezza sequenze	Sequenze trovate	Percentuale
2	4 640 873	0,464 087
3	17 902 642	1,790 264
4	28 309 368	2,830 936
5	34 732 813	3,473 281
6	38 441 358	3,844 135
7	40 810 279	4,081 027
8	42 713 517	4,271 351
9	44 331 576	4,433 157
10	45 707 571	4,570 757
11	46 840 134	4,684 013
12	47 716 945	4,771 694
13	48 287 643	4,828 764
14	48 525 463	4,852 546
15	48 372 274	4,837 227
16	47 807 739	4,780 773
17	46 819 503	4,681 950
18	45 354 697	4,535 469
19	43 463 802	4,346 380
20	41 088 830	4,108 883
21	38 323 020	3,832 302
22	35 154 387	3,515 438
23	31 703 653	3,170 365
24	28 002 162	2,800 216
25	24 206 455	2,420 645
26	20 393 795	2,039 379
27	16 702 882	1,670 288
28	13 243 948	1,324 394
29	10 118 663	1,011 866
30	7 407 866	0,740 786
31	5 166 561	0,516 656
32	3 398 381	0,339 838
33	2 090 579	0,209 057
34	1 185 536	0,118 553
35	609 272	0,060 927
36	277 209	0,027 720
37	107 201	0,010 720
38	33 189	0,003 318
39	7 285	0,000 728
40	929	0,000 092

Table 2: Risultati delle sequenze, le probabilità sono approssimate a 6 cifre decimali

Possiamo poi graficare i dati ottenuti in modo da poterli interpretare meglio:

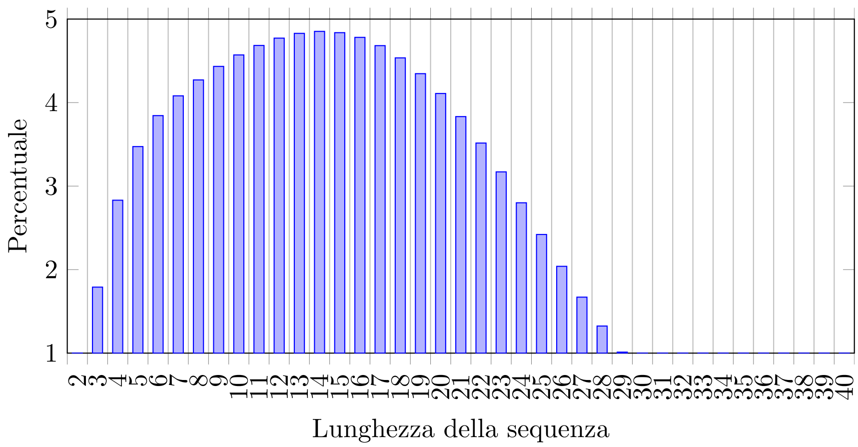


Figure 6: Distribuzione statistica delle sequenze

Possiamo anche analizzare il numero di sequenze corrette e perfette per capire come siano relazionate:

Sequenze	Lungezza corrette	Sequenze perfette	Percentuale
1 000 000 000	4 640 873	1 406 504	30,306 884

Table 3: Sequenze perfette rispetto alle sequenze corrette, le probabilità sono approssimate a 6 cifre decimali

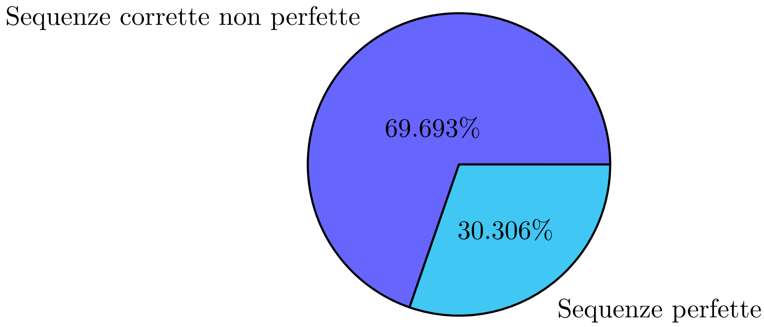


Figure 7: Sequenze perfette rispetto alle sequenze corrette