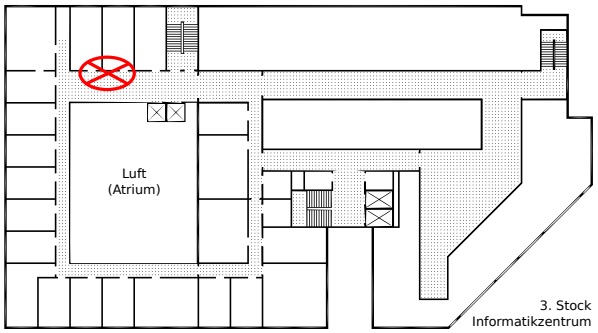
Algorithmen und Datenstrukturen 2 **Sommer 2025**

Dr. Arne Schmidt
Ramin Kosfeld
Chek-Manh Loi

Abgabe: 23.05.2025
Rückgabe: ab 26.05.2025

Hausaufgabenblatt 2

Abgabe der Lösungen bis zum 23.05.2025 um 13:30 Uhr im Hausaufgabenschrank bei Raum IZ 337 (siehe Skizze rechts). Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet. **Auf deine Abgabe unbedingt Namen, Matrikelnummer und deine Gruppennummer schreiben! Die Blätter bitte zusammenheften!**



Hausaufgabe 1 (Subset Sum): **(3+2 Punkte)**

a) Wende das dynamische Programm für Subset Sum aus der Vorlesung auf folgende Instanz an:

i	1	2	3	4	5
z_i	8	3	4	3	2

und $Z = 13$.

Fülle dazu die folgende Tabelle aus, wobei der Eintrag in Zeile i und Spalte x dem Wert $\mathcal{S}(x, i)$ entspricht.

i \ x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0														
1														
2														
3														
4														
5														

b) Wie kann das dynamische Programm für Subset Sum verwendet werden, um Partition zu lösen?

Hausaufgabe 2 (Wechselgeld): **(2+3+4 Punkte)**

Wir möchten einen Verkaufsautomaten programmieren, der das Wechselgeld mit möglichst wenig Münzen zurückgibt. Formal:

Gegeben: Eine Zahl $W \in \mathbb{N}$ und Münzwerte $1 = M_1 < M_2 < \cdots < M_m$.

Gesucht: Nicht-negative, ganze Zahlen $x_1, \ldots, x_m$, sodass

$$\sum_{i=1}^{m} x_i M_i = W \text{ und}$$

$$\sum_{i=1}^{m} x_i \text{ minimal.}$$

Algorithmus 1 zeigt einen Greedy-Algorithmus, der möglichst hochwertige Münzen priorisiert.

```
function CHANGE(W,M_1, ..., M_m)
    for i = m down to 1 do
        x_i := ⌊W / M_i⌋                ▷ Nimm Münze i so oft wie möglich.
        W := W − x_i · M_i
    return x_1, ..., x_m
```

Algorithmus 1: Ein Greedy-Algorithmus, der möglichst wenig Münzen Wechselgeld zurückgeben soll.

a) Algorithmus 1 ist tatsächlich optimal für manche Währungssysteme, zum Beispiel den Euro. Das Euro-Währungssystem besitzt die folgenden Münztypen: 1-, 2-, 5-, 10-, 20-, 50-, 100- und 200-Cent-Münzen. Gibt Algorithmus 1 weiterhin stets die minimale Anzahl an Münzen zurück, wenn zusätzlich 25-Cent-Münzen (wie die vom US-Dollar bekannten *quarters*) eingeführt werden? Begründe Deine Antwort.

b) Sei nun $\text{OPT}(i, x)$ der minimale Wert, wie viele der ersten i Münzen benötigt werden, um den Wert x zu erreichen. Gib eine Rekursionsgleichung an, die $\text{OPT}(i, x)$ für ein Währungssystem mit Münzen im Wert von $1 = M_1 < M_2 < \cdots < M_m$ bestimmt.

c) Zeige: Wenn M_{i+1} für alle $1 \leq i < m$ ohne Rest durch M_i teilbar ist, dann ist Algorithmus 1 optimal.
Hinweis: Angenommen, Z sei der zu erreichende Wert und $Z > M_j$ für ein $2 \leq j \leq m$. Wie oft kann man jede Münze M_i mit $i < j$ maximal verwenden? Außerdem gilt: $\sum_{i=1}^{n} a_{i+1} - a_i = a_{n+1} - a_1$ für alle Folgen a_i (Teleskopsumme).

Hausaufgabe 3 (Minimal Gewichtetes Vertex Cover): (3+3 Punkte)
Eine Menge $C \subseteq V$ von Knoten aus dem Graph $G = (V, E)$ ist genau dann ein *Vertex Cover* von G, wenn für jede Kante mindestens ein Knoten in C liegt, d.h., $\forall \{u, v\} \in E : |\{u, v\} \cap C| \geq 1$.

Ein Graph G ist gewichtet, wenn es eine Kostenfunktion $c : V \mapsto \mathbb{N}$ gibt, die jedem Knoten v ein Gewicht $c(v)$ zuordnet.

a) Zeige: Für jeden gewichteten Baum $T = (V, E)$ existiert ein Vertex Cover M mit

$$\sum_{v \in M} c(v) \leq \sum_{v \in V} c(v)/2,$$

also mit maximal der Hälfte des Gesamtgewichts. (Hinweis: Berechne die Entfernungen aller Knoten zu einem Startknoten.)

b) Wir wollen für einen Baum $T = (V, E)$ das Gewicht eines *minimalen gewichteten Vertex Cover* M berechnen, also das kleinste mögliche Gesamtgewicht $\sum_{v \in M} c(v)$, was ein Vertex Cover im Baum T haben kann. Wie in der Vorlesung nehmen wir dafür an, dass der Baum einen Wurzelknoten hat; die Menge $N(v)$ enthält dann jeweils alle Kinder von v.

Stelle die Rekursionsgleichung für $M(v)$ auf, die das Gewicht des minimalen gewichteten Vertex Cover für den Teilbaum mit Wurzelknoten v berechnet.