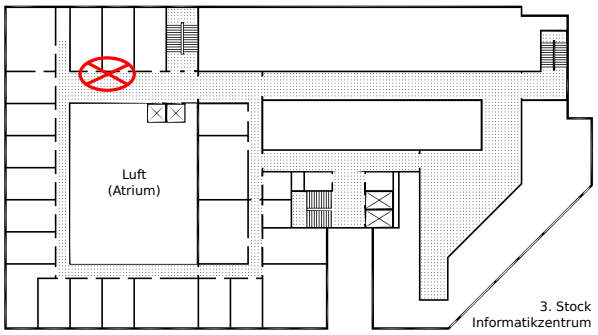
Abteilung Algorithmik **Sommer 2013**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Stephan Friedrichs

Algorithmen und Datenstrukturen II
Übung 2 vom 08. 05. 2013

Abgabe der Lösungen bis zum Mittwoch, den 29. 05. 2013 um 14:00 im Hausaufgabenrückgabeschrank.

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



Auf diesem Blatt gibt es 45 Punkte, erreichbar (und abzugeben) sind aber lediglich (Teil-) Aufgaben im Gesamtwert von **maximal 30** Punkten. Mehr wird nicht gewertet!

Aufgabe 1 (Dynamic Programming für Knapsack): Betrachte Algorithmus 1.15 (DP für Knapsack) aus dem Skript.

a) Wende Algorithmus 1.15 auf folgende Knapsackinstanz an:

i	1	2	3	4
Gewicht z_i	5	4	6	3
Wert p_i	10	40	30	50

Die Rucksackkapazität ist $Z = 10$.

b) Beweise Satz 1.16 („Algorithmus 1.15 berechnet einen besten Lösungswert für das Rucksackproblem in Zeit $O(nZ)$") aus dem Skript.

c) Beweise oder widerlege: In Algorithmus 1.15 können die Schleifen 2.1 und 2.2 zusammengefasst werden, wenn für $x < 0$ implizit $P(x, i) = -\infty$ angenommen wird.

(5+5+5 Punkte)

Aufgabe 2 (Wechselgeld): Betrachte ein Währungssystem mit Münzen im Wert von $S_1, S_2, \ldots, S_k$, wobei $S_1 = 1$ und $S_i < S_{i+1}$ gilt.

a) Entwirf einen Dynamic-Programming-Algorithmus, der bestimmt wie vielen Münzen mindestens benötigt werden, um den Geldbetrag A auszuzahlen. Dabei stehen beliebig viele Münzen jeden Typs zur Verfügung.

b) Wende deinen Algorithmus auf $S_1 = 1, S_2 = 4, S_3 = 5$ und $A = 8$ an.

Zum Beispiel ist für Eurocent $S_1 = 1$, $S_2 = 2$, $S_3 = 5$, $S_4 = 10$, $S_5 = 20$ und $S_6 = 50$ und der Algorithmus muss für den Betrag $A = 14$ eine 3 ausgeben, da drei Münzen benötigt werden $(10 + 2 + 2)$. **(10+5 Punkte)**

Aufgabe 3 (Implementierung Knapsack): Implementiere Algorithmen, die das binäre Knapsackproblem lösen mit Hilfe von...

a) ... Brute Force

b) ... Branch & Bound

c) ... Dynamic Programming

Nutze dazu die Javavorlage und die Testfälle, die auf der Vorlesungshomepage[1] zur Verfügung stehen. Welcher Algorithmus kann welche Instanzen (in maximal 10 Minuten) lösen? Wie viel Zeit benötigen sie dafür? (Hinweis: Die Testinsanzen sind teilweise sehr komplex, deine Algorithmen müssen nicht alle lösen können!)

Wir testen deine Software mit
`javac Knapsack1234567.java && java Knapsack1234567 < instance_xyz`

Zur Abgabe: Ersetze `1234567` durch deine Matrikelnummer und gib die Javadatei per Mail an `sfriedr@ibr.cs.tu-bs.de` ab. Nenne in der Mail Name, Matrikel- und Gruppennummer. Es gilt die selbe Frist wie für die anderen Aufgaben. **(5+5+5 Punkte)**

[1] `http://www.ibr.cs.tu-bs.de/courses/ss13/aud2/`