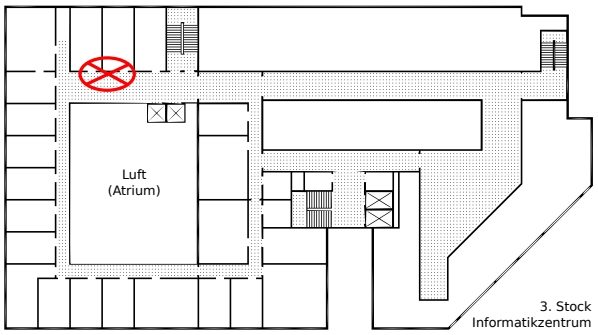
Abteilung Algorithmik **Sommer 2017**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Arne Schmidt

Algorithmen und Datenstrukturen II
Übung 4 vom 29. 05. 2017

Abgabe der Lösungen bis zum Montag, den 19. 06. 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 (Branch and Bound - Knapsack): In dieser Aufgabe betrachten wir den Branch-and-Bound-Algorithmus für Maximum Knapsack aus der Vorlesung (Algorithmus 1.19 inklusive dem optionalen Teil).

a) Betrachte folgende Instanz für Maximum Knapsack:

i	1	2	3	4	5
z_i	13	5	23	7	17
p_i	13	5	23	7	17

und $Z = 42$

Wende den Branch-and-Bound-Algorithmus (Algorithmus 1.19) mit dem optionalen, roten Teil 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 und mit der bisher besten unteren Schranke. 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.

(*Hinweise:* Es muss nicht der ganze Baum beschriftet werden. In der Übung am 31.05.17 wird ein Beispiel durchgerechnet! Benutze zum Berechnen einer unteren

Schranke den Greedy-Algorithmus mit der Verbesserung, dass der Algorithmus Objekte überspringt, die nicht mehr reinpassen, anstatt komplett abzubrechen. Betrachte für eine obere Schranke die fraktionale Variante von Knapsack und den dazu passenden Greedy-Algorithmus.)

b) Betrachte einen Knoten v in einem Entscheidungsbaum. Für diesen Knoten gibt es eine Belegung $b_1, \ldots, b_{l-1}$ und die dazugehörige **obere Schranke** $U := UB(b_1, \ldots, b_{l-1})$. Für welche Teilbäume gilt U **immer** als obere Schranke?

c) Betrachte einen Knoten v in einem Entscheidungsbaum. Für diesen Knoten gibt es eine Belegung $b_1, \ldots, b_{l-1}$ und eine dazu passende **untere Schranke** $L := LB(b_1, \ldots, b_{l-1})$. Für welche Teilbäume gilt L **immer** als untere Schranke?

(11+2+2 Punkte)

Aufgabe 2 (Branch and Bound - Maximal Exact Set Cover): Betrachte folgendes Problem: Gegeben ist eine Menge $\mathcal{U} := \{o_1, \ldots, o_n\}$ von n Objekten und eine Menge $\mathcal{F}$ von Teilmengen von $\mathcal{U}$, d.h. $\mathcal{F} \subseteq \{M | M \subseteq \mathcal{U}\}$.

Gesucht ist eine Menge $S \subseteq \mathcal{F}$, sodass die Vereinigung aller Mengen in S maximal ist und keine zwei Mengen aus S Elemente gemeinsam haben.

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.

Wir möchten nun einen Branch-and-Bound-Algorithmus konstruieren, welcher das oben beschriebene Problem löst. Sei validSet(l) eine Methode, die überprüft, ob unter den ersten $l - 1$ ausgewählten Elementen keine zwei Mengen existieren, die mindestens ein gemeinsames Element besitzen.

a) Entwirf einen Branch-and-Bound-Algorithmus SetCover(l), der das Problem löst und die Methode validSet(l) zum Überprüfen einer gültigen Lösung benutzt.

(*Hinweise:* Die Methode validSet(l) soll als Black-Box verwendet werden; man muss also nicht wissen, was validSet(l) im Detail durchführt. Eine obere Schranke für das Problem ist immer n.)

b) Entwirf einen möglichst schnellen Algorithmus für validSet(l), der entscheidet, ob die derzeitige Auswahl der ersten $l - 1$ Teilmengen eine zulässige Lösung ist. Begründe kurz die Korrektheit Deines Algorithmus!

(*Hinweise:* $S_1 \cap S_2$ gibt eine Menge von Elementen zurück, die in S_1 und S_2 liegen. $|S|$ gibt die Anzahl der Elemente in S zurück. Bevor validSet(l) aufgerufen wird, gab es den Aufruf validSet($l-1$) mit der gleichen Belegung der Teilmengen, aber ohne Teilmenge $l - 1$.)

c) Um eine untere Schranke zu berechnen, kann man eine beliebige gültige Lösung konstruieren. Eine Möglichkeit bieten Greedy-Algorithmen wie bei Knapsack. Beschreibe kurz wie ein möglicher Greedy-Algorithmus für das oben beschriebene Problem aussehen kann. (Die Beschreibung des Algorithmus sollte dabei keinesfalls länger als eine halbe Seite sein!)

(4+8+3 Punkte)

Aufgabe 3 (Implementierung KNAPSACK): Implementiere für KNAPSACK den Branch-and-Bound-Algorithmus aus der Vorlesung (Algorithmus 1.19), wobei **keine zusätzlichen unteren Schranken** berechnet werden sollen.

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

a) Welche Instanzen kannst Du innerhalb von 10 Minuten lösen? *Hinweis:* Dein Algorithmus muss nicht alle Testinstanzen lösen, da diese teils sehr komplex sind.

b) Betrachte den Branch-and-Bound-Algorithmus ohne Berechnen von unteren Schranken. Zeige oder widerlege: Falls direkt vor der Verzweigung $U = P$ gilt, haben wir eine optimale Lösung gefunden.

Wir testen Deine Software mit
javac -cp *:. Knapsack1234567.java && java -cp *:. 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 dieselbe Frist wie für die anderen Aufgaben. **(10+1+4 Punkte)**

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

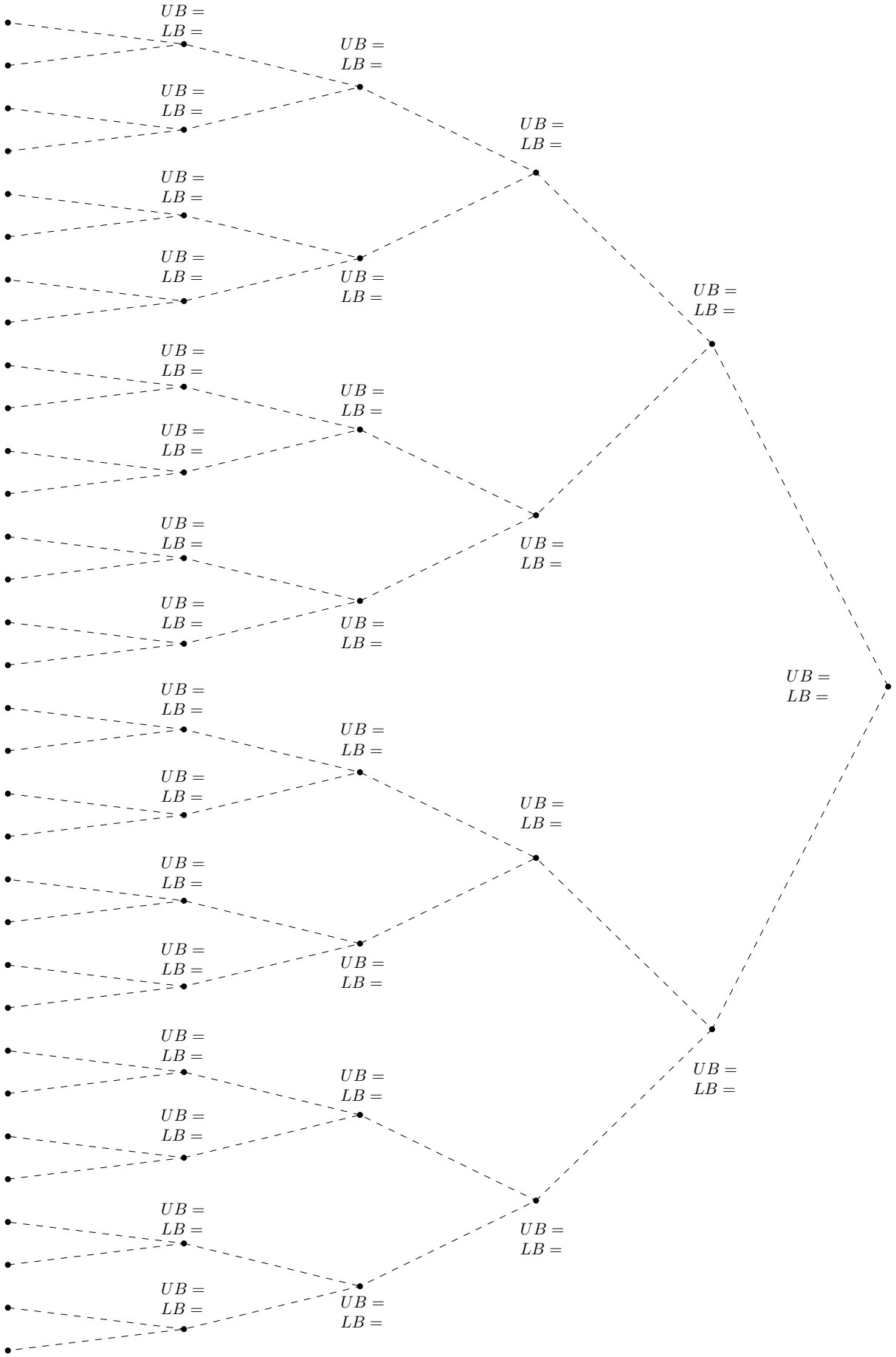


Abbildung 1: Ein Entscheidungsbaum.