Algorithmen und Datenstrukturen 2 **SoSe 2021**
Prof. Dr. Sándor Fekete **Abgabe:** 15.06.21
Matthias Konitzny **Rückgabe:** ab 21.06.21

Hausaufgabenblatt 3

Die Abgabe der Lösungen muss bis zum 15.06.21 um 23:59 Uhr erfolgen. Lösungen müssen per Mail in einer PDF-Datei (Dateiname „`blatt_[nr]_[name1]_[name2]_[grpnr].pdf`“) an die/den jeweilige*n Tutor*in geschickt werden. Email-Adressen sind unter `https://www.ibr.cs.tu-bs.de/alg/index.html` zu finden.

Hausaufgabe 1 (Branch-and-Bound): **(6+3 Punkte)**
In dieser Aufgabe betrachten wir den Branch-and-Bound-Algorithmus für Maximum Knapsack aus der Vorlesung. Wir benutzen $\textsc{Greedy}_0$ (siehe Blatt 1) als untere Schranke und den (abgerundeten Wert des) Greedy-Algorithmus für Fractional Knapsack als obere Schranke.

a) Wende den Branch-and-Bound-Algorithmus auf folgende, nach $\frac{z_i}{p_i}$ aufsteigend sortierte Instanz für Maximum Knapsack an.

i	1	2	3	4
z_i	14	15	8	13
p_i	16	15	7	10

und $Z = 28$

Beachte folgende Punkte:
- Benutze den Enumerationsbaum aus Abbildung 1.[1]
- Beschrifte die Kanten mit der Auswahl, die getroffen wurde.
- Beschrifte die Knoten mit den aktuell besten Schranken (obere und untere).
- Ist eine Auswahl unzulässig, beschrifte den jeweiligen Knoten mit *unzulässig*.
- Sollten Kanten nicht benutzt werden, streiche sie durch.
- Halte in einer Tabelle fest, welche Objekte eine neue, beste Lösung liefern.

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

Hausaufgabe 2 (Gewichtetes Hörsaal-Problem): **(3+3 Punkte)**
Betrachte das gewichtete Hörsaal-Problem:

Gegeben Intervalle $(s_1, e_1), \ldots, (s_n, e_n)$ mit $e_1 \leq e_2 \leq \cdots \leq e_n$ und Gewichten $w_1, \ldots, w_n$

Gesucht Indexmenge $S \subseteq \{1, \ldots, n\}$ an disjunkten Intervallen mit $\sum_{i \in S} w_i$ maximal.

Sei $\mathcal{G}(i)$ der größte Wert, wenn die ersten i Intervalle zur Verfügung stehen. Betrachte die folgende Rekursionsgleichung, die für $\mathcal{G}(i)$ immer einen optimalen Wert zurückgibt:

$$\mathcal{G}(i) = \begin{cases} 0 & \text{, falls } i = 0 \\ \max\left(\mathcal{G}(i-1), \mathcal{G}\left(p(i)\right) + w_i\right) & \text{, sonst} \end{cases}$$

[1] Unter `https://aud2.ibr.cs.tu-bs.de` gibt es den Baum zusätzlich als .ipe und .svg, um die Daten direkt einzutragen.

Dabei ist $p(i) = \max(\{j | e_j \leq s_i\} \cup \{0\})$. ($p(i)$ gibt also den größten Index des Intervalls zurück, das am spätesten, aber noch vor Intervall I_i endet.)

a) Entwirf einen Algorithmus in Pseudocode, der $p(0), \ldots, p(n)$ bestimmt.

b) Entwirf einen effizienten Algorithus in Pseudocode, der $\mathcal{G}(n)$ berechnet.

Hausaufgabe 3 (Dynamische Programmierung für KNAPSACK): (5 Punkte)
Wende das dynamische Programm für MAXIMUM KNAPSACK aus der Vorlesung auf folgende Instanz an:

i	1	2	3	4	5
z_i	3	4	2	7	6
p_i	1	4	1	4	4

und $Z = 12$.

Fülle dazu die folgende Tabelle aus, wobei der Eintrag in Zeile i und Spalte x dem Wert $\mathcal{P}(x, i)$ entspricht.

i \ x	0	1	2	3	4	5	6	7	8	9	10	11	12
0													
1													
2													
3													
4													
5													

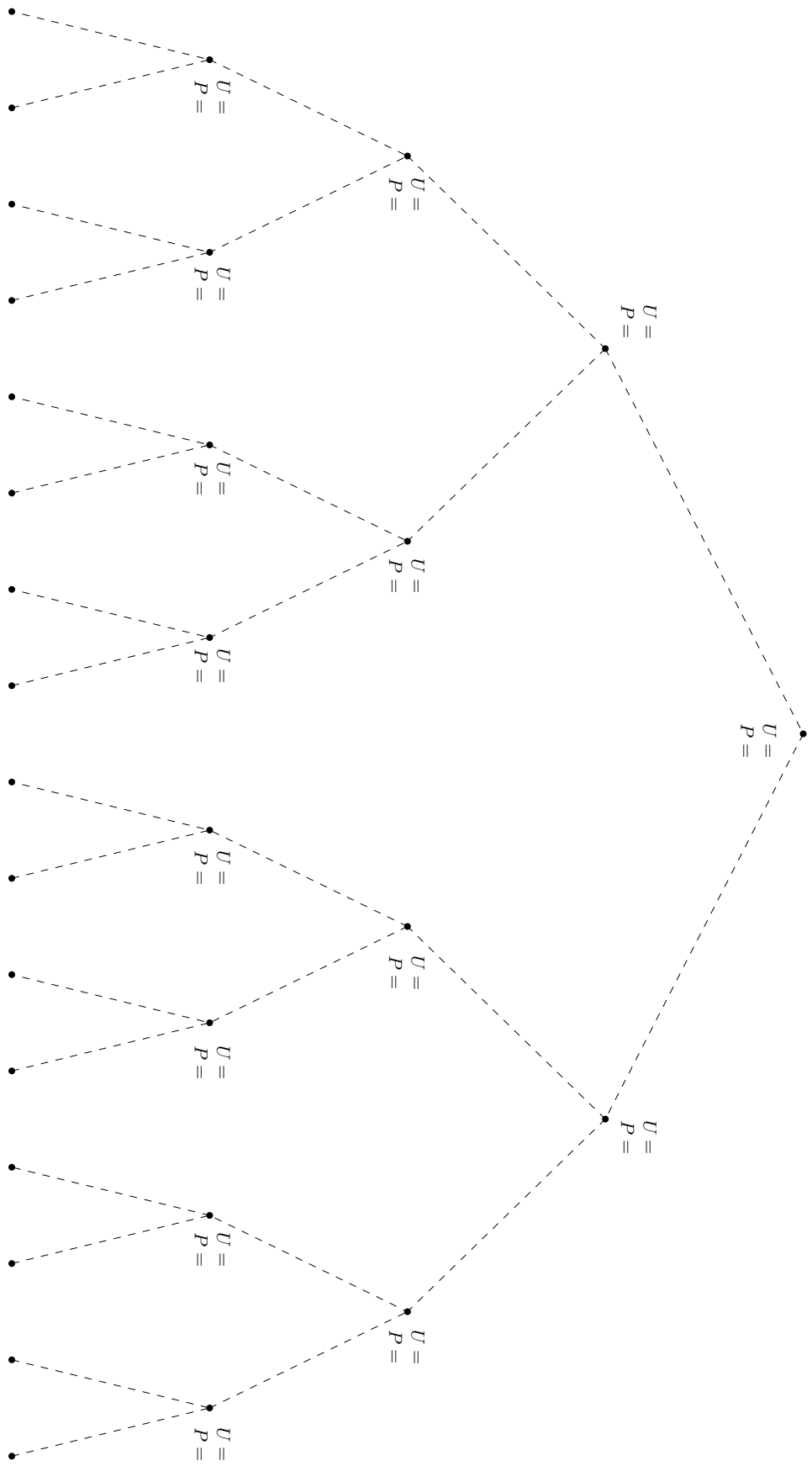

Abbildung 1: Ein Entscheidungsbaum.