Junioraufgabe 1: Wundertüte

Team-ID: 00099

Team-Name: Cryptellum

Bearbeiter/-innen dieser Aufgabe:
Selahaddin Inan

3. November 2023

Inhaltsverzeichnis

1 Lösungsidee

Diese Aufgabe ähnelt sehr an das Behälterproblem , nur mit dem Unterschied, dass die „Größe“ der Behälter (in der Aufgabenstellung Wundertüten genannt) nicht angegeben werden. Stattdessen müssen wir die Unterschiede an „Gewichtung“ (also Spiele pro Wundertüte) zwischen den Behältern so minimal wie möglich halten. Hierzu verteilen wir erst einmal die Spiele, die einigermaßen gleich verteilt werden können auf die Wundertüten, danach verteilen wir die übrig gebliebenen Spiele, die bei der ersten Verteilung nicht gleichmäßig verteilt werden konnten.

2 Umsetzung

2.1 Erste Verteilung

Bei der ersten Verteilung benutzen wir den //-Operator (Floor-Division). Somit erfolgt eine möglichst gleichmäßige Verteilung bei den Spielesorten, die mehr Spiele haben, als es Wundertüten gibt, denn wenn es weniger Spiele als Wundertüte bei einer Spielesorte gibt, kann keine gleichmäßige Verteilung stattfinden.

Beispiel:
Anzahl der Spiele in dieser Spielesorte: 4
Anzahl Wundertüten: 3
$4//3 = 1$ (bedeutet, es wird 1 Spiel pro Tüte verteilt, da wir aber 3 Tüten haben, bleibt 1 Spiel noch zum Verteilen übrig, was dann im nächsten Schritt behandelt wird)

Anderes Beispiel:
(wo keine gleichmäßige Verteilung erst einmal stattfinden kann)
Anzahl der Spiele in dieser Spielesorte: 2
Anzahl Wundertüten: 3
$2//3 = 0$ (2 Spiele übrig, da keine verteilt werden konnten)

2.2 Verteilung der übrig gebliebenen Spiele

Nach der Ersten Verteilung bleiben manchmal Spiele übrig, die noch verteilt werden müssen. Um dies herauszufinden, benutzen wir statt // den %-Operator (Modulo). Hierzu führen wir dieselbe Rechnung wie bei der ersten Verteilung durch, nur diesmal mit dem Modulo-Operator. Dann hätten wir beim ersten Beispiel oben: $4\%3 = 1$ (noch ein Spiel übrig zum verteilen) und beim zweiten Beispiel: $2\%3 = 2$ (noch zwei Spiele zum Verteilen übrig). Nachdem wir die Erste Verteilung gemacht und die Spiele herausgefunden haben, die noch verteilt werden müssen, müssen wir nur noch mit den übrig gebliebenen Spielen pro Spielesorte mit einer for-Schleife durchgehen (wobei wir enumerate() benutzen), und dann einmal pro Wundertüte ein Spiel verteilen, bis wir alle übrig gebliebenen Spiele auch verteilt haben, somit haben wir unsere output-Liste, was dann so aussieht:

3 Beispiele

```
wundertuete0
     Spiel  1 2 3
Wundertüte
         1  2 1 1
         2  1 2 0
         3  1 1 1

wundertuete1
     Spiel  1 2 3
Wundertüte
         1  3 1 2
         2  3 1 2
         3  3 1 2
         4  3 1 2
         5  3 1 2
         6  3 1 2

wundertuete2
     Spiel  1 2 3 4
Wundertüte
         1  2 1 0 0
         2  1 1 1 0
         3  1 1 1 0
         4  1 1 1 0
         5  1 1 0 1
         6  1 1 0 1
         7  1 1 0 1
         8  1 1 0 1
         9  1 1 0 1

wundertuete3
     Spiel  1 2 3 4 5
Wundertüte
         1  1 1 0 0 0
         2  1 1 0 0 0
         3  0 1 1 0 0
         4  0 1 1 0 0
         5  0 1 1 0 0
         6  0 1 1 0 0
         7  0 1 1 0 0
         8  0 1 1 0 0
         9  0 1 0 1 0
        10  0 1 0 1 0
        11  0 1 0 0 1
```

wundertuete4

Spiel	1	2	3	4	5	6
Wundertüte						
1	2	6	3	5	31	5
2	2	6	3	5	30	6
3	2	6	3	5	30	6
4	2	6	2	6	30	6
5	1	7	2	6	30	6
6	1	7	2	6	30	5
7	1	7	2	6	30	5
8	1	7	2	6	30	5
9	1	7	2	6	30	5
10	1	7	2	6	30	5
11	1	7	2	6	30	5
12	1	7	2	6	30	5
13	1	7	2	6	30	5
14	1	7	2	6	30	5
15	1	7	2	6	30	5
16	1	7	2	5	31	5
17	1	6	3	5	31	5

wundertuete5

Spiel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Wundertüte																							
1	1	0	2	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1
2	1	0	2	1	2	0	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
3	1	0	2	1	2	0	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
4	1	0	2	1	2	0	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
5	1	0	2	1	2	0	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
6	1	0	2	1	2	0	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
7	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
8	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
9	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
10	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	2	1
11	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
12	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
13	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
14	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
15	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
16	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
17	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
18	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
19	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
20	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
21	1	0	2	1	1	1	2	0	1	1	1	1	0	1	2	1	1	1	1	2	0	1	2
22	1	0	2	1	1	1	2	0	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
23	1	0	2	1	1	1	2	0	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
24	1	0	2	1	1	1	2	0	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
25	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
26	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
27	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
28	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
29	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
30	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
31	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
32	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2
33	1	0	2	1	1	1	1	1	1	0	2	1	0	1	1	2	1	1	1	2	0	1	2

34	1	0	2	1	1	1	1	1	1	0	2	1	0	0	2	2	1	1	1	2	0	1	2
35	1	0	2	1	1	1	1	1	1	0	2	1	0	0	2	2	1	1	1	2	0	1	1
36	1	0	2	1	1	1	1	1	1	0	2	1	0	0	2	2	1	1	1	2	0	1	1
37	1	0	2	1	1	1	1	1	1	0	2	1	0	0	2	2	1	1	1	2	0	1	1
38	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	2	0	1	1
39	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	2	0	1	1
40	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	2	0	1	1
41	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	2	0	1	1
42	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	2	0	1	1
43	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
44	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
45	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
46	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
47	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
48	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
49	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
50	1	0	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
51	0	1	2	1	1	1	1	1	1	0	2	0	1	0	2	2	1	1	1	1	1	1	1
52	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
53	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
54	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
55	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
56	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
57	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
58	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
59	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
60	0	1	2	1	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
61	0	1	1	2	1	1	1	1	1	0	1	1	1	0	2	2	1	1	1	1	1	1	1
62	0	1	1	2	1	1	1	1	1	0	1	1	0	1	2	2	1	1	1	1	1	1	1
63	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
64	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
65	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
66	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
67	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
68	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
69	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
70	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
71	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	1	1	1
72	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	0	2	1
73	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	0	2	1
74	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	0	2	1
75	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	0	2	1
76	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	0	2	1
77	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	1	1	1	1	0	2	1
78	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	1	1	0	2	1
79	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	1	1	0	2	1
80	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	1	1	0	2	1
81	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	1	1	0	2	1
82	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	0	2	0	2	1
83	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	0	2	0	2	1
84	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	0	2	0	2	1
85	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	0	2	0	2	1
86	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	2	0	2	0	2	1
87	0	1	1	2	1	1	1	1	0	1	1	1	0	1	2	2	0	1	1	2	0	2	1
88	0	1	1	1	2	1	1	1	0	1	1	1	0	1	2	2	0	1	1	2	0	2	1
89	0	1	1	1	2	1	1	1	0	1	1	1	0	1	2	2	0	1	1	2	0	2	1
90	0	1	1	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1
91	0	1	1	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1

92	0	0	2	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1
93	0	0	2	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1
94	0	0	2	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1
95	0	0	2	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1
96	0	0	2	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1
97	0	0	2	1	2	0	2	0	1	1	1	1	0	1	2	2	0	1	1	2	0	2	1

wundertuete6

Spiel	1	2
Wundertüte		
1	1	1
2	1	1

wundertuete7

Spiel	1	2
Wundertüte		
1	10	100
2	10	100
3	10	100
4	10	100
5	10	100
6	10	100
7	10	100
8	10	100
9	10	100
10	10	100

wundertuete8

Spiel	1	2	3
Wundertüte			
1	1	0	1
2	0	1	1
3	0	1	1
4	0	1	1
5	0	0	1

4 Quellcode

```python
for i in range(0, 6):
    with open(f"wundertuete{i}.txt", "r", encoding="utf-8") as datei:
        anzahl_wundertueten = int(datei.readline())

        spiele_sorten_anzahl = int(datei.readline())

        spiele_pro_sorte = [int(datei.readline()) for _ in range(spiele_sorten_anzahl)]

        #verteilen der spiele auf die wundertueten
        output = [[spiel // anzahl_wundertueten for spiel in spiele_pro_sorte] for _ in range(anzahl_wundertueten)]

        #herausfinden, wie viele spiele pro sorte noch uebrig sind
        uebrig_spiele_pro_sorte = [spiel % anzahl_wundertueten for spiel in spiele_pro_sorte]

        #uebrige spiele auf die wundertueten verteilen
        ind = 0
        for index, value in enumerate(uebrig_spiele_pro_sorte):
            for j in range(value):
                output[ind][index] += 1

                ind += 1
                ind %= anzahl_wundertueten

        print("␣".join([str(k) for k in output]), "\n")
```

Quellcode