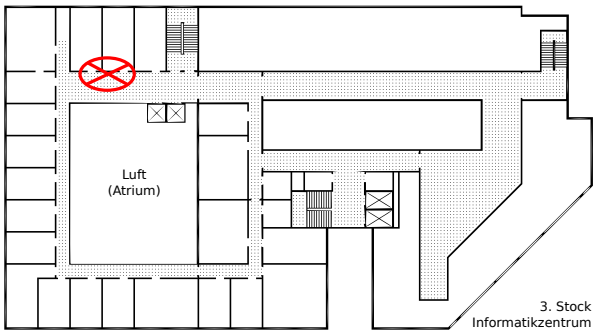
Abteilung Algorithmik **Sommer 2016**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Dr. Christian Scheffer

Algorithmen und Datenstrukturen II
Übung 4 vom 08. 06. 2016

Abgabe der Lösungen bis zum Montag, den 20. 06. 2016 um 13:15 im Hausaufgabenrückgabeschrank.

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



Auf diesem Blatt gibt es 30 Punkte, erreichbar (und abzugeben) sind aber lediglich Aufgaben im Gesamtwert von **maximal 20** Punkten. Mehr wird nicht gewertet! Wähle zwei Aufgaben aus, die Du bearbeitest.

Aufgabe 1 (Branch & Bound): Betrachte Algorithmus 1.19 (Branch & Bound) aus der Vorlesung.

a) Was für Schranken werden in Algorithmus 1.19 verwendet?

b) Beschreibe in eigenen Worten, wie diese Schranken bei Branch & Bound verwendet werden, um die Laufzeit von Algorithmus 1.19 zu verbessern.

c) Wende den Algorithmus Branch & Bound auf die folgende Instanz für $Z = 4$ an

i	1	2	3
Gewicht z_i	2	1	2
Wert p_i	4	3	5.

Gehe hierzu wie folgt vor:

- Male den Entscheidungsbaum auf, der dem Vorgehen von Algorithmus 1.19 für die obige Instanz entspricht.
- Gib UNBEDINGT! für jede Kante an, für welche Entscheidung sie steht.
- Beschrifte jeden Knoten mit der Nummer, die seiner Position in der Verarbeitungsreihenfolge entspricht.
- Gib für jeden Knoten in Deinem Entscheidungsbaum die verwendeten Schranken an. Nummeriere die Knoten in der Reihenfolge, in der sie abgearbeitet werden.

(2+2+6 Punkte)

Aufgabe 2 (Globales Alignment): Betrachte das Problem „Globales Alignment“ aus der großen Übung.

a) Führe den Dynamischen-Programmierungs-Algorithmus für globales Alignment aus der großen Übung auf den Wörtern ABCGAGAA und ABCCADAC mit $\delta = 4$ und $\gamma_{pp'} = 0$, falls $p = p'$ und $\gamma_{pp'} = 3$, falls $p \neq p'$ aus.

b) Wie muss der Dynamische-Programmierungs-Algorithmus für globales Alignment aus der großen Übung modifiziert werden, damit dieser nicht nur den Wert einer optimalen Lösung, sondern auch eine optimale Lösung selbst berechnet?

c) Gib einen Divide-&-Conquer Algorithmus an, der das dynamische Programm für globales Alignment aus der großen Übung umsetzt. Zusätzlich soll Dein Algorithmus Memoisation verwenden.

(5+2+3 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 Deinen entsprechenden Betreuer ab. Nenne in der Mail Name, Matrikel- und Gruppennummer. Es gilt die selbe Frist wie für die anderen Aufgaben. **(3+4+3 Punkte)**

[1] `http://www.ibr.cs.tu-bs.de/courses/ss16/aud2/`