Algorithmen und Datenstrukturen II **SoSe 2020**
Prof. Dr. Sándor Fekete **Abgabe:** 04.06.20
Arne Schmidt **Rückgabe:** ab 10.06.20

Übungsblatt 2

Abgabe der Lösungen muss bis zum 04.06.20 um 23:59 Uhr erfolgen. Lösungen müssen per Mail mit einer pdf-Datei (Name der Datei „`blatt_[nr]_[grp]_[team].pdf`“) an den jeweiligen Tutor geschickt werden. Die Email-Adressen sind unter `https://www.ibr.cs.tu-bs.de/alg/index.html` zu finden.

Hausaufgabe 1 (Longest Common Subsequence): (7 Punkte)
Betrachte das Problem Longest Common Subsequence aus der Übung #2:

Gegeben: Alphabet Z und zwei Sequenzen $X := x_1 \dots x_n$ und $Y = y_1 \dots y_m$, wobei für alle $i \leq n$ und $j \leq m$ gilt $x_i, y_j \in Z$.

Gesucht: Eine längstmögliche Sequenz T, die eine Teilsequenz von X und Y ist.

Eine Teilsequenz eines Wortes entsteht durch Weglassen von Buchstaben. Beispielsweise ist ACTG eine Teilsequenz von ACCTATATGTT.

Sei $\text{LCS}(X^i, Y^j)$ die größtmögliche Länge einer Teilsequenz von X^i und Y^j, wobei X^i (bzw. Y^j) dem Teilwort $x_1 \dots x_i$ (bzw. $y_1 \dots y_j$) entspricht.

Die Rekursionsgleichung, um $\text{LCS}(X^i, Y^j)$ zu bestimmen, lautet wie folgt.

$$\text{LCS}(X^i, Y^j) = \begin{cases} 0, & \text{falls } i = 0 \text{ oder } j = 0 \\ \text{LCS}(X^{i-1}, Y^{j-1}) + 1, & \text{falls } x_i = y_j \\ \max(\text{LCS}(X^{i-1}, Y^j), \text{LCS}(X^i, Y^{j-1})), & \text{sonst} \end{cases}$$

Bestimme für $X = \text{CGGGTAC}$ und $Y = \text{CCAGATA}$ die Länge einer längsten gemeinsamen Teilsequenz, indem du die folgende Tabelle mit Hilfe der Rekursionsgleichung ausfüllst. Dabei entspricht die Zelle in Spalte i und Zeile j dem Eintrag $\text{LCS}(X^i, Y^j)$.

Y \ X	∅	C	G	G	G	T	A	C
∅								
C								
C								
A								
G								
A								
T								
A								

Hausaufgabe 2 (Dynamische Programmierung für Knapsack): (6 Punkte)
Wende das dynamische Programm für Maximum Knapsack aus der Vorlesung auf folgende Instanz an:

i	1	2	3	4	5	6
z_i	3	2	1	4	2	3
p_i	1	4	2	5	2	4

und $Z = 9$.

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

i \ x	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										

Hausaufgabe 3 (Segelflug): (5 Punkte)
Bei einem Segelflugturnier sollen die Teilnehmer aus einer gegebenen Reihenfolge von Checkpoints k Checkpoints auswählen und diese abfliegen. Dabei starten alle bei dem gleichen Checkpoint. Gewonnen hat derjenige Teilnehmer, der am Ende die längste Strecke zurückgelegt hat.

Formal lässt sich dieses Problem folgendermaßen beschreiben:

Gegeben: n Punkte $p_1, \ldots, p_n \in \mathbb{R}^3$, eine Zahl $k \in \mathbb{N}$ mit $k \leq n$ und eine Distanzfunktion $d : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$.

Gesucht: k Punkte $p_{f(1)}, \ldots, p_{f(k)}$ mit der injektiven Funktion $f : \{1, \ldots, k\} \to \{1, \ldots, n\}$, sodass folgendes gilt:

- $f(1) = 1$
- Für alle $1 \leq i, j \leq k$ gilt: Wenn $i < j$, dann ist $f(i) < f(j)$.
- $\sum_{i=1}^{k-1} d(p_{f(i)}, p_{f(i+1)})$ maximal

Sei nun $L(\ell, j)$ die Länge eines längsten Weges von p_1 bis p_j über ℓ Checkpoints (inklusive p_1). Sollte für ein ℓ und j kein Weg existieren, dann ist $L(\ell, j) = -\infty$.

Stelle eine Rekursionsgleichung auf, um $L(\ell, j)$ zu bestimmen.

Hausaufgabe 4 (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 erste 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 $\frac{M_{i+1}}{M_i} \in \mathbb{N}$ 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: $\sum_{i=1}^{n-1} a_{i+1} - a_i = a_n - a_1$)