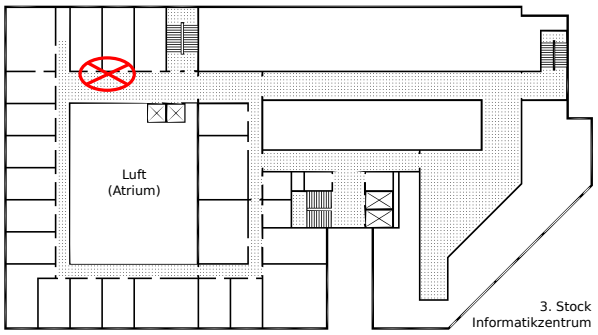
Abteilung Algorithmik **Sommer 2017**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Arne Schmidt

Algorithmen und Datenstrukturen II
Übung 5 vom 19. 06. 2017

Abgabe der Lösungen bis zum Montag, den 03. 07. 2017 um 13:15 im Hausaufgabenrückgabeschrank bei Raum IZ 337. Es werden nur mit einem dokumentechten Stift geschriebene Lösungen gewertet.
Bitte die Blätter zusammenheften und vorne deutlich mit eigenem Namen, Matrikel- und Gruppennummer, sowie Studiengang versehen!



Auf diesem Blatt gibt es 45 Punkte, erreichbar (und abzugeben) sind aber lediglich Aufgaben im Gesamtwert von maximal 30 Punkten. Mehr wird nicht gewertet! Wähle zwei Aufgaben aus, die Du bearbeitest.

Aufgabe 1 (GREEDY$_k$): In dieser Aufgabe betrachten wir den GREEDY$_k$-Algorithmus aus der Vorlesung (Algorithmus 1.26).

a) Wende GREEDY$_k$ auf die folgende Instanz an:

i	1	2	3	4	5
z_i	7	18	23	5	13
p_i	7	18	23	5	13

mit $Z = 42$ und $k = 2$

Gib dazu die folgenden Mengen bzw. Werte tabellarisch an:

- $\overline{S}$
- $\sum_{i \in \overline{S}} z_i$
- $Z - \sum_{i \in \overline{S}} z_i$
- $A_{\overline{S}} := \text{GREEDY}_0(\{z_i | i \notin \overline{S}\}, Z - \sum_{i \in \overline{S}} z_i, \{p_i | i \notin \overline{S}\})$
- $\sum_{i \in \overline{S}} p_i + A_{\overline{S}}$
- G_k
- S

Achte darauf, dass $\overline{S}$ mit der kleinsten Menge anfängt und mit der größten endet. Zusätzlich soll $\overline{S}$ topologisch sortiert sein, das heißt, für zwei gleichgroße Mengen $\overline{S}_1$ und $\overline{S}_2$ kommt $\overline{S}_1$ vor $\overline{S}_2$, falls das kleinste Element $x \in \overline{S}_1 \setminus \overline{S}_2$ kleiner ist als das kleinste Element $y \in \overline{S}_2 \setminus \overline{S}_1$. (*Hinweis:* Die Menge $X \setminus Y$ enthält Elemente aus X, die nicht in Y vorkommen. In der Übung vom 28.06. wird ein Beispiel durchgerechnet.)

b) Was haben der Branch-and-Bound-Algorithmus und $\textsc{Greedy}_k$ für Knapsack mit $k = n$ gemeinsam.

c) Worin unterscheiden sich der Branch-and-Bound-Algorithmus und $\textsc{Greedy}_k$ für Knapsack mit $k = n$?

(*Hinweis:* $\textsc{Greedy}_k$ wird in der Vorlesung vom 22.06.17 vorgestellt.) **(11+2+2 Punkte)**

Aufgabe 2 (Komplexität): Ein großer Onlineshop möchte seinen Dienst mit möglichst wenig Servern betreiben, um Kosten zu sparen. Damit alles läuft, müssen n Jobs mit einer Auslastung von $0 < j_1, \ldots, j_n \leq 1$ dauerhaft ausgeführt werden. Jeder der baugleichen Server kann Jobs mit einer Gesamtauslastung von maximal 1 ausführen und kein Job kann auf mehrere Server aufgeteilt werden.

Ideal wäre ein Algorithmus, der für jede Wahl von $j_1, \ldots, j_n$ bestimmt, wie viele Server gebraucht werden, und wie die Jobs darauf verteilt werden müssen. Leider hat die IT noch keinen effizienten Algorithmus gefunden, sondern lediglich sogenannte Approximationsalgorithmen. Ein ρ-Approximationsalgorithmus, mit $\rho \geq 1$, für das Serverproblem bedeutet, dass maximal $\rho \cdot \textsc{opt}$ Server benutzt werden, wobei $\textsc{opt}$ die kleinste Anzahl an Servern ist, die benötigt werden.

a) Es ist bekannt, dass Partition (Problem 1.9 aus dem Vorlesungsskript) ein NP-vollständiges Problem ist. Hilf der IT und zeige, dass man mit Hilfe eines ρ-Approximationsalgorithmus für das Serverproblem auch jede Instanz von Partition lösen kann, wenn $1 \leq \rho < \frac{3}{2}$ ist.

b) Angenommen, es gibt keinen effizienten Algorithmus der Partition löst, was bedeutet dann das Ergebnis von a) für das Serverproblem?

(12+3 Punkte)

Aufgabe 3 (Implementierung KNAPSACK): Implementiere für das KNAPSACK Problem den GREEDY_k Algorithmus aus der Vorlesung. (*Hinweis:* Die Funktion *Combinations* aus dem commons-math package und die Funktion *ArrayUtils.contains* aus dem commons-lang package können sehr hilfreich sein! [1,2]) Dein Programm soll folgendes ausgeben können:

- $\overline{S}$
- $\sum_{i \in \overline{S}} z_i$
- $Z - \sum_{i \in \overline{S}} z_i$
- $A_{\overline{S}} := \text{GREEDY}_0(\{z_i | i \notin \overline{S}\}, Z - \sum_{i \in \overline{S}} z_i, \{p_i | i \notin \overline{S}\})$
- $\sum_{i \in \overline{S}} p_i + A_{\overline{S}}$ und
- G_k

Nutze dazu die Javavorlage und die Testfälle, die auf der Vorlesungsseite[3] zur Verfügung stehen.

a) Wählen wir nun $k = 5$. Welche Instanzen kannst Du innerhalb von 10 Minuten lösen? Verzichte dazu auf die Ausgabe der oben genannten Punkte. *Hinweis:* Dein Algorithmus muss nicht alle Testinstanzen lösen, da diese teils sehr komplex sind.

b) Welche Laufzeit, abhängig von k und n, besitzt GREEDY_k? Erläutere kurz Deine Antwort! (*Hinweise:* $\binom{a}{b} \leq a^b$ und $\sum_{i=0}^{k} n^i = \frac{n^{k+1} - 1}{n - 1}$)

Wir testen Deine Software mit
javac -cp *:. Knapsack1234567.java && java *:. Knapsack1234567 < instance_xyz

Zur Abgabe des Programms: 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 dieselbe Frist wie für die anderen Aufgaben.

(*Hinweis:* GREEDY_k wird in der Vorlesung vom 22.06.17 vorgestellt.) **(10+2+3 Punkte)**

[1] `http://commons.apache.org/proper/commons-math/javadocs/api-3.4/org/apache/commons/math3/util/Combinations.html`

[2] `https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/ArrayUtils.html`

[3] http://www.ibr.cs.tu-bs.de/courses/ss17/aud2/