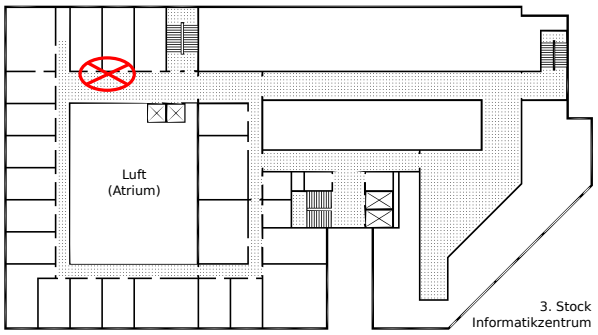
Abteilung Algorithmik **Sommer 2015**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

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

Algorithmen und Datenstrukturen II
Übung 2 vom 06. 05. 2015

Abgabe der Lösungen bis zum Montag, den 18. 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	5	6
Gewicht z_i	3	8	6	7	2	4
Wert p_i	10	40	30	50	10	25

Die Rucksackkapazität ist $Z = 12$.

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 kann die Schleife 2.1 auch in umgekehrter Reihenfolge durchlaufen werden, d.h. in der Reihenfolge $x = z_i - 1$, $x = z_i - 2$,...,$x = 0$.

(7+5+3 Punkte)

Aufgabe 2 (Brückenbau): Betrachte einen Fluss F, der durch eine horizontale Gerade gegeben ist. Des Weiteren seien n Städte $\{1_N, ..., n_N\}$ gegeben, die auf einer Horizontalen h_N nördlich von F liegen. Analog seien n Städte $\{1_S, ..., n_S\}$ gegeben, die auf einer Horizontalen h_S südlich von F liegen, siehe zum Beispiel Abbildung 1. Jede Stadt im Norden hat genau eine Partnerstadt im Süden und anders herum. Eine Brücke zwischen einer

Stadt im Norden und einer Stadt im Süden entspricht dem Segment zwischen beiden Städten.

Die Indizes zweier Partnerstädte müssen nicht übereinstimmen. Zum Beispiel kann 2_N Partnerstadt von 5_S sein.

a) Entwirf einen Dynamic-Programming-Algorithmus, der die Kardinaltät einer möglichst Großen Menge von kreuzungsfreien Brücken bestimmt, die jeweils zwei Partnerstädte miteinander verbinden. Welche Laufzeit hat Dein Algorithmus?

b) Wende Deinen Algorithmus auf die in Abbildung 1 dargestellte Städtesituation an, wobei folgende Partnerschaften gegeben seien: $(1_N, 3_S)$, $(2_N, 5_S)$, $(3_N, 2_S)$, $(4_N, 4_S)$ und $(5_N, 1_S)$.

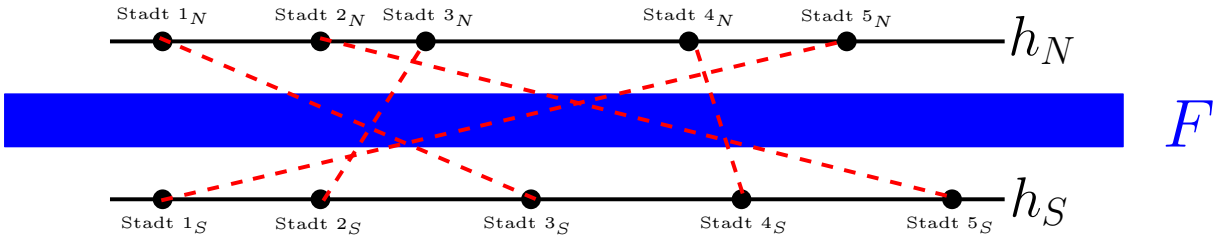


Abbildung 1: Eine Städtekonfiguration: Die gestrichelten Linien deuten die Partnerschaften an.

(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 `scheffer@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/ss15/aud2/`