Technische Universität Braunschweig

Algorithmen und Datenstrukturen 2

Arne Schmidt

Beispiel 1.1 (Sortiert nach z_i/p_i)

Zeitschranke: 120 Punktschranke: 44

Aufgabe i	6	4	13	12	9	3	15	8	16	10	1	14	5	11	2	7
Zeit z_i	4	8	16	20	8	40	40	40	24	32	20	20	16	28	32	32
Punkte p_i	4	5	10	9	2	10	10	9	4	5	3	3	2	3	3	2

Wir wählen Aufgaben 3, 4, 6, 9, 12, 13 und zu einem Teil Aufgabe 15.
Zu wie vielen Teilen können wir Aufgabe 15 wählen?
3, 4, 6, 9, 12, 13 kosten 96 Minuten; bleiben noch 24 → Aufgabe 15 kann zu 24 von 40 Minuten gelöst werden (Anteil 0.6) und bringt damit $\frac{24}{40} \cdot 10 = 0.6 \cdot 10 = 6$.

Zusammen:
120 Punkte, 46 Minuten

Greedy für Fractional Knapsack

Algorithmus 1.4 Greedy-Algorithmus für FRACTIONAL KNAPSACK

Eingabe: $z_1, ..., z_n, Z, p_1, ..., p_n$

Ausgabe: $x_1, ..., x_n \in [0, 1]$

mit

$$\sum_{i=1}^{n} z_i x_i \leq Z$$

und

$$\sum_{i=1}^{n} p_i x_i = Maximal$$

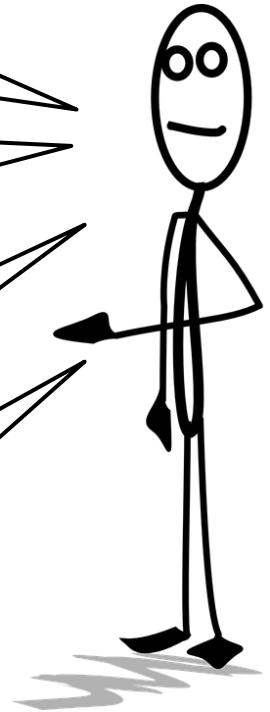
1: Sortiere {1,...,n} nach $\frac{z_i}{p_i}$ aufsteigend;
Dies ergibt die Permutation $\pi(1), ..., \pi(n)$.
Setze $j = 1$.
2: **while** $(\sum_{i=1}^{j} z_{\pi(i)} \leq Z)$ **do**
3: $x_{\pi(j)} := 1$
4: $j := j + 1$
5: Setze $x_{\pi(j)} := \frac{Z - \sum_{i=1}^{j-1} z_{\pi(i)}}{z_{\pi(j)}}$
6: **return**

Verfahren:
Sortiere alle Elemente

Gehe alle Elemente in dieser Reihenfolge durch.

Nimm alle der Reihenfolge auf.

Das erste, das nicht mehr passt, wird anteilig gepackt.



Korrektheit, Laufzeit und Optimalität

Satz 1.5:
Algorithmus 1.4 liefert eine optimale Lösung für Problem 1.3 (Fractional Knapsack) in Zeit $O(n \log n)$.

Wir wollen drei Teile zeigen:
(1) Laufzeit ist $O(n \log n)$.
(2) Wir erhalten eine zulässige Lösung.
(3) Es gibt keine bessere zulässige Lösung.

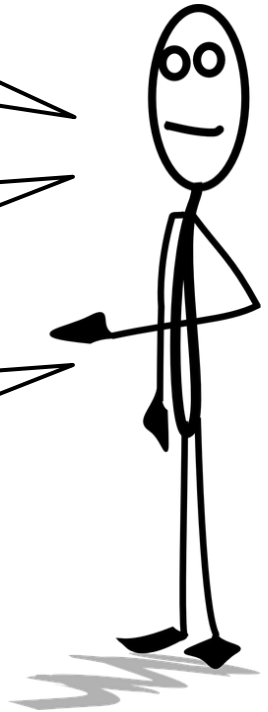
Dazu: ab an die Tafel!

Greedy-Algorithmen

Greedy-Algorithmen wählen immer die **lokal beste Option**.

Vergangene Entscheidungen spielen keine Rolle mehr.

Sie bieten oft „gute Lösungen“.



Gotta go fast

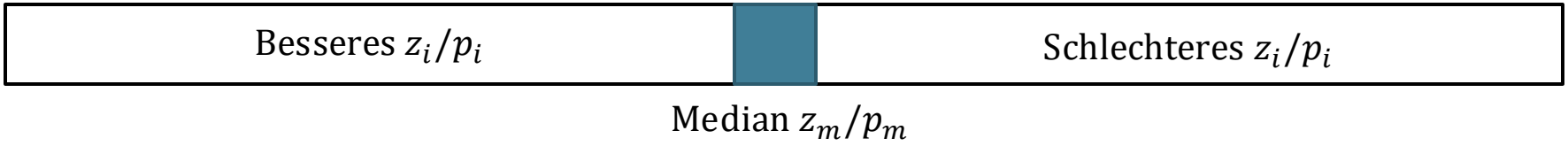
Geht das noch schneller?

Wir müssen nur das beste Objekt finden, das als erstes nicht mehr passt.

Zeitschranke: 120 Punktschranke: 44

Aufgabe i	**6**	**4**	**13**	**12**	**9**	**3**	**15**	**8**	**16**	**10**	**1**	**14**	**5**	**11**	**2**	**7**
Zeit z_i	4	8	16	20	8	40	40	40	24	32	20	20	16	28	32	32
Punkte p_i	4	5	10	9	2	10	10	9	4	5	3	3	2	3	3	2

Linearzeit Algorithmus (Idee)



2 Situationen:

1. Alle Objekte mit $\frac{z_i}{z_p} < \frac{z_m}{p_m}$ passen zusammen in den Rucksack.
2. Alle Objekte mit $\frac{z_i}{z_p} < \frac{z_m}{p_m}$ passen nicht zusammen in den Rucksack.

In Fall 1: Nimm linke Hälfte in den Rucksack auf und wiederhole Vorgehen auf dem Rest.
In Fall 2: Lösche alle Objekte mit $\frac{z_i}{p_i} \geq \frac{z_m}{p_m}$ und wiederhole Vorgehen auf dem Rest.

Laufzeit: $T(n) = T\left(\frac{n}{2}\right) + \Theta(n) \Rightarrow T(n) \in \Theta(n)$

Weitere Probleme

Gibt es noch andere Knapsack-Probleme?

Integer Knapsack

Problem 1.6 (INTEGER KNAPSACK).

Gegeben:

- n Objekte $1, ..., n$ mit jeweils Größe z_i Gewinn p_i
- Größenschranke Z

Gesucht:

Für jedes Objekt ein Wert

$$x_i \in \mathbb{N}$$

mit

$$\sum_{i=1}^{n} z_i x_i \leq Z$$

und

$$\sum_{i=1}^{n} p_i x_i = Maximal$$

Spezialfall: „Isodens“ Knapsack

Problem 1.7.
Gegeben:

- n Objekte $1, ..., n$ mit jeweils Größe z_i
- Größenschranke Z

Gesucht:
Eine Menge

$$S \subseteq \{1, ..., n\}$$

mit

$$\sum_{i \in S} z_i \leq Z$$

und

$$\sum_{i \in S} z_i = Maximal$$

Spezialfall des Spezialfalls: Subsetsum

Alle Objekte besitzen gleiche Dichte

Zielschranke soll exakt erreicht werden.

Problem 1.8 (SUBSET SUM).

Gegeben:

- n Objekte $1, ..., n$ mit jeweils Größe z_i
- Zielgröße Z

Gesucht:

Eine Menge

$$S \subseteq \{1, ..., n\}$$

mit

$$\sum_{i \in S} z_i = Z$$

Spezialfall^3 : Partition

Problem 1.9 (PARTITION).

Gegeben:

- n Objekte $1, ..., n$ mit jeweils Größe z_i

Gesucht:

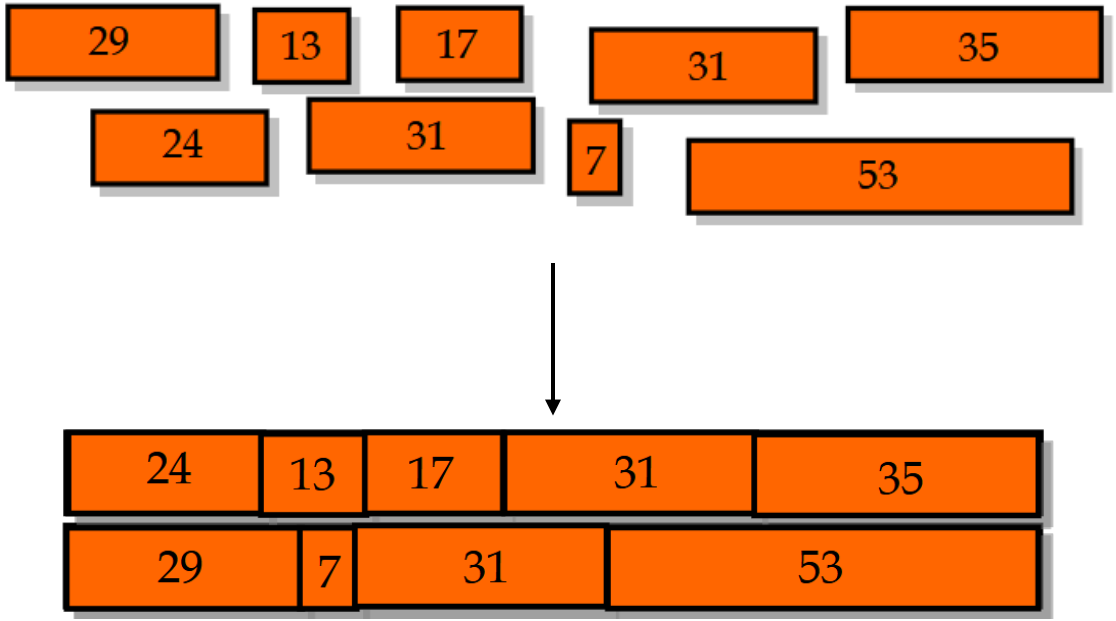
Eine Menge

$$S \subseteq \{1, ..., n\}$$

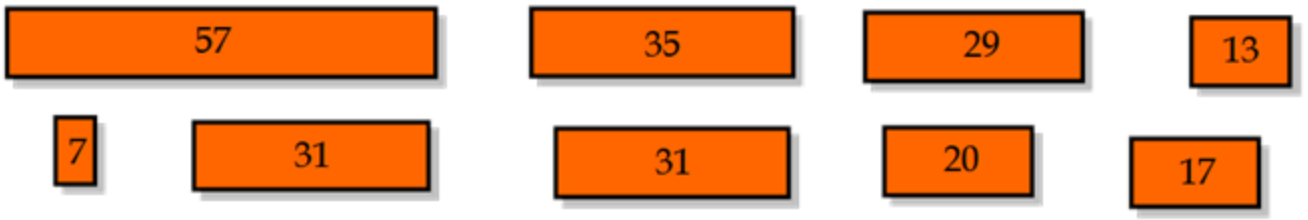
mit

$$\sum_{i \in S} z_i = \sum_{i \notin S} z_i$$

Rückblick: Algorithmen und Datenstrukturen

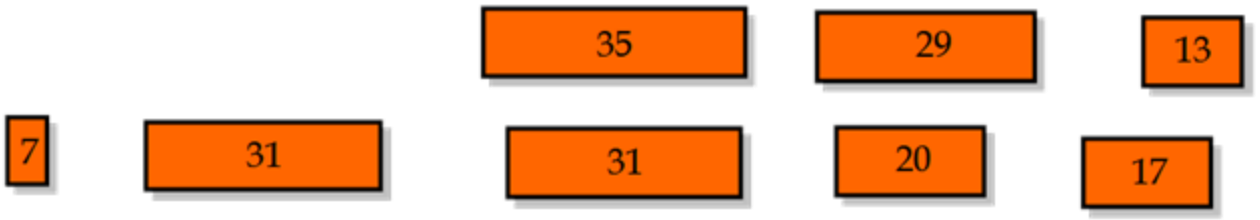


Eine andere Instanz



Eine andere Instanz

Rest: 63



Rest: 120

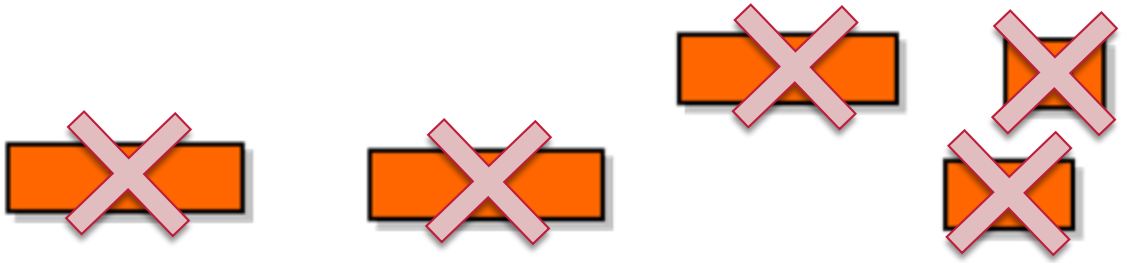
Eine andere Instanz

Rest: 28

Rest: 120

Eine andere Instanz

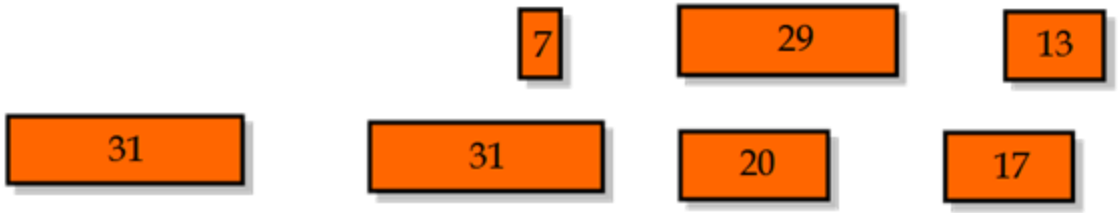
Rest: 1



Rest: 120

Eine andere Instanz

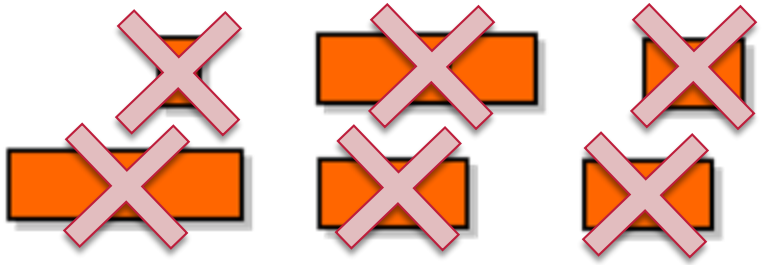
Rest: 63



Rest: 85

Eine andere Instanz

Rest: 32



Rest: 85

Eine andere Instanz

Rest: 63

Rest: 23

Eine andere Instanz

Rest: 34

Muss hier sein!

Rest: 23

Eine andere Instanz

Rest: 34

Rest: 3

Eine andere Instanz

Rest: -3

Rest: 3

Eine andere Instanz

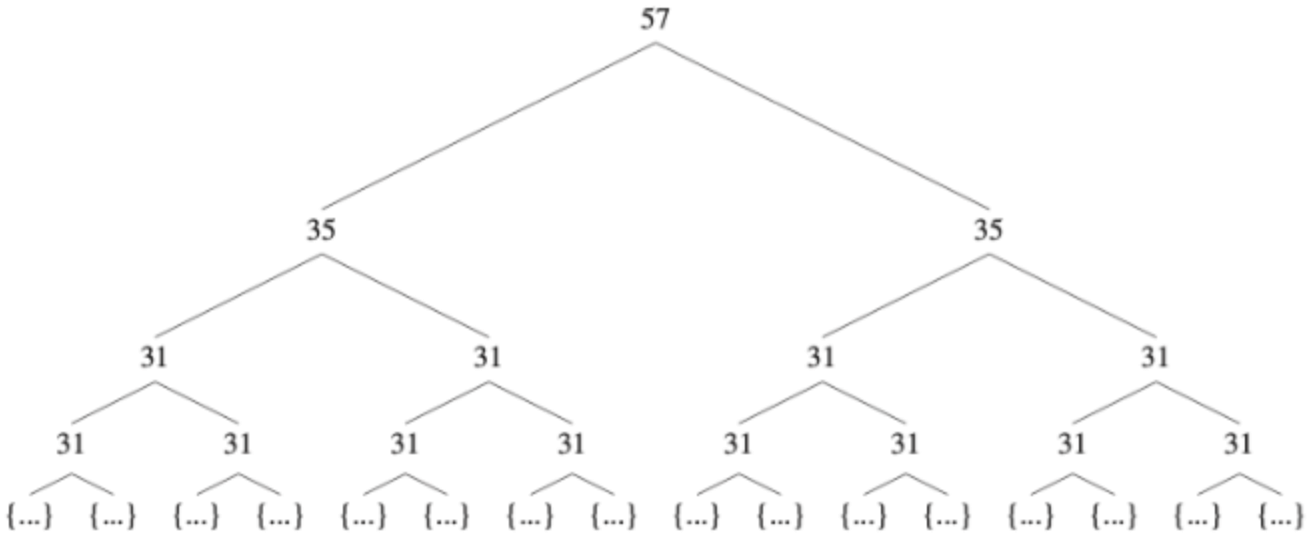
Rest: -3

Keine Lösung möglich?!

Rest: 3

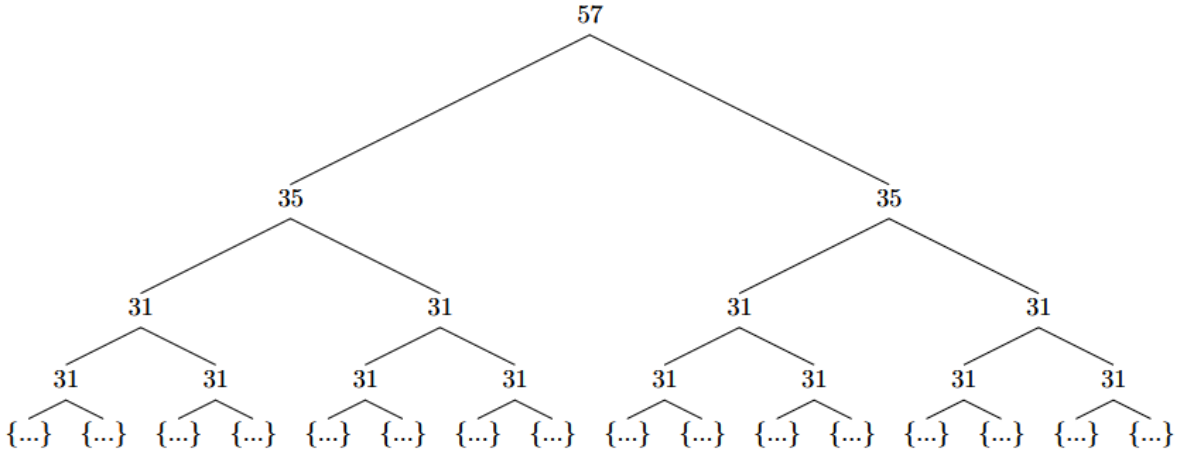
Enumerationsprinzip

120

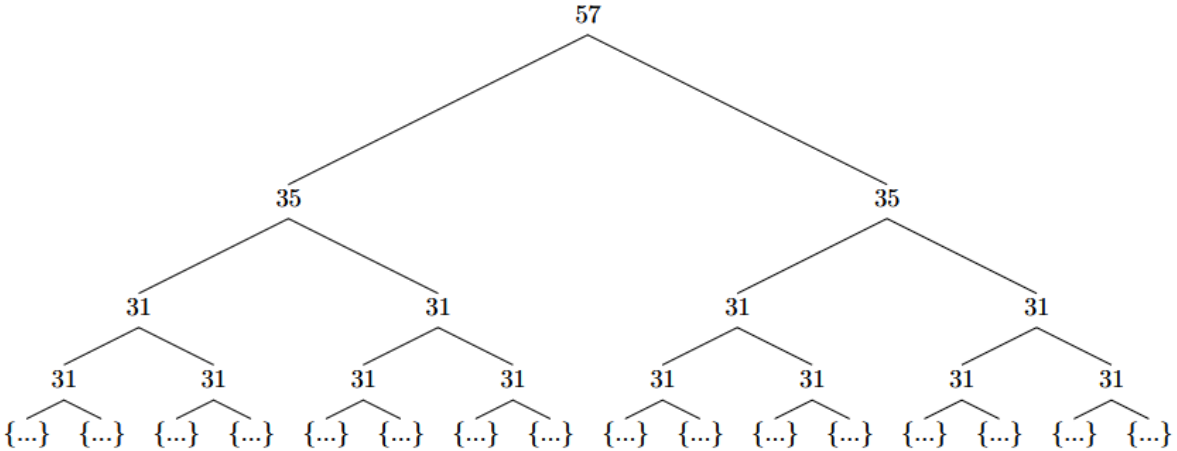


120

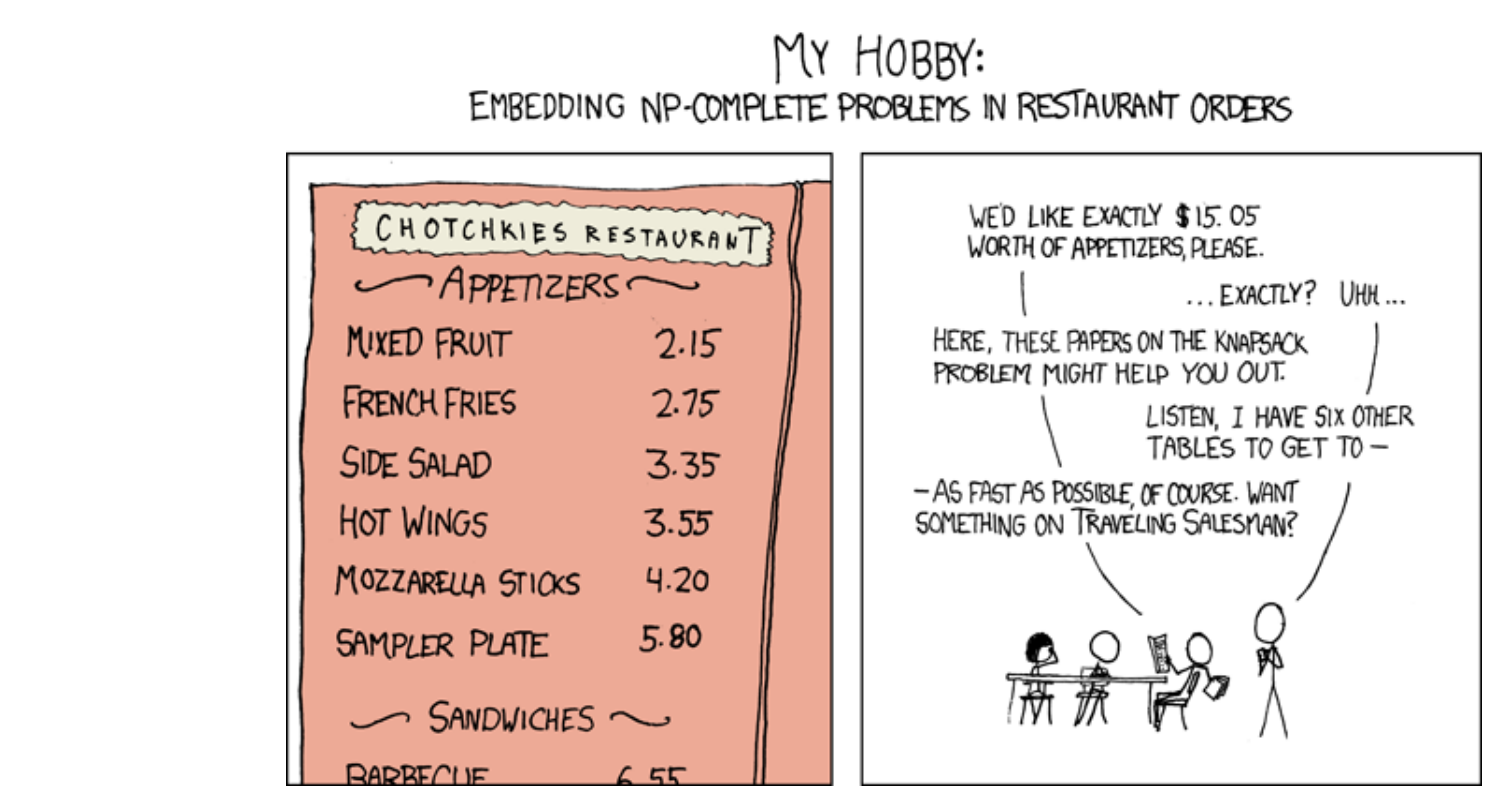
Enumerationsbaum



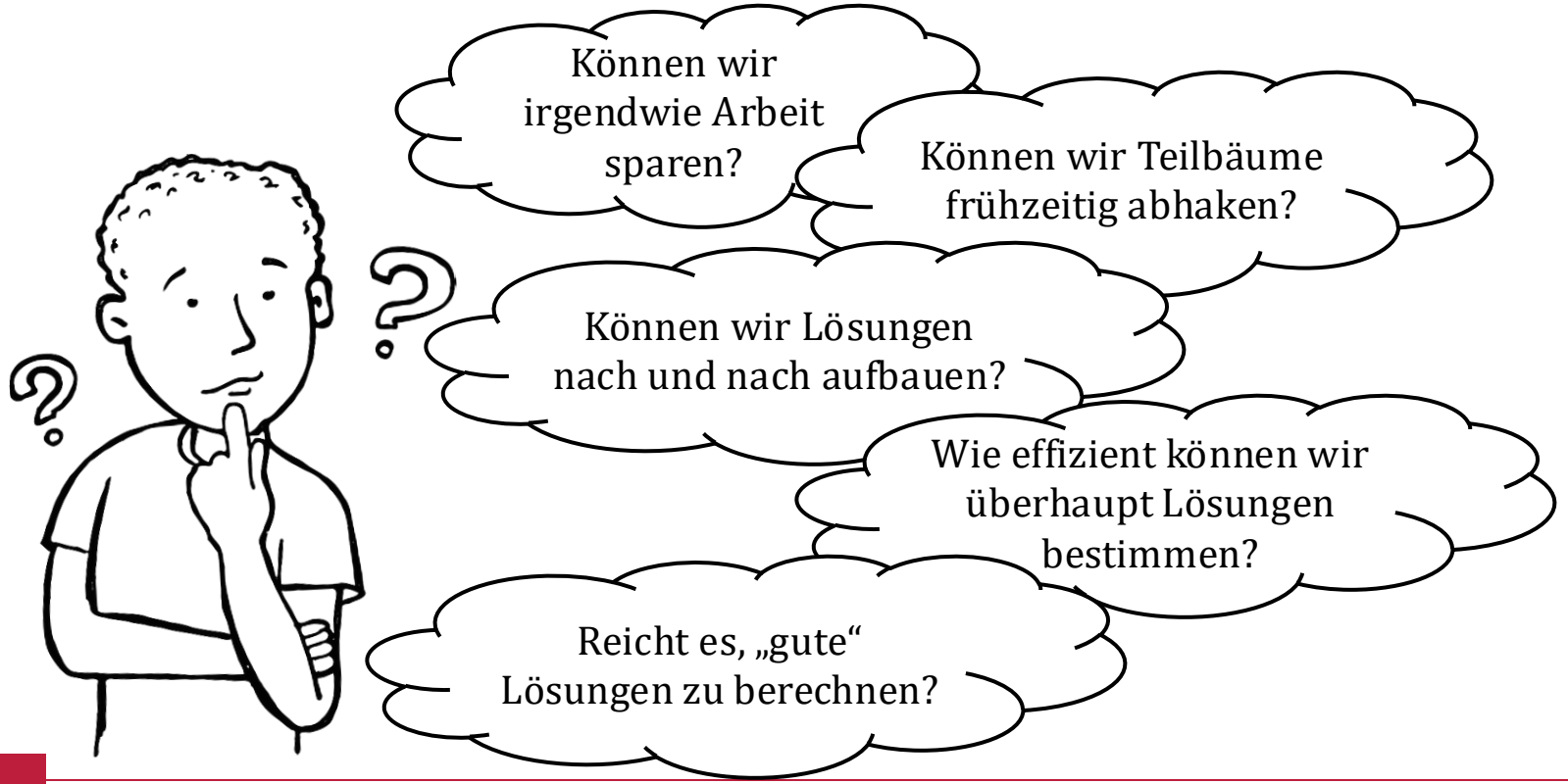
Enumerationsbaum

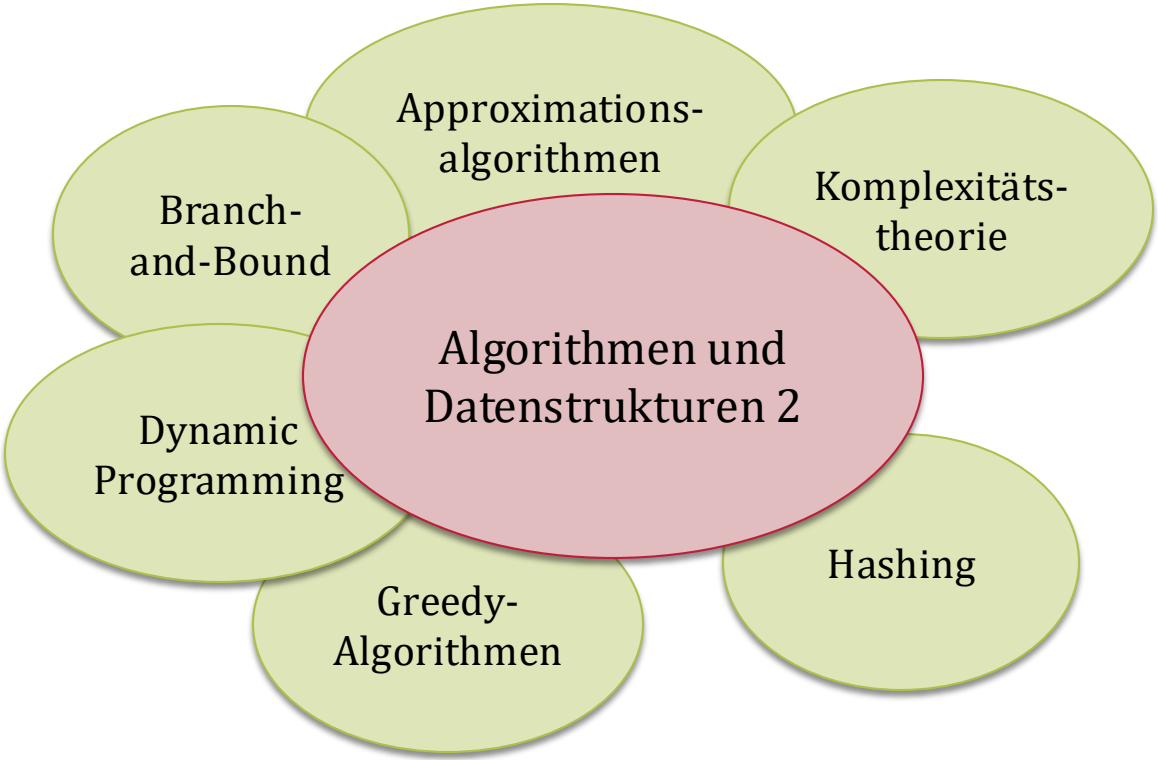


Ein Besuch im Restaurant (xkcd #287)



Überlegungen




Approximations-
algorithmen
Branch-
and-Bound
Komplexitäts-
theorie
Algorithmen und
Datenstrukturen 2
Dynamic
Programming
Hashing
Greedy-
Algorithmen