LU07.L08 - Vorgegebene Massnahmen umsetzen

Ergebnisse

Gemessen mit `min(timeit.repeat(…, number=1, repeat=5))` auf einem Notebook, Python 3.12. **Ihre absoluten Zahlen werden abweichen - die Grössenordnung der Faktoren nicht.**

Massnahme	vorher	nachher	Faktor	Begründung
1 list → set	0.107 s	0.000068 s	~1500	`in` auf einer Liste prüft im Schnitt die halbe Liste (O(n)), das Set berechnet einen Hash (O(1))
2 Schleife → dict-Index	0.0585 s	0.00028 s	~200	Statt 2000 × 2000 Vergleichen (O(n²)) wird der Index einmal gebaut und dann direkt zugegriffen (O(n))
3 Memoization	0.0899 s	0.0000003 s	sehr gross	Ohne Cache berechnet der Aufrufbaum dieselben Teilprobleme tausendfach, mit Cache jedes genau einmal
4 Arbeit aus der Schleife	0.0877 s	0.00025 s	~350	`sorted` lief 2000-mal mit identischer Eingabe und identischem Ergebnis

Code

```
# 1
gesperrt = {f"user{i}" for i in range(10000)}     # geschweifte statt eckige
Klammern


# 2
def zuordnen(bestellungen, kunden):
    index = {k["id"]: k for k in kunden}
    return [(index[b["kunde_id"]]["name"], b["betrag"]) for b in
bestellungen]


# 3
from functools import cache

@cache
def fibonacci(n):
    return n if n < 2 else fibonacci(n - 1) + fibonacci(n - 2)


# 4
def top_drei_pro_kunde(kunden, produkte):
    top = [p["name"] for p in sorted(produkte, key=lambda p:
```

```
p["preis"])[:3]]
    return {k["name"]: list(top) for k in kunden}
```

Fallen in dieser Aufgabe

- **Massnahme 2:** Die Vorher-Version behält die Reihenfolge der Bestellungen. Die Index-Version auch - aber nur, weil über `bestellungen` iteriert wird. Wer über kunden iteriert, bekommt eine andere Reihenfolge und damit ein anderes Ergebnis.
- **Massnahme 3:** `@cache` funktioniert nur bei pure functions mit hashbaren Argumenten. Eine Liste als Argument wirft `TypeError: unhashable type`.
- **Massnahme 4:** `list(top)` statt `top` - sonst teilen sich alle 2000 Kunden **dieselbe** Liste. Solange niemand sie verändert, fällt das nicht auf; sobald doch, ändern sich alle Einträge gleichzeitig. Das ist der Referenz-Effekt aus LU02d.
- **Ohne `assert` ist keine Messung gültig.** Eine schnellere Funktion mit anderem Ergebnis ist keine Optimierung, sondern ein Bug.

Zum Weiterdenken

Massnahme 1 kostet Speicher: Das Set legt zusätzlich zur Liste eine Hash-Tabelle an. Bei 10'000 Einträgen ist das irrelevant, bei 50 Millionen nicht. Diese Abwägung - Zeit gegen Speicher - ist genau das, was auf Niveau **D1A** begründet werden muss.

M323-LU07, M323-D1I