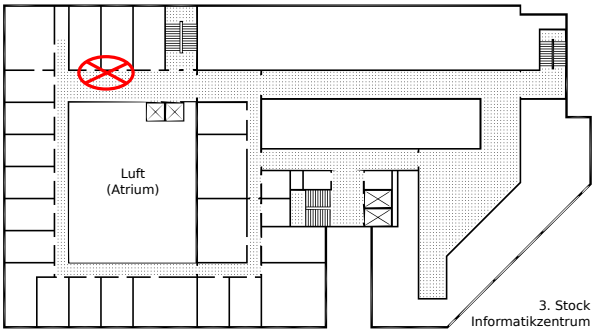
Abteilung Algorithmik **Sommer 2013**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Stephan Friedrichs

Algorithmen und Datenstrukturen II
Übung 1 vom 24. 04. 2013

Abgabe der Lösungen bis zum Mittwoch, den 08. 05. 2013 um 14:00 im Hausaufgabenrückgabeschrank.

Bitte die Blätter vorne deutlich mit eigenem Namen sowie Matrikel- und Gruppennummer versehen!



Aufgabe 1 (Knapsack): Algorithmus 1.4 aus der Vorlesung löst Problem 1.3 (Fractional Knapsack) optimal. Löst Algorithmus 1.4, wenn man Schritt 3 weglässt, auch Problem 1.2' (Binary Maximum Knapsack) optimal? Begründe Deine Antwort. **(3 Punkte)**

Aufgabe 2 (Subset Sum): In dieser Aufgabe geht es um Algorithmus 1.11 (Dynamic Programming für Subset Sum) aus der Vorlesung. Der bestimmt in seiner gegebenen Form für den Input $z_1, \ldots, z_n$ mit $Z = \sum_{i=1}^{n} z_i$ für alle $x \in \{0, \ldots, Z\}$, ob sie als Summe aus $z_1, \ldots, z_n$ erzeugbar sind.

a) Modifiziere Algorithmus 1.11 dahingehend, dass er nicht nur für jede Zahl angibt *ob* sie erreichbar ist, sondern für die erreichbaren Zahlen zusätzlich beschreibt, die Summe *welcher* Zahlen sie sind.

b) Wende Deinen Algorithmus auf $2, 3, 6$ an. Betrachte die Eingabezahlen dabei in der gegebenen Reihenfolge. Markiere nach Anwendung Deines Algorithmus' deutlich, woran man erkennt, ob bzw. wie die 5 erreichbar ist.

(3+2 Punkte)

Aufgabe 3 (Subset Sum): Alternativ zu Algorithmus 1.11 (siehe oben) kann das Subset-Sum-Problem der Inputzahlen $z_1, \ldots, z_n$ gelöst werden, indem man alle Kombinationen von „z_i wählen" bzw. „z_i nicht wählen" ausprobiert (der Brute-Force-Ansatz).

a) Wie viele Kombinationen probiert der Brute-Force-Ansatz schlimmstenfalls durch? Was ist seine Laufzeitkomplexität?

b) Konstruiere eine Eingabe $z_1, \ldots, z_n$, bei der Algorithmus 1.11 effizienter ist als Brute Force.

c) Konstruiere eine Eingabe $z_1, \ldots, z_n$, bei der Brute Force effizienter als Algorithmus 1.11 ist.

(1+2+2 Punkte)

Aufgabe 4 (Longest Common Subsequence): Es seien die Sequenzen $X = x_1 \cdots x_n$ und $Y = y_1 \cdots y_m$ gegeben. Eine (möglicherweise leere) *Teilsequenz* entsteht durch Weglassen von Elementen, zum Beispiel ist ACTG eine Teilsequenz von ACCTATATGTT. Ist Z eine Teilsequenz von X und auch von Y, heißt Z *gemeinsame Teilsequenz* von X und Y. Eine gemeinsame Teilsequenz von X und Y maximaler Länge heißt *Longest Common Subsequence* (LCS) von X und Y. Sei $Z = z_1 \cdots z_k$ eine LCS von X und Y.

a) Ist LCS von X und Y eindeutig? Begründe Deine Antwort.

b) Zeige: Falls $x_n = y_m$, ist $z_1 \cdots z_{k-1}$ eine LCS von $x_1 \cdots x_{n-1}$ und $y_1 \cdots y_{m-1}$.

c) Zeige: Falls $x_n \neq y_m$ und $z_k \neq x_n$, ist Z eine LCS von $x_1 \cdots x_{n-1}$ und Y.

(1+3+3 Punkte)