Aufgabe 4: Würfelglück

Team-ID: 00155

Team: Opensourcehacker

Bearbeiter/-innen dieser Aufgabe:
Ron Jost

22. November 2021

Inhaltsverzeichnis

Lösungsidee

Zu Beginn werden alle mögliche Paarungen der Würfel berechnet. Dann wird für jeden Würfel, die Hälfte der vom Nutzer geforderten Wiederholungen, als Anfangsspieler und die andere Hälfte als zweiter Spieler simuliert. Die Spielsimulation habe ich unterteilt. Zu Beginn habe ich das Spielfeld in Zielfelder, Startboxen und Spielfeld unterteilt. Das Spielfeld habe ich als 1d Array repräsentiert, welches 40 Felder hat und das Gelbe Startfeld als Array[0] hat, die Zielfelder und Starboxen ebenso. Spieler 1, der der anfängt spielt immer mit Gelb, Spieler 2 immer mit Rot. Zu Beginn wird das gelbe und das rote Startfeld mit der Zahl 1, respektive 2 belegt und eine 1, respektive 2 aus den Boxen, mit einer 0 ersetzt. Die unbelegten Felder des Spielfelds werden durch Nullen repräsentiert. Ein Zug läuft wie folgt ab: Spieler x würfelt, d.h. es wird zufällig eine der Würfelseiten ausgewählt. Darauf folgt der Bewegungsalgorithmus. Würfelt Spieler 1, wird zuerst gecheckt, ob die Augenzahl 0 entspricht, denn dann wird der Zug übersprungen. Würfelt Spieler 1 keine 6 und keine 0, wird zunächst überprüft, ob sein Startfeld besetzt ist, d.h Spielfeld[0] = 1 gegeben ist, denn dann ist der erste Schritt, zu überprüfen, ob Spielfeld[0] != 1 ist, und es somit möglich ist, die Spielfigur dorthin zu bewegen. Falls dem so ist, ändert man den Wert an dem Index zu 1 und ändert den Index am Startfeld. Steht auf dem Feld auf das der Würfel uns führt, dann wird in diesem Fall die Position mit 1 überschrieben, die Startposition wird frei, also zu null. Nach diesem Prinzip des Schlagens und Bewegens funktioniert ein Zug immer. Steht keine Figur auf dem Startfeld, muss mit der vorderst möglichen Spielfigur bewegt werden. So betrachtet man zunächst die Zielfelder, steht dort eine Figur und lässt sich mit der Augenzahl weiterbewegen, bewegt man diese Figur und der Zug ist beendet. Steht keine Figur im Ziel, nimmt man die am weitesten fortgeschrittene Figur , welche

man bewegen kann, d.h das man sie entweder ins Ziel, auf ein freies Feld oder eine andere Figur schlagen kann. Die Zugmöglichkeiten werden, sollte eine 6 zufällig als Augenzahl ausgewählt werden, um die Möglichkeit ergänzt, eine Figur aus der Box ins Spiel zu bringen, dafür muss das Startfeld frei sein. Sollte dies der Fall sein, wird eine 1 in der Box durch eine 0 ersetzt. Generell ist man, wenn man eine 6 gewürfelt hat, nochmal dran und hat nochmal einen Zug.

Wenn Spieler 2 dran ist, funktioniert der Zug genau gleich, außer, dass anstatt Einsen, Zweien genutzt werden. Wie kann das sein? Im Voraus eines jeden Zuges von Spieler2, wird das Feld einmal so rearrangiert, dass an Position 0 des Feldes sich die Startposition von Rot befindet und an Position 39 das rote Ziel. Dann führt man den Zug aus, und arrangiert dass Feld dann wieder in die Spieler 1 Form zurück. Die Spieler wechseln sich mit ihren Zügen so oft ab, bis ein Ziel voll ist, also eine Seite gewonnen hat, oder das vom Nutzer angegebene Zuglimit pro Spiel überschritten wurde und kein Sieger bestimmbar ist.

Umsetzung

Das Programm wird in Python implementiert. Als Kommandozeilenparameter werden die einzulesende Würfeldatei, die Anzahl der Wiederholungen pro Würfelmatch und die maximale Zuganzahl pro Spiel übergeben. Dann wird zu Beginn mithilfe des Moduls itertools per itertools.combinations alle möglichen Würfelpaarungen berechnet. Per for loop wird dann für jede dieser Partien, das Spiel x mal wiederholt. x/2 fängt Würfel 1 an, die andere Hälfte fängt Würfel 2 an. Die Spielsimulation erfolgt per der Funktion ***Spiel_simulation(player1würfel, player2würfel, start_spieler (1oder2), Anzahl_max_Züge_pro_Spiel).*** Die Funktion ruft zu erst die Funktion ***Feld()*** auf, welche die Spielerboxen, die Ziele und das initiale Spielfeld voller Nullen erstellt. Die Spielerboxen sind Arrays, welche 4x die 1 respektive 4x die 2 für Spieler 2 enthalten. Die Ziele werden in einem Dictionary in der Form Farbe: (Feld, [0,0,0,0](Zielarray)) gespeichert. Daraufhin wird in der Spielsimulationsfunktion die Funktion ***Spiel(Start_Spieler, Feld, player1_würfel, player2_würfel, Anzahl_Züge_pro_Spiel_max)*** ausgeführt, mit den jeweilig vorher ermittelten und eingegebenen Werten. Zu Beginn der Funktion **Spiel** wird eine Figur aus der Box von Spieler1 und eine Figur aus der Box von Spieler 2 entnommen und auf das Spielfeld, auf die Startpositionen gestellt. Dann wird je nachdem ob Spieler1 oder Spieler2 dran ist , die Funktion ***move(Start_Spieler, Feld, player_würfel, player1box, player2box, Ziel)*** aufgerufen, welche den Zug nach dem in der Lösungsidee dargelegten Pfad, berechnet. Dies wird abwechselnd solange gemacht, bis ein Ziel voll ist, oder die zulässige Zuganzahl überschritten wurde. Besonders Interessant ist hierbei die Umrechnung des Spieler 1 Feldes in das von Spieler 2 und wieder zurück in der Funktion move:

Falls Spieler 2 dran ist: Zu Beginn
Umkonstruktion vom Feld: (Wird am Ende des Zuges immer wieder umkonstruiert)
```
Spieler_2_Feld = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Ziel = (39, Ziel[1])
for element in range(20,40):
   Spieler_2_Feld[element-20] = Feld[element]
for element in range(20):
   Spieler_2_Feld[element+20] = Feld[element]
```

Am Ende des Zuges:
Umwandlung Feld_Spieler 2:
```
Feld = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
for element in range(20, 40):
   Feld[element -20] = Spieler_2_Feld[element]
for element in range(20):
   Feld[element + 20] = Spieler_2_Feld[element]
```

Am Ende, werden alle Spielergebnisse in Form von 1 oder 2 in einem Array für jedes Match gespeichert. Daraus errechnet man dann bei der Ausgabe dann die Gewinnwahrscheinlichkeit des Würfels gegen den anderen.

Beispiele

python3 Aufgabe4.py wuerfel0.txt 1000 500

Würfel 1 : [1, 2, 3, 4, 5, 6]36.8% W Quote vs Würfel 2: [1, 1, 1, 6, 6, 6]63.2% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]58.1% W Quote vs Würfel 2: [1, 2, 3, 4]41.9% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]40.3% W Quote vs Würfel 2: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]59.7% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]38.7% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]61.3% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]48.9% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]51.1% W Quote

Würfel 1 : [1, 1, 1, 6, 6, 6]80.3% W Quote vs Würfel 2: [1, 2, 3, 4]19.7% W Quote

Würfel 1 : [1, 1, 1, 6, 6, 6]63.7% W Quote vs Würfel 2: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]36.3% W Quote

Würfel 1 : [1, 1, 1, 6, 6, 6]58.3% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]41.7% W Quote

Würfel 1 : [1, 1, 1, 6, 6, 6]59.0% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]41.0% W Quote

Würfel 1 : [1, 2, 3, 4]0.0% W Quote vs Würfel 2: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]100.0% W Quote

Würfel 1 : [1, 2, 3, 4]0.0% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]100.0% W Quote

Würfel 1 : [1, 2, 3, 4]0.0% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]100.0% W Quote

Würfel 1 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]32.3% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]67.7% W Quote

Würfel 1 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]43.6% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]56.4% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]39.8% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]60.2% W Quote

python3 Aufgabe4.py wuerfel1.txt 1000 500

Würfel 1 : [1, 2, 3, 4, 5, 6]33.3% W Quote vs Würfel 2: [2, 3, 4, 5, 6, 7]66.7% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]31.8% W Quote vs Würfel 2: [3, 4, 5, 6, 7, 8]68.2% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]34.7% W Quote vs Würfel 2: [4, 5, 6, 7, 8, 9]65.3% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]40.3% W Quote vs Würfel 2: [5, 6, 7, 8, 9, 10]59.7% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]43.2% W Quote vs Würfel 2: [6, 7, 8, 9, 10, 11]56.8% W Quote

Würfel 1 : [2, 3, 4, 5, 6, 7]32.2% W Quote vs Würfel 2: [3, 4, 5, 6, 7, 8]63.8% W Quote

Würfel 1 : [2, 3, 4, 5, 6, 7]29.3% W Quote vs Würfel 2: [4, 5, 6, 7, 8, 9]63.5% W Quote

Würfel 1 : [2, 3, 4, 5, 6, 7]31.3% W Quote vs Würfel 2: [5, 6, 7, 8, 9, 10]60.1% W Quote

Würfel 1 : [2, 3, 4, 5, 6, 7]34.4% W Quote vs Würfel 2: [6, 7, 8, 9, 10, 11]55.2% W Quote

Würfel 1 : [3, 4, 5, 6, 7, 8]25.0% W Quote vs Würfel 2: [4, 5, 6, 7, 8, 9]62.4% W Quote

Würfel 1 : [3, 4, 5, 6, 7, 8]23.6% W Quote vs Würfel 2: [5, 6, 7, 8, 9, 10]59.3% W Quote

Würfel 1 : [3, 4, 5, 6, 7, 8]26.6% W Quote vs Würfel 2: [6, 7, 8, 9, 10, 11]54.5% W Quote

Würfel 1 : [4, 5, 6, 7, 8, 9]22.0% W Quote vs Würfel 2: [5, 6, 7, 8, 9, 10]59.4% W Quote

Würfel 1 : [4, 5, 6, 7, 8, 9]25.0% W Quote vs Würfel 2: [6, 7, 8, 9, 10, 11]54.5% W Quote

Würfel 1 : [5, 6, 7, 8, 9, 10]13.4% W Quote vs Würfel 2: [6, 7, 8, 9, 10, 11]53.6% W Quote

python3 Aufgabe4.py wuerfel2.txt 1000 500

Würfel 1 : [1, 1, 1, 1, 1, 6]23.0% W Quote vs Würfel 2: [1, 1, 1, 1, 6, 6]77.0% W Quote

Würfel 1 : [1, 1, 1, 1, 1, 6]5.7% W Quote vs Würfel 2: [1, 1, 1, 6, 6, 6]94.3% W Quote

Würfel 1 : [1, 1, 1, 1, 1, 6]2.5% W Quote vs Würfel 2: [1, 1, 6, 6, 6, 6]97.5% W Quote

Würfel 1 : [1, 1, 1, 1, 1, 6]14.3% W Quote vs Würfel 2: [1, 6, 6, 6, 6, 6]85.7% W Quote

Würfel 1 : [1, 1, 1, 1, 6, 6]50.3% W Quote vs Würfel 2: [1, 1, 1, 6, 6, 6]49.7% W Quote

Würfel 1 : [1, 1, 1, 1, 6, 6]39.3% W Quote vs Würfel 2: [1, 1, 6, 6, 6, 6]60.7% W Quote

Würfel 1 : [1, 1, 1, 1, 6, 6]44.6% W Quote vs Würfel 2: [1, 6, 6, 6, 6, 6]55.4% W Quote

Würfel 1 : [1, 1, 1, 6, 6, 6]66.4% W Quote vs Würfel 2: [1, 1, 6, 6, 6, 6]33.6% W Quote

Würfel 1 : [1, 1, 1, 6, 6, 6]56.3% W Quote vs Würfel 2: [1, 6, 6, 6, 6, 6]43.7% W Quote

Würfel 1 : [1, 1, 6, 6, 6, 6]78.8% W Quote vs Würfel 2: [1, 6, 6, 6, 6, 6]21.2% W Quote

python3 Aufgabe4.py wuerfel3.txt 1000 500

Würfel 1 : [1, 2, 5, 6]48.8% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6]51.2% W Quote

Würfel 1 : [1, 2, 5, 6]43.1% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8]56.9% W Quote

Würfel 1 : [1, 2, 5, 6]47.5% W Quote vs Würfel 2: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]52.5% W Quote

Würfel 1 : [1, 2, 5, 6]44.2% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]55.8% W Quote

Würfel 1 : [1, 2, 5, 6]51.7% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]48.3% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]34.6% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8]65.4% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]39.9% W Quote vs Würfel 2: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]60.1% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]37.4% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]62.6% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6]49.5% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]50.5% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6, 7, 8]40.1% W Quote vs Würfel 2: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]59.9% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6, 7, 8]35.6% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]64.4% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6, 7, 8]45.4% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]54.6% W Quote

Würfel 1 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]29.9% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]70.1% W Quote

Würfel 1 : [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]41.5% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]58.5% W Quote

Würfel 1 : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]40.7% W Quote vs Würfel 2: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]59.3% W Quote

Quellcode

```
# BWINF: Nr. 40
# Runde: 1
# Aufgabe: 4
# Autor: Ron Jost
# Team: Opensourcehacker (00155)
import sys
import itertools
import random
würfel_datei = sys.argv[1]

würfel_daten = open(würfel_datei, 'r').readlines()

würfel_daten_bearbeitet = []

for line in würfel_daten:

    würfel_daten_bearbeitet.append(line.replace('\n', ''))

Anzahl_Würfel = würfel_daten_bearbeitet[0]

Würfel = {}

for i in range(int(Anzahl_Würfel)):

    würfel_info = würfel_daten_bearbeitet[i+1]

    würfel_seiten = würfel_info[:würfel_info.find(' ')]

    würfel_info = würfel_info[würfel_info.find(' ')+1:].split()

    würfel_info = [int(x) for x in würfel_info]

    Würfel[i] = (int(würfel_seiten), würfel_info)
```

```
# Berechne mögliche Partien:
Partien = []
for key in Würfel:
    Partien.append(key)
Partien = list(itertools.combinations(Partien,2))
for Partie in Partien:
    player1_würfel = Würfel[Partie[0]][1]
    player2_würfel = Würfel[Partie[1]][1]
#(Würfel)
def Spielfeld():
    # Haus der Figuren, die auf ins Spiel bringen warten:
    player1_Box = [1,1,1,1]
    player2_Box = [2,2,2,2]
    # Spielfeld Startpunkt der Indizes ist das Gelbe A Feld und dann in Uhrzeigerrichtung:
    Feld = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    Start_indizes = {'Gelb': 0,
                     'Grün': 10,
                     'Rot': 20,
                     'Schwarz': 30
                     }
    Ziele = {'Grün': (9, [0,0,0,0]),
             'Rot': (19, [0,0,0,0]),
             'Schwarz': (29, [0,0,0,0]),
             'Gelb': (39,[0,0,0,0])
            }
    return player1_Box, player2_Box, Feld, Ziele, Start_indizes
def move(Spieler, Feld, Würfel, player1_box, player2_box, Ziel):
    '''
```

```
    Regeln:
    Der vorderste Stein wird gezogen
    Man kann wenn kein Stein bewegbar ist, auf dem Feld im Zielfeld nach vorne
    Solange noch andere Figuren in der Box warten muss das Start Feld sofort freigemacht werden
    Besonderheit 6:
    Man muss einen Stein aus der Box ins Feld bringen, wenn das Feld frei ist + in der box etwas ist
    -> Man kann nach dem Zug direkt nochmal ziehen
    Es muss immer geschlagen werden --> Back to box
    '''
    if Spieler == 1:
        # Wird mit Gelb gespielt; Würfel 1
        Augenzahl = random.choice(Würfel)
        # Wenn keine 6 und keine 0 ist
        if Augenzahl != 6 and Augenzahl != 0:
            # Erster check, ob Startfeld belegt ist und man sich wegbewegen kann
            if Feld[0] == 1 and Feld[Augenzahl] != 1:
                # Ohne Schlagen bewegen
                if Feld[Augenzahl] == 0:
                    Feld[Augenzahl] = 1
                    Feld[0] = 0
                # Mit Schlagen bewegen
                else:
                    Feld[Augenzahl] = 1
                    player2_box[player2_box.index(0)] = 2
                    Feld[0] = 0
            # Vordersten Stein bewegen
            else:
                failed_Ziel = False
                # Höchste Steine: Im Ziel:
```

```
        Ziel_indizes = [i for i, x in enumerate(Ziel[1]) if x == 1][::-1]
        if len(Ziel_indizes) != 0:
            for element in Ziel_indizes:
                if (element + Augenzahl) > len(Ziel[1])-1:
                    pass
                else:
                    if Ziel[1][element + Augenzahl] == 0:
                        Ziel[1][element+Augenzahl] = 1
                        Ziel[1][element] = 0
                        break
            failed_Ziel = True
        # Wenn man keinen Stein im Ziel bewegen kann
        if len(Ziel_indizes) == 0 or failed_Ziel == True:
            indices = [i for i, x in enumerate(Feld) if x == 1][::-1]
            for Stein in indices:
                # Vor dem Ziel move ohne Schlagen
                if Stein + Augenzahl <= Ziel[0]:
                    if Feld[Stein + Augenzahl] == 0:
                        Feld[Stein + Augenzahl] = 1
                        Feld[Stein] = 0
                        break
                    # Gegnerische Figur
                    if Feld[Stein+ Augenzahl] == 2:
                        # SCHLAGEN
                        Feld[Stein + Augenzahl] = 1
                        player2_box[player2_box.index(0)] = 2
                        Feld[Stein] = 0
                        break
                    # Falls keins von beiden, passiert nichts, nächster Stein
```

```
                else:
                    Züge_nach_Zielfeld = Augenzahl - ( Ziel[0] - Stein)
                    # Falls man über das Ziel herrausschießen würde
                    if Züge_nach_Zielfeld > 4:
                        pass
                    # Falls es passen würde
                    else:
                        if Ziel[1][Züge_nach_Zielfeld-1] == 1:
                            pass
                        else:
                            Ziel[1][Züge_nach_Zielfeld-1] = 1
                            Feld[Stein] = 0
                            break
    else:
        if Augenzahl == 0:
            pass
        if Augenzahl == 6:
            # Erster Check: Ob sich noch Figuren in der Box befinden:
            Box_indizes = [i for i, x in enumerate(player1_box) if x == 1]
            # Falls sich Figuren in der Box befinden + das Startfeld nicht belegt ist
            if len(Box_indizes) != 0 and Feld[0] != 1:
                # Falls es frei ist:
                if Feld[0] == 0:
                    Feld[0] = 1
                    player1_box[player1_box.index(1)] = 0
                # Falls geschlagen werden muss
                else:
                    Feld[0] = 1
                    player2_box[player2_box.index(0)] = 2
```

```
            player1_box[player1_box.index(1)] = 0
        # Falls sich keine Figuren mehr in der Box befinden oder das Feld belegt ist:
        else:
            indices = [i for i, x in enumerate(Feld) if x == 1]
            # Erster check, ob Startfeld belegt ist und man sich wegbewegen kann
            if Feld[0] == 1 and Feld[Augenzahl] != 1:
                # Ohne Schlagen bewegen
                if Feld[Augenzahl] == 0:
                    Feld[Augenzahl] = 1
                    Feld[0] = 0
                # Mit Schlagen bewegen
                else:
                    Feld[Augenzahl] = 1
                    Feld[0] = 0
                    player2_box[player2_box.index(0)] = 2
            # Vordersten Stein bewegen
            else:
                failed_Ziel = False
                # Höchste Steine: Im Ziel:
                Ziel_indizes = [i for i, x in enumerate(Ziel[1]) if x == 1][::-1]
                if len(Ziel_indizes) != 0:
                    for element in Ziel_indizes:
                        if (element + Augenzahl) > len(Ziel[1]) - 1:
                            pass
                        else:
                            if Ziel[1][element + Augenzahl] == 0:
                                Ziel[1][element + Augenzahl] = 1
                                Ziel[1][element] = 0
                                break
```

```
            failed_Ziel = True
        # Wenn man keinen Stein im Ziel bewegen kann
        if len(Ziel_indizes) == 0 or failed_Ziel == True:
            indices = [i for i, x in enumerate(Feld) if x == 1][::-1]
            for Stein in indices:
                # Vor dem Ziel move ohne Schlagen
                if Stein + Augenzahl <= Ziel[0]:
                    if Feld[Stein + Augenzahl] == 0:
                        Feld[Stein + Augenzahl] = 1
                        Feld[Stein] = 0
                        break
                    # Gegnerische Figur
                    if Feld[Stein + Augenzahl] == 2:
                        Feld[Stein + Augenzahl] = 1
                        player2_box[player2_box.index(0)] = 2
                        Feld[Stein] = 0
                        break
                    # Falls keins von beiden, passiert nichts, nächster Stein
                else:
                    Züge_nach_Zielfeld = Augenzahl - (Ziel[0] - Stein)
                    # Falls man über das Ziel herrausschießen würde
                    if Züge_nach_Zielfeld > 4:
                        pass
                    # Falls es passen würde
                    else:
                        if Ziel[1][Züge_nach_Zielfeld - 1] == 1:
                            pass
                        else:
                            Ziel[1][Züge_nach_Zielfeld - 1] = 1
```

```
                    Feld[Stein] = 0
                    break
            # Da Spieler 1 eine 6 gewürfelt hat, darf er nochmal
            move(1, Feld, Würfel, player1_box,player2_box, Ziel)


    if Spieler == 2:
        # Umkonstruktion von Feld: ( WIrd am Ende des Zuges immer wieder umkonstruiert)
        Spieler_2_Feld = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        Ziel = (39, Ziel[1])
        for element in range(20,40):
            Spieler_2_Feld[element-20] = Feld[element]
        for element in range(20):
            Spieler_2_Feld[element+20] = Feld[element]
        # Wird mit Rot gespielt
        Augenzahl = random.choice(Würfel)
        # Wenn keine 6 und keine 0 ist
        if Augenzahl != 6 and Augenzahl != 0:
            # Erster check, ob Startfeld belegt ist und man sich wegbewegen kann
            if Spieler_2_Feld[0] == 2 and Spieler_2_Feld[Augenzahl] != 2:
                # Ohne Schlagen bewegen
                if Spieler_2_Feld[Augenzahl] == 0:
                    Spieler_2_Feld[Augenzahl] = 2
                    Spieler_2_Feld[0] = 0
                    # Mit Schlagen bewegen
                else:
                    # SCHLAGEN
                    Spieler_2_Feld[Augenzahl] = 2
                    Spieler_2_Feld[0] = 0
```

```
            player1_box[player1_box.index(0)] = 1


        # Vordersten Stein bewegen
        else:
            failed_Ziel = False
            # Höchste Steine: Im Ziel:
            Ziel_indizes = [i for i, x in enumerate(Ziel[1]) if x == 2][::-1]
            if len(Ziel_indizes) != 0:
                for element in Ziel_indizes:
                    if (element + Augenzahl) > len(Ziel[1]) - 1:
                        pass
                    else:
                        if Ziel[1][element + Augenzahl] == 0:
                            Ziel[1][element + Augenzahl] = 2
                            Ziel[1][element] = 0
                            break
                failed_Ziel = True
                # Wenn man keinen Stein im Ziel bewegen kann
            if len(Ziel_indizes) == 0 or failed_Ziel == True:
                indices = [i for i, x in enumerate(Spieler_2_Feld) if x == 2][::-1]
                for Stein in indices:
                    # Vor dem Ziel move ohne Schlagen
                    if Stein + Augenzahl <= Ziel[0]:
                        if Spieler_2_Feld[Stein + Augenzahl] == 0:
                            Spieler_2_Feld[Stein + Augenzahl] = 2
                            Spieler_2_Feld[Stein] = 0
                            break
                        # Gegnerische Figur
                        if Spieler_2_Feld[Stein + Augenzahl] == 1:
```

```
                    # SCHLAGEN
                    Spieler_2_Feld[Stein + Augenzahl] = 2
                    Spieler_2_Feld[Stein] = 0
                    player1_box[player1_box.index(0)] = 1
                    break
                # Falls keins von beiden, passiert nichts, nächster Stein
            else:
                Züge_nach_Zielfeld = Augenzahl - (Ziel[0] - Stein)
                # Falls man über das Ziel herrausschießen würde
                if Züge_nach_Zielfeld > 4:
                    pass
                # Falls es passen würde
                else:
                    if Ziel[1][Züge_nach_Zielfeld - 1] == 2:
                        pass
                    else:
                        Ziel[1][Züge_nach_Zielfeld - 1] = 2
                        Spieler_2_Feld[Stein] = 0
                    break
    else:
        if Augenzahl == 0:
            pass
        if Augenzahl == 6:
            # Erster Check: Ob sich noch Figuren in der Box befinden:
            Box_indizes = [i for i, x in enumerate(player2_box) if x == 2]
            # Falls sich Figuren in der Box befinden + das Startfeld nicht belegt ist
            if len(Box_indizes) != 0 and Spieler_2_Feld[0] != 2:
                # Falls es frei ist:
                if Spieler_2_Feld[0] == 0:
```

```
            Spieler_2_Feld[0] = 2
            player2_box[Box_indizes[0]] = 0
        # Falls geschlagen werden muss
        else:
            Spieler_2_Feld[0] = 2
            player2_box[player2_box.index(2)] = 0
            player1_box[player1_box.index(0)] = 1
    # Falls sich keine Figuren mehr in der Box befinden oder das Feld belegt ist:
    else:
        # Erster check, ob Startfeld belegt ist und man sich wegbewegen kann
        if Spieler_2_Feld[0] == 2 and Spieler_2_Feld[Augenzahl] != 2:
            # Ohne Schlagen bewegen
            if Spieler_2_Feld[Augenzahl] == 0:
                Spieler_2_Feld[Augenzahl] = 2
                Spieler_2_Feld[0] = 0
                # Mit Schlagen bewegen
            else:
                Spieler_2_Feld[Augenzahl] = 2
                Spieler_2_Feld[0] = 0
                player1_box[player1_box.index(0)] = 1
        # Vordersten Stein bewegen
        else:
            failed_Ziel = False
            # Höchste Steine: Im Ziel:
            Ziel_indizes = [i for i, x in enumerate(Ziel[1]) if x == 2][::-1]
            if len(Ziel_indizes) != 0:
                for element in Ziel_indizes:
                    if (element + Augenzahl) > len(Ziel[1]) - 1:
                        pass
```

```
            else:
                if Ziel[1][element + Augenzahl] == 0:
                    Ziel[1][element + Augenzahl] = 2
                    Ziel[1][element] = 0
                    break
        failed_Ziel = True
        # Wenn man keinen Stein im Ziel bewegen kann
    if len(Ziel_indizes) == 0 or failed_Ziel == True:
        indices = [i for i, x in enumerate(Spieler_2_Feld) if x == 2][::-1]
        for Stein in indices:
            # Vor dem Ziel move ohne Schlagen
            if Stein + Augenzahl <= Ziel[0]:
                if Spieler_2_Feld[Stein + Augenzahl] == 0:
                    Spieler_2_Feld[Stein + Augenzahl] = 2
                    Spieler_2_Feld[Stein] = 0
                    break
                # Gegnerische Figur
                if Spieler_2_Feld[Stein + Augenzahl] == 1:
                    # SCHLAGEN
                    Spieler_2_Feld[Stein + Augenzahl] = 2
                    Spieler_2_Feld[Stein] = 0
                    player1_box[player1_box.index(0)] = 1
                    break
                # Falls keins von beiden, passiert nichts, nächster Stein
            else:
                Züge_nach_Zielfeld = Augenzahl - (Ziel[0] - Stein)
                # Falls man über das Ziel herrausschießen würde
                if Züge_nach_Zielfeld > 4:
                    pass
```

```
                    # Falls es passen würde
                    else:
                        if Ziel[1][Züge_nach_Zielfeld - 1] == 2:
                            pass
                        else:
                            Ziel[1][Züge_nach_Zielfeld - 1] = 2
                            Spieler_2_Feld[Stein] = 0
                        break
        # Umwandlung Feld_Spieler 2:
        Feld = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        for element in range(20, 40):
            Feld[element -20] = Spieler_2_Feld[element]
        for element in range(20):
            Feld[element + 20] = Spieler_2_Feld[element]
    return Feld, player1_box, player2_box, Ziel
def Spiel(Start_Spieler, Feld, player1_würfel, player2_würfel, Anzahl_Züge_pro_Spiel_max):
    # Spieler 1 spielt mit Gelb, Spieler 2 mit ROt
    Feld_Spielstart = Feld[2]
    Start_indizes = Feld[4]
    player1_Box = Feld[0]
    player2_Box = Feld[1]
    Ziele = Feld[3]
    Feld_Spielstart[Start_indizes['Gelb']] = 1
    Feld_Spielstart[Start_indizes['Rot']] = 2
    player1_Box[0] = 0
    player2_Box[0] = 0
    if Start_Spieler == 1:
        Sieg_1 = False
```

```
    Sieg_2 = False
    move_er_1 = move(1, Feld_Spielstart, player1_würfel, player1_Box, player2_Box,
Ziele['Gelb'])
    Feld = move_er_1[0]
    player1_Box = move_er_1[1]
    player2_Box = move_er_1[2]
    Ziel_gelb = move_er_1[3]
    move_er_2 = move(2, Feld, player2_würfel, player1_Box, player2_Box, Ziele['Rot'])
    Feld = move_er_2[0]
    player1_Box = move_er_2[1]
    player2_Box = move_er_2[2]
    Ziel_rot = move_er_2[3]
    # Maximal 1000 Züge pro Spiel
    züge_score = 0
    while Sieg_1 == False and Sieg_2 == False and züge_score <= Anzahl_Züge_pro_Spiel_max:
      move_er_1 = move(1, Feld, player1_würfel, player1_Box, player2_Box, Ziel_gelb)
      Feld = move_er_1[0]
      player1_Box = move_er_1[1]
      player2_Box = move_er_1[2]
      Ziel_gelb = move_er_1[3]
      if Ziel_gelb[1].count(1) == 4:
        Sieg_1 = True
        break
      move_er_2 = move(2, Feld, player2_würfel, player1_Box, player2_Box, Ziel_rot)
      Feld = move_er_2[0]
      player1_Box = move_er_2[1]
      player2_Box = move_er_2[2]
      Ziel_rot = move_er_2[3]
      if Ziel_rot[1].count(2):
        Sieg_2 = True
```

```
            break
        züge_score = züge_score + 1
    if Start_Spieler == 2:
        Sieg_1 = False
        Sieg_2 = False
        move_er_2 = move(2, Feld_Spielstart, player2_würfel, player1_Box, player2_Box,
Ziele['Rot'])
        Feld = move_er_2[0]
        player1_Box = move_er_2[1]
        player2_Box = move_er_2[2]
        Ziel_rot = move_er_2[3]
        move_er_1 = move(1, Feld, player1_würfel, player1_Box, player2_Box, Ziele['Gelb'])
        Feld = move_er_1[0]
        player1_Box = move_er_1[1]
        player2_Box = move_er_1[2]
        Ziel_gelb = move_er_1[3]
        züge_score = 0
        while Sieg_1 == False and Sieg_2 == False and züge_score <= Anzahl_Züge_pro_Spiel_max:
            move_er_2 = move(2, Feld, player2_würfel, player1_Box, player2_Box, Ziel_rot)
            Feld = move_er_2[0]
            player1_Box = move_er_2[1]
            player2_Box = move_er_2[2]
            Ziel_rot = move_er_2[3]
            if Ziel_rot[1].count(2) == 4:
                Sieg_2 = True
                break
            move_er_1 = move(1, Feld, player1_würfel, player1_Box, player2_Box, Ziel_gelb)
            Feld = move_er_1[0]
            player1_Box = move_er_1[1]
            player2_Box = move_er_1[2]
```

```
            Ziel_gelb = move_er_1[3]
            if Ziel_gelb[1].count(1) == 4:
                Sieg_1 = True
                break
            züge_score = züge_score + 1
    return Sieg_1, Sieg_2
def Spiel_simulation(player1_würfel, player2_würfel, Start_Spieler, Anzahl_max_Zug):
    Feld = Spielfeld()
    Sieg = Spiel(Start_Spieler, Feld, player1_würfel, player2_würfel, Anzahl_max_Zug)
    if Sieg[0]:
        return 1
    if Sieg[1]:
        return 2
Anzahl_wiederholungen_Partie = int(sys.argv[2])
Anzahl_Züge_pro_Spiel_max = int(sys.argv[3])
for Partie in Partien:
    Siege = []
    player1_würfel = Würfel[Partie[0]][1]
    player2_würfel = Würfel[Partie[1]][1]
    for i in range(int(Anzahl_wiederholungen_Partie/2)):
        Sieg = Spiel_simulation(player1_würfel, player2_würfel, 1, Anzahl_Züge_pro_Spiel_max)
        Siege.append(Sieg)
    for i in range(int(Anzahl_wiederholungen_Partie/2)):
        Sieg = Spiel_simulation(player1_würfel, player2_würfel, 2, Anzahl_Züge_pro_Spiel_max)
        Siege.append(Sieg)
    print('Würfel 1 : ' + str(player1_würfel) + str(round(((Siege.count(1)/len(Siege)) * 100), 3)) + '%
W Quote vs ' + 'Würfel 2: ' + str(player2_würfel) + str(round(Siege.count(2)/len(Siege) * 100, 3))+
'% W Quote')
```