Im Folgenden wird ohne Rechnereinsatz $\mathbb{E}(321)$ rekursiv berechnet.

$$\begin{aligned}
\mathbb{E}(321) &= p_1(\mathbb{E}(221)+1)+p_2(\mathbb{E}(311)+1)+p_3(\mathbb{E}(320)+1)\\
&= p_1+p_2+p_3+p_1\mathbb{E}(221)+p_2\mathbb{E}(311)+p_3\mathbb{E}(320)\\
&= 1+p_1\mathbb{E}(221)+p_2\mathbb{E}(311)+p_3\mathbb{E}(320)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}(320) &= p_1+p_2+p_3+p_1\mathbb{E}(220)+p_2\mathbb{E}(310)+p_3\mathbb{E}(320)\\
&= \frac{1}{1-p_3}+\frac{p_1}{1-p_3}\mathbb{E}(121)+\frac{p_2}{1-p_3}\mathbb{E}(211)\\
&= \frac{1}{s}+\frac{p_1}{s}\mathbb{E}(220)+\frac{p_2}{s}\mathbb{E}(310);\ s:=\frac{1}{p_1+p_2}
\end{aligned}$$

$$\mathbb{E}(220) = \frac{1}{s}+\frac{p_1}{s}\mathbb{E}(120)+\frac{p_2}{s}\mathbb{E}(210)$$

$$\mathbb{E}(120) = \frac{1}{s}+\frac{p_1}{s}\mathbb{E}(020)+\frac{p_2}{s}\mathbb{E}(110)$$

$$\begin{aligned}
\mathbb{E}(110) &= \frac{1}{s}+\frac{p_1}{s}\mathbb{E}(010)+\frac{p_2}{s}\mathbb{E}(100)\\
&= \frac{19}{5}
\end{aligned}$$

$$= \frac{158}{25}$$

$$\mathbb{E}(210) = \frac{1}{s}+\frac{p_1}{s}\mathbb{E}(110)+\frac{p_2}{s}\mathbb{E}(200) = \frac{127}{25}$$

$$= \frac{878}{125}$$

$$\mathbb{E}(310) = \frac{1}{s}+\frac{p_1}{s}\mathbb{E}(210)+\frac{p_2}{s}\mathbb{E}(300) = \frac{831}{125}$$

$$\mathbb{E}(320) = \frac{5046}{625}$$

Zwischenergebnis 1: $\mathbb{E}(321) = 1 + p_1\mathbb{E}(221) + p_2\mathbb{E}(311) + p_3 \cdot \frac{5046}{625}$

$$\mathbb{E}(311) = 1 + p_1\mathbb{E}(211) + p_2\mathbb{E}(301) + p_3\mathbb{E}(310)$$

$$\mathbb{E}(211) = 1 + p_1\mathbb{E}(111) + p_2\mathbb{E}(201) + p_3\mathbb{E}(210)$$

$$\mathbb{E}(111) = 1 + p_1\mathbb{E}(011) + p_2\mathbb{E}(101) + p_3\mathbb{E}(110)$$

$$\begin{aligned}\mathbb{E}(011) &= \frac{1}{s} + \frac{p_2}{s}\mathbb{E}(001) + \frac{p_3}{s}\mathbb{E}(010);\ s = \frac{1}{p_2 + p_3} \\ &= 7\end{aligned}$$

$$\begin{aligned}\mathbb{E}(101) &= \frac{1}{s} + \frac{p_1}{s}\mathbb{E}(001) + \frac{p_3}{s}\mathbb{E}(100);\ s = \frac{1}{p_1 + p_3} \\ &= \frac{13}{2}\end{aligned}$$

$$= \frac{73}{10}$$

$$\begin{aligned}\mathbb{E}(201) &= \frac{1}{s} + \frac{p_1}{s}\mathbb{E}(101) + \frac{p_3}{s}\mathbb{E}(200); \quad s = \frac{1}{p_1 + p_3} \\ &= \frac{59}{5}\end{aligned}$$

$$\begin{aligned}\mathbb{E}(210) &= \frac{1}{s} + \frac{p_1}{s}\mathbb{E}(110) + \frac{p_2}{s}\mathbb{E}(200); \quad s = \frac{1}{p_1 + p_2} \\ &= \frac{127}{25}\end{aligned}$$

$$= \frac{1591}{200}$$

$$\mathbb{E}(301) = \frac{1}{s} + \frac{p_1}{s}\mathbb{E}(201) + \frac{p_3}{s}\mathbb{E}(300); \quad s = \frac{1}{p_1 + p_3}$$

$$\begin{aligned}\mathbb{E}(201) &= \frac{1}{s} + \frac{p_1}{s}\mathbb{E}(101) + \frac{p_3}{s}\mathbb{E}(200); \quad s = \frac{1}{p_1 + p_3} \\ &= \frac{59}{8}\end{aligned}$$

$$= \frac{273}{32}$$

$$\begin{aligned}\mathbb{E}(310) &= \frac{1}{s} + \frac{p_1}{s}\mathbb{E}(210) + \frac{p_2}{s}\mathbb{E}(300); \quad s = \frac{1}{p_1 + p_2} \\ &= \frac{831}{125}\end{aligned}$$

$$\mathbb{E}(311) = \frac{35717}{4000}$$

Zwischenergebnis 2: $\mathbb{E}(321) = 1 + p_1\mathbb{E}(221) + p_2 \cdot \frac{35717}{4000} + p_3 \cdot \frac{5046}{625}$

$$\begin{aligned}\mathbb{E}(221) &= 1 + p_1\mathbb{E}(121) + p_2\mathbb{E}(211) + p_3\mathbb{E}(220)\end{aligned}$$

$$\begin{aligned}\mathbb{E}(121) &= 1 + p_1\mathbb{E}(021) + p_2\mathbb{E}(111) + p_3\mathbb{E}(120)\end{aligned}$$

$$\begin{aligned}\mathbb{E}(021) &= \frac{1}{s} + \frac{p_2}{s}\mathbb{E}(011) + \frac{p_3}{s}\mathbb{E}(020);\ s = \frac{1}{p_2 + p_3}\\ &= \frac{26}{3}\end{aligned}$$

$$= \frac{441}{50}$$

$$= \frac{27697}{3000}$$

Endergebnis

$$\begin{aligned}\mathbb{E}(321) &= 1 + p_1\mathbb{E}(221) + p_2\mathbb{E}(311) + p_3\mathbb{E}(320)\\ &= 1 + p_1 \cdot \frac{27697}{3000} + p_2 \cdot \frac{35717}{4000} + p_3 \cdot \frac{5046}{625}\\ &= \frac{596\,291}{60\,000} \approx 9.9382\end{aligned}$$

Dieses Beispiel zeigt schon den Kern einer allgemeinen Berechnungsformel auf. Eine formalisierte Darstellung erleichtert die Erstellung eines (fehlerfreien) Programms. Es zeigt sich schon hier, dass ein Rechnereinsatz (sehr) sinnvoll ist.

Eine Anwendung

Der Erwartungswert der Anzahl der Bernoulli-Versuche bis zum n-ten Treffer (mit Trefferwahrscheinlichkeit p) ist n/p. Dies kann auch mithilfe eines Spezialfalls (alle Chips liegen auch genau einem Feld) Schritt für Schritt gezeigt werden. Die ersten zwei Schritte:

$$\begin{aligned}\mathbb{E}(100) &= 1 + p_1\mathbb{E}(000) + p_2\mathbb{E}(100) + p_3\mathbb{E}(100)\\ &= \frac{1}{p_1} + \frac{1}{p_1} \cdot \mathbb{E}(000)\\ &= \frac{1}{p_1}\end{aligned}$$

$$\begin{aligned}\mathbb{E}(200) &= 1 + p_1\mathbb{E}(100) + p_2\mathbb{E}(200) + p_3\mathbb{E}(200)\\ &= \frac{1}{p_1} + \frac{p_1}{p_1} \cdot \mathbb{E}(100)\\ &= \frac{1}{p_1} + \frac{p_1}{p_1}\\ &= \frac{2}{p_1}\end{aligned}$$

Rekursionsformel und Programm für den Ein-Personen-Fall

Mit $V = (v_1, \dots, v_m)$ wird die Spielsituation von n Chips auf m Feldern bezeichnet, die mit den Wahrscheinlichkeiten $p_1, \dots, p_m$ getroffen werden.
Im Fall $v_j \geq 0$ wird durch Entfernen eines Chips von Feld j die entstehende Spielsituation durch

$$V_j := (v_1, \dots, v_j - 1, v_{j+1}, \dots, v_m)$$

dargestellt.
Abbruchbedingung: Das Spiel endet, sobald keine Chips mehr vorhanden sind. Dies kann durch

$$\sum_{i=1}^{m} v_i = 0. \tag{1}$$

dargestellt werden. Dann gilt $\mathbb{E}(V) = 0$.
Weiterhin wird die Summe der positiven Felderwahrscheinlichkeiten $p_j,\ j = 1, \dots, m$, benötigt. Auf den entsprechenden Felder befindet sich mindestens ein Chip. Mithilfe der Indikatorfunktion kann diese Bedingung definiert werden:

$$s := \sum_{j=1}^{m} p_j \, \mathbf{1}(v_j > 0).$$

Damit ergibt sich die folgende (kompakte) Berechnungsvorschrift für $\mathbb{E}(V)$:

$$\mathbb{E}(V) = \begin{cases} 0, & \text{falls (1)}, \\ \dfrac{1}{s} + \displaystyle\sum_{j=1}^{m} \frac{p_j}{s} \cdot \mathbf{1}(v_j > 0) \cdot \mathbb{E}(V_j), & \text{sonst.} \end{cases}$$

Beispiele:

$$\mathbb{E}(3214) = 1 + p_1 \cdot \mathbb{E}(2214) + p_2 \cdot \mathbb{E}(3114) + p_3 \cdot \mathbb{E}(3204) + p_4 \cdot \mathbb{E}(3213)$$

$$\mathbb{E}(3010) = \frac{1}{p_1 + p_3} + \frac{p_1}{p_1 + p_3} \cdot \mathbb{E}(2010) + \frac{p_3}{p_1 + p_3} \cdot \mathbb{E}(3000)$$

Die Rekursionsformel kann mit der Mittelwertsregel nachvollzogen werden:

$$\begin{aligned}
\mathbb{E}(3010) &= 1 + p_1 \cdot \mathbb{E}(2010) + p_2 \cdot \mathbb{E}(3010) + p_3 \cdot \mathbb{E}(3000) + p_4 \cdot \mathbb{E}(3010) \\
\mathbb{E}(3010)(1 - p_2 - p_4) &= 1 + p_1 \cdot \mathbb{E}(2010) + p_3 \cdot \mathbb{E}(3000) \\
&= \frac{1}{1 - p_2 - p_4} + \frac{p_1}{1 - p_2 - p_4}\mathbb{E}(2010) + \frac{p_3}{1 - p_2 - p_4}\mathbb{E}(3000) \\
&= \frac{1}{p_1 + p_3} + \frac{p_1}{p_1 + p_3}\mathbb{E}(2010) + \frac{p_3}{p_1 + p_3}\mathbb{E}(3000)
\end{aligned}$$

Programmdokumentation

Das Python-Skript berechnet exakt den Erwartungswert der Anzahl benötigter Würfe, bis alle Chips abgeräumt sind, bei einem Spiel mit m Feldern und einer vorgegebenen Ein-Personen-Spielstrategie $V = (v_1, \ldots, v_m)$ und vorgegebenen Wahrscheinlichkeiten für die einzelnen Felder.

```python
# Der 1-Personen-Fall
from fractions import Fraction
from functools import lru_cache

# 1) Wahrscheinlichkeiten p_j fuer die m Felder (als Fraction fuer exakte Brueche)
p = (Fraction(1, 2), Fraction(1, 3), Fraction(1, 6))
m = len(p)

# 2) Abbruchpruefung: Spiel endet, wenn keine Chips mehr da sind
def is_terminal(V):
    # V ist ein Tupel (v_1, ..., v_m)
    return sum(V) == 0

# 3) Zustandsupdate:
#    Entferne bei Feld j einen Chip, falls v_j > 0
def next_state(C, j):
    # C ist das aktuelle Tupel, j der Index des Felds
    return C[:j] + (C[j] - 1 if C[j] > 0 else 0,) + C[j+1:]

# 4) Rekursive Berechnung mit Memoization
@lru_cache(maxsize=None)
def E(V):
    """
    Berechnet den Erwartungswert E(V) ab Zustand V.
    V      : Tupel der Laenge m mit den aktuellen Chipzahlen.
    return: Fraction (exakte Darstellung) des Erwartungswerts.
    """
    # 4.1 Terminalfall
    if is_terminal(V):
        return Fraction(0)
    # 4.2 Normierung: Summe der Wahrscheinlichkeiten bei nicht-leeren Feldern
    s = sum(p[j] for j in range(m) if V[j] > 0)
    # 4.3 Initialer Wert: 1 Runde (aktiver Wurf)
    total = Fraction(1, s)
    # 4.4 Rekursiver Beitrag
    for j in range(m):
        if V[j] > 0:
            Vj = next_state(V, j)
            total += (p[j] / s) * E(Vj)
    return total

# 5) Beispielaufruf und Cache-Info
strategy = (3, 2, 1)
result = E(strategy)
print(f"E({strategy}) = {result}  (exakt)")
print(f"E({strategy}) = {float(result):.5f}  (gerundet)")
print("Cache-Statistik:", E.cache_info())
```

Programmdokumentation für den Ein-Personen-Fall

Module `fractions`, `functools`

`Fraction` ermöglicht exakte Bruchrechnung ohne Rundungsfehler
`lru_cache` dient der Memoization; einmal berechnete Werte von $\mathbb{E}(V)$ werden zwischengespeichert.

Wahrscheinlichkeiten `p`

Das Tupel `p` enthält die Trefferwahrscheinlichkeiten p_j für jedes Feld $j = 0, \ldots, m-1$.

`is_terminal(V)`

Liegen keine Chips mehr (Summe aller Komponenten =0), endet das Spiel. Dies entspricht dem Abbruchfall $E(V) = 0$.

`next_state(C,j)`

Berechnet das Tupel-Update nach einem Wurf auf Feld j in einer Funktion: Ein Chip wird abgezogen, falls vorhanden.

`E(V)`

Die Kernfunktion:

1. Prüft den Terminalfall.
2. Berechnet die Normierung $s := \sum_{j=1}^{m} p_j\, \mathbf{1}(v_j > 0)$..
3. Initialisiert den Erwartungswert mit $1/s$ (für den aktuellen Wurf).
4. Addiert für jedes nicht-leere Feld j den rekursiven Anteil $\frac{p_j}{s}\, E(V_j)$.

Memoization `@lru_cache`

Verhindert wiederholte Neuberechnung desselben Zustands V.
Mit `E.cache_info()` kann man Hits (schon einmal berechnete Werte) und Misses (neu berechnete Werte) auslesen.

1 Der Zwei-Personen-Fall

Die vorangegangen Überlegungen lassen sich konzeptionell auf den Fall übertragen, dass mehrere Personen gegeneinander spielen. Der Einfachheit halber beschränken wir uns auf den Fall, dass zwei Personen A und B mit den verschiedenen Setzstrategien $S = (s_1, \ldots, s_m)$ und $T = (t_1, \ldots, t_m)$ gegeneinander antreten und das Ergebnis jedes Wurfes für beide maßgeblich ist. Wer zuerst alle Chips abgeräumt hat, gewinnt. Entsteht eine Situation, in der A und B auf jedem Feld gleich viele Chips liegen haben, so ist das Spiel (mit unentschiedenem Ausgang) beendet.
Jede Spielsituation ist jetzt durch ein Paar (V, W) von m-Tupeln $V = (v_1, \ldots, v_m)$ und $W = (w_1, \ldots, w_m)$ gegeben. Hierbei bezeichnen v_j bzw. w_j die Anzahl der Chips, die A bzw. B noch auf Feld j stehen haben ($j = 1, \ldots, m$). Zu guter Letzt bezeichne (V, W) den Erwartungswert *der Anzahl noch nötiger Würfe bis zum Spielende in der Spielsituation* (V, W). Bei identischer Spielsituation ist das Spiel mit einem Unentschieden beendet, sodass kein weiterer Wurf erfolgt. In diesem Fall gilt also $(V, W) = 0$. Es erfolgt auch kein weiterer Wurf, wenn entweder A oder B keinen Chip mehr im Spiel haben. Diese beiden Abbruchbedingungen können symbolisch folgendermaßen dargestellt werden:

$$V = W \quad \vee \quad \sum_i v_i = 0 \quad \vee \quad \sum_i w_i = 0. \tag{2}$$

Weiter definieren wir die Normierung

$$s := \sum_{j=1}^{m} p_j\, \mathbf{1}(v_j + w_j > 0),$$

was die Gesamtwahrscheinlichkeit darstellt, dass in einem Wurf mindestens einer der beiden Spieler einen Chip verliert. Damit ergibt sich die folgende Berechnungsvorschrift für $\mathbb{E}(V, W)$:

$$\mathbb{E}(V,W) = \begin{cases} 0, & \text{falls (2)}, \\ \dfrac{1}{s} + \displaystyle\sum_{j=1}^{m} \frac{p_j}{s} \cdot \mathbf{1}(v_j + w_j > 0) \cdot \mathbb{E}(V_j, W_j), & \text{sonst.} \end{cases}$$

Die neuen Spielsituationen V_j und W_j sind so wie beim 1-Personen-Spiel definiert.

Programm für den 2-Personen-Fall

```python
from fractions import Fraction
from functools import lru_cache

p = (Fraction(1, 2), Fraction(1, 3), Fraction(1, 6))
m = len(p)

def is_terminal(V, W):
    return V == W or sum(V) == 0 or sum(W) == 0

def next_state(C, j):
    return C[:j] + (C[j] - 1 if C[j] > 0 else 0,) + C[j+1:]

@lru_cache(maxsize=None)

def E(V, W):
    if is_terminal(V, W): return Fraction(0)
    sum_E = Fraction(0)
    for j in range(m):
        pj = p[j]
        if V[j] + W[j] > 0:
            Vj = next_state(V, j)
            Wj = next_state(W, j)
            sum_E += pj * E(Vj, Wj)
    r = sum(p[j] for j in range(m) if V[j] + W[j] > 0)
    return (Fraction(1) + sum_E) / r

A_strategy = (3, 2, 1)
B_strategy = (4, 2, 0)
val = E(A_strategy, B_strategy)
print(f"E({A_strategy}, {B_strategy}) = {val}")
```

Beispiel

$$\begin{aligned} \mathbb{E}(100, 010) &= 1 + p_1\mathbb{E}(000, 010) + p_2\mathbb{E}(100, 000) + p_3\mathbb{E}(100, 010) \\ &= \frac{1}{p_1 + p_2} + \frac{p_1}{p_1 + p_2}\mathbb{E}(000, 010) + \frac{p_2}{p_1 + p_2}\mathbb{E}(100, 000) \\ &= \frac{6}{5}; \quad p_1 = \frac{1}{2},\ p_2 = \frac{1}{3} \end{aligned}$$

Ergebnis: $\mathbb{E}(100, 010) < \min(\mathbb{E}(V); \mathbb{E}(W))$.