Aufgabe 4: Würfelglück

Team-ID: 00023

Team-Name: Kallisto

Bearbeiter/-innen dieser Aufgabe: Alexander Lu

22.11.2021

Inhaltsverzeichnis:

Lösungsidee:

Es soll ein Programm geschrieben werden, welches das Mensch Ärger Dich Nicht Brettspiel für eine beliebige Anzahl an Durchlaufen simuliert. Dabei sollten verschiedene Würfel verwendet werden die der Würfel mit der höchsten Gewinnwahrscheinlichkeit zurückgegeben werden. Die Lösungsidee ist ganz simpel: Es wird das Spiel sehr oft simuliert mit zwei Spielern und verschiedenen Würfeln und es wird jedem Würfel einen Index zugeordnet, der die Gewinnanzahl der jeweiligen Würfel repräsentiert. Am Ende wird dann der Würfel empfohlen, der die höchste Gewinnanzahl hat.

Umsetzung:

Die Implementierung erfolgt in der Skripting Sprache Python, weil es schnell gehen soll. Die Klasse Aufgabe wird implementiert und enthält Hilfsmethoden für die Eingabe und Eingabeformatierung. Die Einstiegsmethode oder auch „Main“ Methode *main* beginnt den Lösungsvorgang indem sie eine Eingabedatei, als Argument, mithilfe der Methode *readInput* einliest und die Eingabe in das geeignete Format formatiert. So entsteht die Instanz variable *WUERFEL* welche in einer Matrix Liste jeweils ein Würfel repräsentiert. Die Methode *main* ruft die Methode *berechneWahrscheinlichkeiten* auf, die jeweils, für eine gegeben beliebige Anzahl, Spiele zwischen zweier Spieler simuliert. Eine ganze Spielreihe besteht normalerweise aus !Wurfelanzahl Spielen, da jeder Würfel mit jedem anderem Würfel spielt. Hierbei wäre es noch möglich, gleiche Würfel gegen gleiche Würfel auszulassen, da die Gewinnwahrscheinlichkeit auf beide Seiten auf durchschnittlich 50% liegt. Außerdem werden Partien nicht gespielt, falls beide Würfel keine 6 haben oder keine 1 weil Partien mit solchen Würfeln nicht unbedingt Spielbar sind. Die Methode *setSpieler*, die vor der Spielsimulation aufgerufen wird, setzt einen Anfangszustand für beide Spieler, bzw setzt die SPIELER_STATUS variable auf ihren Anfangswert. Die Variable SPIELER_STATUS beinhaltet Informationen wie die Anzahl der Spielfiguren im B-FELD oder die Position aller Spielfiguren, den Würfel eines Spielers oder das letzte gewinn Feld. Die Spielsimulation läuft folgendermaßen ab:

In einer unendlichen Schleife wird pro Durchgang je ein Zug gespielt und die Variable **amZug** wird am ende jedes Durchlaufes auf den gegen Spieler gesetzt. Als nächstes werden die Spielinformationen des Spielers und Gegenspielers aufgerufen und es wird der Würfel zufällig geworfen. In der bedingten Anweisung wird gecheckt, falls der Würfelwert 6 ist und sich noch Spielerfiguren auf der Position 0 befinden, welche das B-FELD repräsentiert. Ist dies der Fall, so ist der Spieler gezwungen diese Spielfiguren auf die Position 1 zu setzen. Falls die Position 1 schon besetzt wird, wird im zweiten Teil der bedingten Anweisung die Spielfigur auf Position 1 um 6 nach vorne verschoben. Falls der Würfelwert keine 6 ist, so wird die Liste aller Positionen der Spielfiguren absteigend sortiert und es wird versucht die Spielfigur mit der weitesten Position zu bewegen. Ist dies nicht möglich, da z.B. kein Spieler draußen ist oder sich schon im Endfeld befindet und die Würfelzahl nicht passt, so wird die Spielerfigur mit der nächst höheren Position bewegt, falls das nicht geht wird solange die nächst höhere Spielfigur gesucht, die bewegt werden kann. Falls sich eine Spielerfigur im letzten Spielfeld befindet, so gilt diese Spielfigur als fertig und wird auf -1 gesetzt, während die Variable LAST_POSITION um -1 addiert wird. Nachdem die Würfelaktion und Spielfigurenaktion durchgeführt wurde, so wird überprüft, ob sich eine Gegnerspielfigur bereits auf dem Feld befindet, falls ja, wird sie wieder in das B-FELD gekickt. Als letztes wird der Fall abgedeckt, dass wenn ein Spieler eine 6 würfelt, der Spieler noch einmal würfeln muss. Als letzter Schritt, wie bereits erwähnt, wird dem nächsten Spieler den Zug übergeben.

Beispiele:

Input:
6
6 1 2 3 4 5 6
6 1 1 1 6 6 6
4 1 2 3 4
10 0 1 2 3 4 5 6 7 8 9
12 1 2 3 4 5 6 7 8 9 10 11 12
20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Output:

Für Mensch ärgere dich empfiehlt sich von den gegebenen Wuerfeln dieser Wuerfel am meisten:
[6, 1, 1, 1, 6, 6, 6]
Dieser Wuerfel hat von 70 Spielen, 22 gewonnen. Das ist eine Sieges Wahrscheinlichkeit von 31.43%
Die Wahrscheinlichkeitsverteilung ist: [17, 22, 0, 11, 11, 9], 0 da keine 6 in einem Würfel ist und man somit nicht aus dem B-FELD kommt.

Input:
6
6 1 2 3 4 5 6
6 2 3 4 5 6 7
6 3 4 5 6 7 8
6 4 5 6 7 8 9
6 5 6 7 8 9 10
6 6 7 8 9 10 11
Output:
Für Mensch ärgere dich empfiehlt sich von den gegebenen Wuerfeln dieser Wuerfel am meisten:
[6, 1, 2, 3, 4, 5, 6]

Dieser Wuerfel hat von 1320 Spielen, 898 gewonnen. Das ist eine Sieges Wahrscheinlichkeit von 68.03%
Die Wahrscheinlichkeitsverteilung ist: [898, 121, 111, 91, 61, 38]

Input:

6
4 1 2 5 6
6 1 2 3 4 5 6
8 1 2 3 4 5 6 7 8
10 0 1 2 3 4 5 6 7 8 9
12 1 2 3 4 5 6 7 8 9 10 11 12
20 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Output:

Für Mensch ärgere dich empfiehlt sich von den gegebenen Wuerfeln dieser Wuerfel am meisten: [6, 1, 2, 3, 4, 5, 6]
Dieser Wuerfel hat von 1440 Spielen, 394 gewonnen. Das ist eine Sieges Wahrscheinlichkeit von 27.36%
Die Wahrscheinlichkeitsverteilung ist: [332, 394, 251, 204, 181, 78]

Input (Selfmade):

6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6
1 2 3 4 5 6

Output:

Für Mensch ärgere dich empfiehlt sich von den gegebenen Wuerfeln dieser Wuerfel am meisten: [1, 2, 3, 4, 5, 6]
Dieser Wuerfel hat von 1440 Spielen, 251 gewonnen. Das ist eine Sieges Wahrscheinlichkeit von 17.43%
Die Wahrscheinlichkeitsverteilung ist: [251, 249, 234, 232, 232, 242]
Für mehr versuche würde sich die maximale Sieges Wahrscheinlichkeit nach 1/6 nähern

Quellcode:

```
import sys
sys.setrecursionlimit(10000)

import random
class Aufgabe:
    def __init__(self):
        self.WUERFEL = []
        self.SPIELER_STATUS = {}

    def _setSpieler(self, spieler1, wuerfel1, spieler2, wuerfel2):
        self.SPIELER_STATUS = {
            "schwarz":  {
                "B-FELD"        : 4,
                "WUERFEL"       : [],
```

```
                "POSITIONEN"    : [0] * 4,
                "LAST_POSITION" : 44
            },
            "gruen":    {
                "B-FELD"        : 4,
                "WUERFEL"       : [],
                "POSITIONEN"    : [0] * 4,
                "LAST_POSITION" : 44
            }
        }

        self.SPIELER_STATUS[spieler1]["WUERFEL"] = wuerfel1
        self.SPIELER_STATUS[spieler2]["WUERFEL"] = wuerfel2

    def _getErgebnis(self, wuerfel):
        return random.choice(wuerfel)

    def _getIndex(self, list, value):
        return list.index(value)

    def _checkGegenspieler(self, indexOfSpieler, amZug, naechsterZug):
        """

            Check ob sich ein Gegenspieler auf dem Feld befindet das man
betreten will.
            Falls ja, wird der Gegenspieler zurück auf das B-Feld gestoßten
und man selbst
            schreitet auf das Feld hinauf

        """

        spielerStats = self.SPIELER_STATUS[amZug]
        gegenSpielerStats = self.SPIELER_STATUS[naechsterZug]

        positionGegenspieler = spielerStats["POSITIONEN"][indexOfSpieler] + 20
        positionGegenspieler2 = spielerStats["POSITIONEN"][indexOfSpieler] -
20
        if positionGegenspieler in gegenSpielerStats["POSITIONEN"]:
            gegenSpielerStats["POSITIONEN"][self._getIndex(gegenSpielerStats["
POSITIONEN"], positionGegenspieler)] = 0
            gegenSpielerStats["B-FELD"] += 1

        if positionGegenspieler2 in gegenSpielerStats["POSITIONEN"]:
            gegenSpielerStats["POSITIONEN"][self._getIndex(gegenSpielerStats["
POSITIONEN"], positionGegenspieler2)] = 0
            gegenSpielerStats["B-FELD"] += 1

    def _simulierSpiel(self, startSpieler):
        amZug = startSpieler
```

```
        while True:
            if amZug == "schwarz":
                naechsterZug = "gruen"
            else:
                naechsterZug = "schwarz"

            # Spiel spiel bis gewinner
            spielerStats = self.SPIELER_STATUS[amZug]
            wuerfelWert = self._getErgebnis(spielerStats["WUERFEL"])

            #  check if winner

            """

                Falls es noch Spieler im B Feld gibt und es keine gleichen
Spieler
                sich auf dem selben Feld befinden, dann bekommt ein B Feld
Spieler
                die Position 1

            """

            if wuerfelWert == 6 and 0 in spielerStats["POSITIONEN"]:
                if 1 not in spielerStats["POSITIONEN"]:
                    spielerStats["B-FELD"] -= 1
                    indexOfSpieler =
self._getIndex(spielerStats["POSITIONEN"], 0)
                    spielerStats["POSITIONEN"][indexOfSpieler] = 1
                else:
                    indexOfSpieler =
self._getIndex(spielerStats["POSITIONEN"], 1)
                    spielerStats["POSITIONEN"][indexOfSpieler] += wuerfelWert

                #self._checkGegenspieler(indexOfSpieler, amZug, naechsterZug)
            else:
                positionen = sorted(spielerStats["POSITIONEN"][:],
reverse=True)

                for i in range(len(positionen)):
                    if  positionen[i] == 0 or positionen[i] == -1 or \
                        wuerfelWert > spielerStats["LAST_POSITION"] -
positionen[i] or \
                        positionen[i] + wuerfelWert in
spielerStats["POSITIONEN"]:
                        continue

                    indexOfSpieler =
self._getIndex(spielerStats["POSITIONEN"], positionen[i])
                    spielerStats["POSITIONEN"][indexOfSpieler] += wuerfelWert
```

```
                    self._checkGegenspieler(indexOfSpieler, amZug,
naechsterZug)

                    break

                while spielerStats["LAST_POSITION"] in
spielerStats["POSITIONEN"]:
                    spielerStats["POSITIONEN"][self._getIndex(spielerStats["PO
SITIONEN"], spielerStats["LAST_POSITION"])] = -1
                    spielerStats["LAST_POSITION"] -= 1

            if wuerfelWert == 6:
                return self._simulierSpiel(amZug)

            if max(spielerStats["POSITIONEN"]) == -1:
                return amZug

            amZug = naechsterZug

    def _readInput(self, file):
        f = open("Aufgabe 4/assets/" + file, "r")
        content = f.read()
        f.close()

        data = content.splitlines()

        for i in range(0, len(data)):
            if(i == 0):
                continue
            else:
                wuerfel = data[i].split(" ")
                if len(wuerfel) != 0:
                    self.WUERFEL.append(list(map(int, data[i].split(" "))))

    def _berechneWahrscheinlichkeiten(self, durchlaeufe):W
        Gewinner = [0] * len(self.WUERFEL)
        for _ in range(durchlaeufe):
            for index in range(len(self.WUERFEL)):
                wuerfel = self.WUERFEL[index]
                for index2 in range(len(self.WUERFEL)):
                    wuerfel2 = self.WUERFEL[index2]
                    if 6 not in wuerfel and 6 not in wuerfel2 or (1 not in
wuerfel and 1 not in wuerfel2):
                        continue

                    for i in range(2):
                        if i == 0:
                            spieler = "schwarz"
                            gegenSpieler = "gruen"
                        elif i == 1:
```

```
                            spieler = "gruen"
                            gegenSpieler = "schwarz"

                        self._setSpieler(spieler, wuerfel, gegenSpieler,
wuerfel2)
                        gewinner = self._simulierSpiel("schwarz")

                        if gewinner == spieler:
                            Gewinner[index] += 1
                        elif gewinner == gegenSpieler:
                            Gewinner[index2] += 1

        self._formatierLoesung(Gewinner)

    def _formatierLoesung(self, wahrscheinlichkeiten):
        spielAnzahl = sum(wahrscheinlichkeiten)
        meisteSiege = max(wahrscheinlichkeiten)
        wuerfelMeisteSiege =
self.WUERFEL[wahrscheinlichkeiten.index(meisteSiege)]

        print("Für Mensch ärgere dich empfiehlt sich von den gegebenen
Wuerfeln dieser Wuerfel am meisten:")
        print(wuerfelMeisteSiege)
        print(f"Dieser Wuerfel hat von {spielAnzahl} Spielen, {meisteSiege}
gewonnen. Das ist eine Sieges Wahrscheinlichkeit von
{round((meisteSiege/spielAnzahl)*100, 2)}%")
        print(f"Die Wahrscheinlichkeitsverteilung ist:
{wahrscheinlichkeiten}")

    def main(self, file):
        self._readInput(file)
        self._berechneWahrscheinlichkeiten(20)



test = Aufgabe()
test.main("wuerfel4.txt")
```