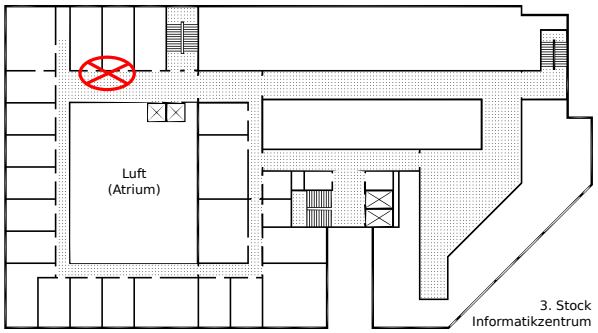
Abteilung Algorithmik **Sommer 2018**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Arne Schmidt

Algorithmen und Datenstrukturen II
Übung 5 vom 21. 06. 2018

Abgabe der Lösungen bis zum Donnerstag, den 05. 07. 2018 um 13:15 im Hausaufgabenschrank bei Raum IZ 337. Es werden nur mit einem dokumentenechten Stift (kein Rot!) geschriebene Lösungen gewertet.
Bitte die Blätter zusammenheften und vorne deutlich mit eigenem Namen, Matrikel- und Gruppennummer, sowie Studiengang versehen!



Dieses Blatt ist in drei Bereiche unterteilt: Allgemeiner Teil (A), theoretischer Teil (T) und praktischer Teil (P). Abgegeben muss der A-Teil und entweder T oder P. Mögliche Kombinationen sind also A und T bzw. A und P mit einem Gesamtwert von jeweils 30 Punkten.

A — Allgemeiner Teil

Aufgabe 1 (GREEDY$_k$): **(13 Punkte)**

In dieser Aufgabe betrachten wir den GREEDY$_k$-Algorithmus aus der Vorlesung (Algorithmus 1.26). Wende GREEDY$_k$ auf die folgende Instanz an. (Hinweis: Die Objekte sind bereits nach $\frac{z_i}{p_i}$ sortiert; d.h. GREEDY$_0$ wird nichts mehr sortieren müssen.)

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 $\overline{S}$ lexikographisch sortiert sein: 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.)

Aufgabe 2 (Approximation mit GREEDY$_0$): **(5+2 Punkte)**
Betrachte den Algorithmus GREEDY$_0$. Angenommen, jedes Objekt hat ein Gewicht von höchstens $\frac{Z}{\alpha}$ für ein festes $\alpha \in \{1, 2, 3, 4, \dots\}$. Wir nehmen außerdem an, dass $z_i = p_i$ für jedes Objekt i gilt und dass die Summe aller z_i die Kapazität Z überschreitet.

a) Zeige: Nach Anwenden von GREEDY$_0$ gilt $\sum_{i=1}^{n} x_i z_i \geq Z(1 - \frac{1}{\alpha})$.

b) Welche Approximationsgüte besitzt GREEDY$_0$ mit diesen speziellen Objekten? (Hinweis: Die Güte soll abhängig von α angegeben werden.)

T — Theoretischer Teil

Aufgabe 3 (Komplexität): **(2+6+2 Punkte)**
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 $0 < j_1, \dots, j_n \leq 1$ 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 $j_1, \dots, j_n$ 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.

a) Was bedeutet ein ρ-Approximationsalgorithmus für das Serverproblem?

b) Es ist bekannt, dass PARTITION 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.

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

P — Praktischer Teil

Aufgabe 4 (Implementierung KNAPSACK): **(10 Punkte)**
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])

Dein Programm soll folgendes ausgeben können:

[1] `http://commons.apache.org/proper/commons-math/javadocs/api-3.4/org/apache/commons/math3/util/Combinations.html` und `https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/ArrayUtils.html`

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

Nutze dazu die Javavorlage und die Testfälle, die auf der Vorlesungsseite[2] zur Verfügung stehen.

a) Löse so viele Instanzen wie möglich für $k \in \{0, 1, 2\}$ (die Lösungen werden in csv-Dateien gespeichert). Verzichte dabei auf die Ausgabe der oben genannten Punkte. (Hinweis: Da für $k = 2$ die Zeit für große Instanzen mehr als 10 Minuten betragen kann, müssen nicht alle gelöst werden. Es bietet sich also an, dem Programm für jede Instanz ein Zeitlimit von 10 Minuten zu übergeben. (Das Zeitlimit darf natürlich auch weggelassen werden, um alle Instanzen zu lösen.) Referenzwert: $k = 2$, $n = 10000$ benötigt etwa 20 Minuten bei einer Rechenleistung von 2.3 GHz.)

b) Wie schnell wird die Lösung berechnet? Erstelle dazu folgenden *Scatter-Plot*:

- Die x-Achse entspricht der Anzahl der Objekte
- Die y-Achse entspricht dem Quotienten aus der Zeit, die zum Lösen benötigt wurde, und n^{k+1}.
- Es sollen alle Datenpunkte für $k \in \{0, 1, 2\}$ im Plot erkennbar sein. (Hier können zum Beispiel andere Farben und/oder Symbole für die Datenpunkte gewählt werden.)

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

Zur Abgabe: Ersetze 1234567 durch Deine Matrikelnummer und gib sowohl die Javadatei, die vom Programm generierten csv-Dateien als auch den zu konstruierenden Scatter-Plot (z.B. als pdf-Datei) 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:* $\textsc{Greedy}_k$ wird in der Vorlesung vom 21.06.17 vorgestellt.)

[2] `http://www.ibr.cs.tu-bs.de/courses/ss18/aud2/`