LU07.L09 - Datenstruktur begründet auswählen

Die drei Fassungen

```python
def mit_liste(ausleihen, gesperrte):
    out = []
    for k in ausleihen:
        for konto, grund in gesperrte:
            if konto == k:
                out.append((k, grund))
                break
    return out


def mit_set(ausleihen, gesperrte):
    konten = {k for k, _ in gesperrte}
    gruende = dict(gesperrte)
    return [(k, gruende[k]) for k in ausleihen if k in konten]


def mit_dict(ausleihen, gesperrte):
    index = dict(gesperrte)
    return [(k, index[k]) for k in ausleihen if k in index]


assert mit_liste(ausleihen, gesperrte) == mit_set(ausleihen, gesperrte) ==
mit_dict(ausleihen, gesperrte)
```

Messung

5000 Ausleihen, wachsende Sperrliste (Python 3.12, Werte eines Testlaufs):

Gesperrte Konten	Liste	Set + Dict	Dict	Indexaufbau
100	0.0057 s	0.00026 s	0.00028 s	0.000004 s
2 000	0.1159 s	0.00044 s	0.00042 s	0.000076 s
20 000	1.1453 s	0.00282 s	0.00142 s	0.001014 s

Die Liste wird bei zehnfacher Datenmenge rund zehnmal langsamer (O(n) pro Abfrage, O(n·m) gesamt). Dict und Set bleiben nahezu konstant - der Anstieg dort stammt fast vollständig aus dem **Aufbau** des Index, nicht aus den Abfragen.

Antworten

Ab wann lohnt sich der Index? Schon bei 100 Einträgen ist die Dict-Fassung 20-mal schneller, und der Aufbau kostet vier Mikrosekunden. Die Faustregel lautet deshalb: Sobald in einer Schleife gesucht wird, lohnt sich der Index praktisch immer. Nicht lohnen tut er sich, wenn die Struktur für **eine einzige** Abfrage aufgebaut wird - dann kostet der Aufbau mehr, als die Suche spart.

Was geht verloren?

- Set und Dict verlieren die **Reihenfolge** der Originaldaten (Dict behält die Einfügereihenfolge, aber nicht die der Ausleihen).
- **Duplikate** verschwinden. Steht ein Konto zweimal mit verschiedenen Gründen in der Liste, gewinnt beim Dict der letzte Eintrag - stillschweigend.
- Die Schlüssel müssen **hashbar** sein.
- Es wird zusätzlicher **Speicher** belegt.

Wann ist die Liste richtig?

- Wenn nur **einmal** gesucht wird.
- Wenn die Sperrliste sehr klein ist und die Reihenfolge fachlich zählt (etwa «zuerst eingetragene Sperre zuerst anzeigen»).
- Wenn Duplikate erhalten bleiben müssen, weil jedes eine eigene Sperre mit eigenem Datum ist.

Warum Dict statt Set? Beide prüfen die Zugehörigkeit in O(1). Das Set braucht aber eine **zweite** Struktur für den Grund, also zwei Durchläufe über dieselben Daten und zwei Lookups pro Treffer. Das Dict liefert Prüfung und Grund in einer Struktur - sichtbar in der Messung bei 20'000 Einträgen (0.00142 s gegenüber 0.00282 s).

Empfehlung: `dict`, sobald mehr als eine Abfrage stattfindet. Das `set` ist die richtige Wahl, wenn Sie nur die Frage «ist enthalten?» beantworten und keinen Wert dazu brauchen.

Formulierungsbeispiel für das Portfolio (D1A)

> Für die Sperrprüfung habe ich die Liste durch ein Dict ersetzt. Gemessen mit 5000 Ausleihen gegen 20'000 gesperrte Konten: 1.145 s vorher, 0.0014 s nachher, bei identischem Ergebnis (per `assert` geprüft). Grund: Die Listenfassung durchsucht pro Ausleihe im Schnitt die halbe Sperrliste (O(n)), das Dict greift über den Hash direkt zu (O(1)). Der Preis ist zusätzlicher Speicher und der Verlust von Duplikaten - da eine Kontonummer nur einmal gesperrt sein kann, ist das hier unproblematisch. Wäre die Reihenfolge der Sperreinträge fachlich relevant, hätte ich stattdessen einen zusätzlichen Index neben der Liste gehalten.

M323-LU07, M323-D1A