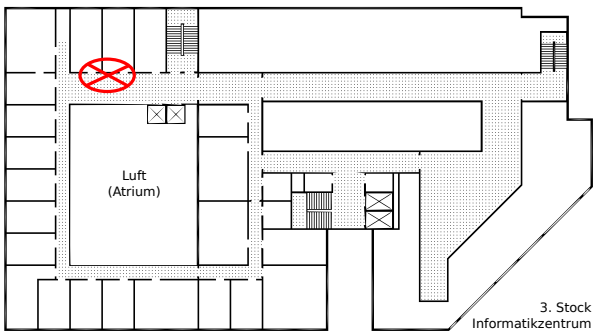
Algorithmen und Datenstrukturen 2 **SoSe 2022**
Prof. Dr. Sándor Fekete **Abgabe:** 01.06.22
Matthias Konitzny **Rückgabe:** ab 08.06.22

Hausaufgabenblatt 2

Abgabe der Lösungen bis zum 01.06.22 um 12:00 Uhr im Hausaufgabenschrank bei Raum IZ 337 (siehe Skizze rechts). Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet. **Bitte die Blätter zusammenheften und vorne deutlich mit beiden Namen, Matrikel- Übungs- und Gruppennummer versehen!**



Hausaufgabe 1 (Wechselgeld): **(3+4+5 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 Wechselgeld zurückgeben soll.

a) Algorithmus 1 ist tatsächlich optimal für übliche Währungssysteme (Euro, US-Dollar, etc.). Das Euro-Währungssystem besitzt dabei die folgenden Münztypen: 1-, 2-, 5-, 10-, 20-, 50-, 100- und 200-Cent-Münzen. Gibt Algorithmus 1 die minimale Anzahl an Münzen zurück, wenn 4-Cent-Münzen 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) In dieser Teilaufgabe kann zwischen einer theoretischen und einer praktischen Aufgabe gewählt werden. Es muss lediglich **eine** der Aufgaben (c.1 oder c.2) bearbeitet werden, um die vollen Punkte zu erreichen.

(Hinweis: Möchtest du dennoch beide Aufgaben bearbeiten, werten wir die Aufgabe mit der höheren Punktzahl.)

c.1 Theoretische Aufgabe **(5 Punkte)**

Zeige: Wenn $\frac{M_{i+1}}{M_i} \in \mathbb{N}$ für alle $1 \leq i < m$ gilt, dann ist der Algorithmus 1 optimal.
(Hinweis:
(1) 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?
(2) Für eine Folge $a_1, \ldots, a_n$ gilt: $\sum_{i=1}^{n-1} a_{i+1} - a_i = a_n - a_1$)

c.2 Praktische Aufgabe **(2+3 Punkte)**

In dieser Aufgabe sollen die Algorithmen zur Lösung des Wechselgeld-Problems in Python implementiert werden. Falls Python nicht installiert ist, kann ein Browser-basierter Interpreter unter `https://brython.info/tests/editor.html` genutzt werden.

Nutze für die Entwicklung das Template unter `https://ibr.cs.tu-bs.de/courses/ss22/aud2/aufgaben/blatt2_template.py`. Dieses enthält auch eine Reihe von Beispielinstanzen mit denen die Implementierung überprüft werden kann.

Abgabe: Die Lösung dieser Aufgabe muss in Form einer Python-Datei (.py) an die/den jeweilge*n Übungsleiter*in gesendet werden. In der Mail muss der Name, die Matrikel- Übungs- und Gruppennummer angegeben werden. Die jeweiligen Mailadressen gibt es unter `https://aud2.ibr.cs.tu-bs.de/index.php/organisation/`.

(i) Implementiere Algorithmus 1. Die Implementierung sollte maximal 10 Zeilen lang sein und muss eine Liste mit den Werten $x_1, \ldots, x_m$ zurückgeben.

(ii) Implementiere deinen Dynamic Programming Ansatz aus Aufgabenteil b). Eine naive Implementierung ist hierbei ausreichend. Modifiziere deinen Algorithmus so, dass zusätzlich zur Gesamtmenge der gewählten Münzen auch eine Liste mit der Anzahl der einzelnen Münzwerte $x_1, \ldots, x_m$ zurückgeben wird. Der Algorithmus sollte nicht mehr als 30 Zeilen umfassen.

Hausaufgabe 2 (Subset Sum): **(6+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	7	4	1	9	3

und $Z = 15$.

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	14	15
0																
1																
2																
3																
4																
5																

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