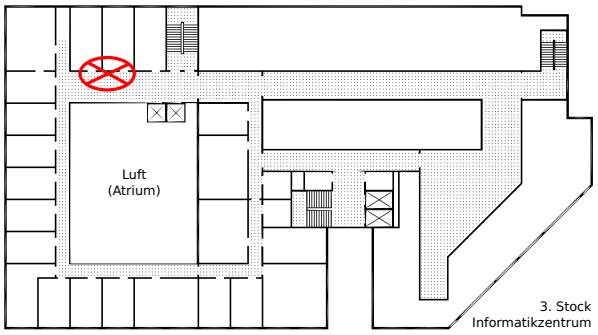
**Abteilung Algorithmik** **Sommer 2018**
**Institut für Betriebssysteme und Rechnerverbund**
**TU Braunschweig**

Prof. Dr. Sándor P. Fekete
Arne Schmidt

# Algorithmen und Datenstrukturen II
# Übung 4 vom 07. 06. 2018

Abgabe der Lösungen bis zum Donnerstag, den 21. 06. 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. Die Auswahl gilt nur für dieses Blatt, d.h. auf dem nächsten Blatt kann sich wieder umentschieden werden.

## A — Allgemeiner Teil

**Aufgabe 1 (Branch and Bound - Knapsack): (11+3+3+3 Punkte)**
In dieser Aufgabe betrachten wir den Branch-and-Bound-Algorithmus für Maximum Knapsack aus der Vorlesung (Algorithmus 1.19) mit dem Berechnen von $\text{Greedy}_0$ (siehe Blatt 1) als untere Schranke und dem Greedy-Algorithmus für Fractional Knapsack (Algorithmus 1.4) als obere Schranke.

a) Betrachte folgende, vorsortierte Instanz für Maximum Knapsack:

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $z_i$ | 24 | 6 | 28 | 18 |
| $p_i$ | 24 | 6 | 28 | 18 |

und $Z = 42$

Wende den Branch-and-Bound-Algorithmus auf diese Instanz an. Gib dazu den Entscheidungsbaum an, den der Algorithmus konstruiert, indem Du die Kanten und Knoten in dem Baum aus Abbildung 1 auf Seite 4 folgendermaßen beschriftest: Beschrifte die Kanten mit der Entscheidung, die getroffen wurde ($b_l = 0$ oder 1) und die Knoten mit der aktuell geltenden oberen Schranke ($U$) und mit der bisher besten unteren Schranke ($P$). Falls eine aktuelle Belegung nicht zulässig ist, dann beschrifte den Knoten mit *unzulässig*. Halte außerdem in einer Tabelle fest, bei

welcher Belegung der Wert von $P$ aktualisiert wird. Falls an einem Knoten nicht weiter verzweigt wird, streiche die beiden anliegenden Kanten durch. (Hinweis: Es muss nicht der ganze Baum beschriftet werden. In der Übung am 06.06.17 wird ein Beispiel durchgerechnet!)

b) Betrachte einen Knoten $v$ in einem Entscheidungsbaum, für den die gültige Belegung $b_1, \ldots, b_{l-1}$ gilt. Sei $U := UB(b_1, \ldots, b_{l-1})$ eine dazu passende **obere Schranke**. Für welche Teilbäume gilt $U$ **immer** als obere Schranke?

c) Betrachte einen Knoten $v$ in einem Entscheidungsbaum, für den die gültige Belegung $b_1, \ldots, b_{l-1}$ gilt. Sei $P := LB(b_1, \ldots, b_{l-1})$ eine dazu passende **untere Schranke**. Für welche Teilbäume gilt $P$ **immer** als untere Schranke?

d) Zeige: Für jedes $n \geq 1$ gibt es mindestens eine Instanz mit $n$ Objekten, bei der der Branch-and-Bound-Algorithmus keine rekursiven Aufrufe startet.

# T — Theoretischer Teil

**Aufgabe 2 (Mehr Branch-and-Bound für KNAPSACK):** **(5 Punkte)**
Betrachte den Branch-and-Bound-Algorithmus für KNAPSACK mit dem Berechnen von Schranken wie in Aufgabe 1. Zeige oder widerlege: Falls direkt vor der Verzweigung $U = P$ gilt, haben wir eine optimale Lösung gefunden.

**Aufgabe 3 (Maximal Exact Set Cover):** **(5 Punkte)**
Betrachte folgendes Problem:

**Gegeben:** Eine Menge $\mathcal{U} := \{o_1, \ldots, o_n\}$ von $n$ Objekten und eine Menge $\mathcal{F}$ von Teilmengen von $\mathcal{U}$, d.h. $\mathcal{F} \subseteq \{M \mid M \subseteq \mathcal{U}\}$.

**Gesucht:** Eine Menge $S \subseteq \mathcal{F}$, sodass die Anzahl der Elemente in der Vereinigung aller Mengen in $S$ maximal ist und keine zwei Mengen aus $S$ Elemente gemeinsam haben.
Formal: $\max_{S \subseteq \mathcal{F}} \left( x \mid x = \sum_{s \in S} |s| \wedge \forall s_1, s_2 \in S : s_1 \cap s_2 = \emptyset \right)$

Sei beispielsweise $\mathcal{U} = \{1, 2, 3, 4, 5\}$ und $\mathcal{F} := \{\{1, 2\}, \{1, 3\}, \{3, 4\}, \{4, 5\}\}$. Man kann sich schnell davon überzeugen, dass $S = \{\{1, 2\}, \{3, 4\}\}$ eine beste Lösung ist.

Ein möglicher Branch-and-Bound-Algorithmus (siehe Algorithmus 1) prüft nun als erstes, ob die aktuelle Auswahl der Mengen aus $\mathcal{F}$ gültig ist (Zeile 2). Wird uns *true* zurückgegeben, prüfen wir, ob die aktuelle Auswahl eine Verbesserung mit sich bringt (Zeile 3-4). Befinden wir uns in einem Blatt des Entscheidungsbaumes, brauchen wir nicht weiter verzweigen (Zeile 5). Andernfalls verzweigen wir, wenn wir noch die Möglichkeit besitzen, Objekte zu überdecken (Zeilen 6-10).

Betrachten wir nun die Funktion VALIDSET etwas genauer. Hier prüfen wir, ob es eine Menge $F_j$ gibt, für die $b_j = 1$ gilt und Elemente enthält, die auch in $F_{\ell-1}$ liegen (sofern $b_{\ell-1} = 1$ gilt).

Warum reicht es, $F_{\ell-1}$ gegen alle vorherigen Mengen zu testen, und warum müssen nicht alle Mengen paarweise verglichen werden?

```
1: function SetCover(ℓ)
2:     if not validSet(ℓ) then return
3:     Sei P := Σ_{j=1}^{ℓ−1} b_j|F_j|
4:     if P > N then N := P;
5:     if ℓ > |F| then return
6:     if n > N then
7:         b_ℓ = 0; SetCover(ℓ + 1);
8:         b_ℓ = 1; SetCover(ℓ + 1);
```

```
1: function validSet(ℓ)
2:     if b_{ℓ−1} = 0 then return true
3:     for j = 1 to ℓ − 2 do
4:         if b_j = 1 and |F_j ∩ F_{ℓ−1}| > 0 then
5:             return false
6:     return true
```

**Algorithmus 1:** Branch-and-Bound-Algorithmus für Maximal Exact Set Cover. Der Algorithmus bekommt folgende Werte übergeben: Objektmenge $\mathcal{U} = \{o_1, \ldots, o_n\}$, Menge von Teilmengen $\mathcal{F}$, den besten bisher bekannten Wert $N$, die Rekursionstiefe $\ell$ und die fixierten Objekte $b_j \in \{0, 1\}$ für $j \in \{1, \ldots, \ell - 1\}$

# P — Praktischer Teil

**Aufgabe 4 (Implementierung Knapsack): (10 Punkte)**
Implementiere für Knapsack den Branch-and-Bound-Algorithmus aus der Vorlesung (Algorithmus 1.19) mit dem Berechnen von $\text{Greedy}_0$ (siehe Blatt 1) als untere Schranke und dem Greedy-Algorithmus für Fractional Knapsack (Algorithmus 1.4) als obere Schranke.

Nutze dazu die Javavorlage und die Testfälle, die auf der Vorlesungsseite[1] zur Verfügung stehen.

a) Löse **alle zufällig generierten** Instanzen, d.h. diejenigen, die sich im Ordner *random* befinden (die Lösungen werden in csv-Dateien gespeichert).

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

- Die $x$-Achse entspricht der Anzahl der Objekte
- Die $y$-Achse entspricht der Zeit, die zum Lösen benötigt wurde.

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

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.

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

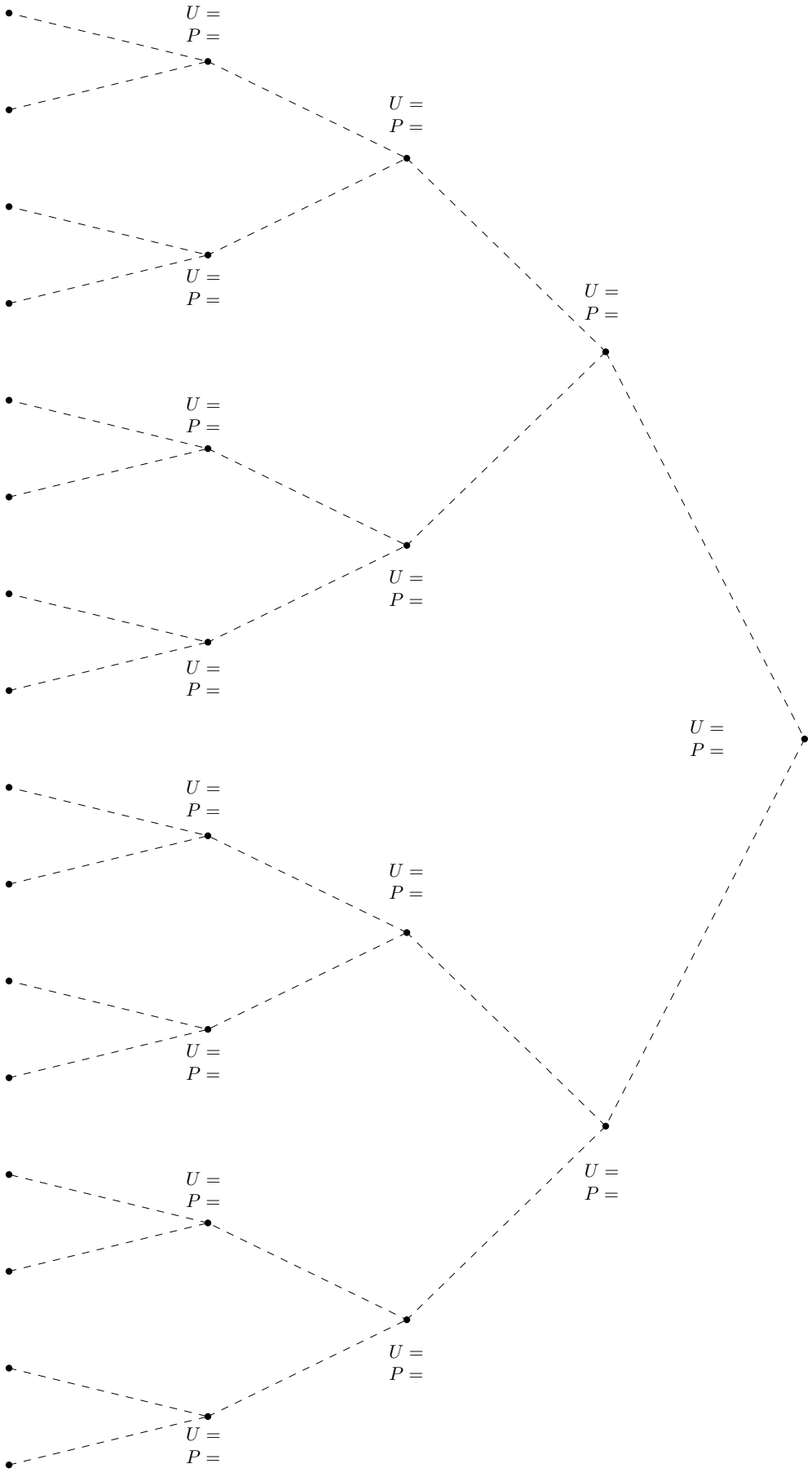



**Abbildung 1:** Ein Entscheidungsbaum.