Aufgaben SW09

Randomness

Maurin Donat Thalmann, Patrick Bucher, Pascal Kiser

17.04.2020

Aufgabe 1: Dragon Die

Regeln

Wähle eine Zahl zwischen 1 und 6

- Du hast 1W6
- Das Haus hat einen Spezialwürfel [Drache, 2, 3, 4, 5, 6]
- Beide würfeln
- Die höhere Augenzahl gewinnt
- ABER: Das Haus gewinnt immer, wenn es Drachen würfelt
- Wenn Du gewinnst, bekommst Du CHF 2
- Wenn das Haus gewinnt, verlierst Du CHF 1
- (Bei Unentschieden wird der Wurf wiederholt)

Lösung

Es gibt 36 mögliche Würfelkombinationen:

$$6 \cdot 6 = 36$$

Fünf davon sind Unentschieden, und fallen deshalb weg. Das Spiel hat somit 31 verschiedene mögliche Würfelkombinationen:

$$36 - 5 = 31$$

Insgesamt gibt es 10 positive und 21 negative Kombinationen, siehe folgende Tabelle:

H\P	1	2	3	4	5	6
🐉	☠	☠	☠	☠	☠	☠
2	☠	↱	💰	💰	💰	💰
3	☠	☠	↱	💰	💰	💰
4	☠	☠	☠	↱	💰	💰
5	☠	☠	☠	☠	↱	💰
6	☠	☠	☠	☠	☠	↱

Daraus kann man einfach die erwarteten Gewinne ableiten:

```
   D  2  3  4  5  6
1 -1 -1 -1 -1 -1 -1 = -6
2 -1  0 -1 -1 -1 -1 = -5
3 -1 +2  0 -1 -1 -1 = -2
4 -1 +2 +2  0 -1 -1 = +1
5 -1 +2 +2 +2  0 -1 = +4
6 -1 +2 +2 +2 +2  0 = +7
                    = -1
```

Man sieht sofort: das Spielen lohnt sich langfristig nicht. Der erwartete Gewinn pro Runde beträgt:

$$\left(\frac{10}{31} \cdot 2\right) - \left(\frac{21}{31} \cdot 1\right) = -0.\overline{032258064516129}$$

Der erwartete Gewinn ist somit kleiner als der erwartete Gewinn beim Nicht-Spielen (≈ 0). Es sollte also nicht gespielt werden.

Simulation

```python
#!/usr/bin/env python3

import random


def simulate(n=1):
    outcome = 0
    player_choices = [1, 2, 3, 4, 5, 6]
    house_choices = ['D', 2, 3, 4, 5, 6]
    for i in range(n):
```

```
        player = random.choice(player_choices)
        house = random.choice(house_choices)
        if house == 'D' or house > player:
            outcome -= 1
        elif player > house:
            outcome += 2
        else:
            continue  # draw
    return outcome


for i in [1, 10, 100, 1000, 10000, 100000, 1000000]:
    result = simulate(i)
    outcome = result/i
    print(f'simulate {i} times: outcome={outcome}')
```

Ausgabe:

```
simulate 1 times: outcome=2.0
simulate 10 times: outcome=0.0
simulate 100 times: outcome=-0.07
simulate 1000 times: outcome=0.036
simulate 10000 times: outcome=-0.0271
simulate 100000 times: outcome=-0.02963
simulate 1000000 times: outcome=-0.02836
```

Aufgabe 2: Chuck-a-Luck

Regeln

- Wähle eine Zahl zwischen 1 und 6
- Würfel 3W6
- Für jeden Würfel, der Deine Zahl zeigt, verdienst Du CHF 1
- Du kannst also mit einem Wurf 1, 2 oder 3 Franken gewinnen!
- Falls kein Würfel die Zahl zeigt, verlierst Du CHF 1

Lösung

Für dieses Spiel gibt es pro Runde genau vier Möglichkeiten:

1. Genau eine Zahl stimmt
2. Genau zwei Zahlen stimmen

3. Genau drei Zahlen stimmen
4. Keine Zahl stimmt.

Der erste Fall hat drei Permutationen, weshalb wir die einzelnen Wahrscheinlichkeiten nochmals mit 3 multiplizieren:

$$3 \cdot \frac{1}{6} \cdot \frac{5}{6} \cdot \frac{5}{6} = \frac{75}{216}$$

Der zweite analog dazu:

$$3 \cdot \frac{1}{6} \cdot \frac{1}{6} \cdot \frac{5}{6} = \frac{15}{216}$$

Beim dritten gibt aber nur eine Permutation:

$$\frac{1}{6} \cdot \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{216}$$

Ebenso beim vierten Fall:

$$\frac{5}{6} \cdot \frac{5}{6} \cdot \frac{5}{6} = \frac{125}{216} = 0.578\overline{703}$$

Nun kann man den erwarteten Gewinn folgendermassen berechnen:

$$\left(\frac{75}{216} \cdot 1\right) + \left(\frac{15}{216} \cdot 2\right) + \left(\frac{1}{216} \cdot 3\right) + \left(\frac{125}{216} \cdot -1\right) = -0.078\overline{703}$$

Fazit: Auch dieses Spiel lohnt sich nicht.

Simulation

```python
#!/usr/bin/env python3

import random


def simulate(n=1):
    total = 0
    for i in range(n):
        outcome = 0
        number = random.randint(1, 7)
        dice = [0, 0, 0]
```

```
        for j in range(3):
            dice[j] = random.randint(1, 7)
            if dice[j] == number:
                outcome += 1
        if outcome == 0:
            outcome -= 1
        total += outcome
    return total


for i in [1, 10, 100, 1000, 10000, 100000, 1000000]:
    result = simulate(i)
    outcome = result/i
    print(f'simulate {i} times: outcome={outcome}')
```

Ausgabe:

```
simulate 1 times: outcome=-1.0
simulate 10 times: outcome=0.1
simulate 100 times: outcome=-0.35
simulate 1000 times: outcome=-0.259
simulate 10000 times: outcome=-0.19
simulate 100000 times: outcome=-0.20489
simulate 1000000 times: outcome=-0.203498
```