VL 4

14

Verbesserungen:

- Schneide Teilbäume ab wenn möglich
  ↳ z.T. Einsparungen, bleibt aber i.d.R. exponentiell

  Später: "Branch and Bound"

- Betrachte nicht Rückwärtsrekursion
  (d.h. Probiere stufenweise aus, reduziere dabei größere Teilinstanzen auf kleinere Teilinstanzen)
  sondern Vorwärtsrekursion:
  Baue größere Lösungen aus kleineren Lösungen auf!

Tabelle:

i \ x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...	240
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
1	1	0	0	0	0	0	0	1	0	...													0
2	1							1						1						1			0
3	...																						

Update-Regel:

(1) $\varphi(x,0) = 0$ für alle $x \in \{1,\dots,Z\}$
$\varphi(0,0) = 1$
} Initialisierung

(2) $\varphi(x, i-1) = 1 \Rightarrow \varphi(x, i) = 1$ $\Leftarrow$ Geht vorher, geht immer noch (ohne z_i)

$\varphi(x - z_i, i-1) = 1 \Rightarrow \varphi(x, i) = 1$ $\Leftarrow$ Geht mit z_i weil $x - z_i$ mit $\{1,\dots,i-1\}$ geht

ALGORITHMUS 1.11 (Dynamic Programming für Subset Sum)

Input: $z_1, \ldots, z_n$ mit $\sum_{i=1}^{n} z_i = z$

Output: Boolsche Funktion

$$\varphi : \{1, \ldots, z\} \times \{1, \ldots, n\} \to \{0, 1\}$$
$$(x, i) \mapsto \varphi((x, i))$$

mit $$\varphi((x,i)) = \begin{cases} 1 & \text{wenn } x \text{ aus } z_1, \ldots, z_i \text{ erzeugbar} \\ 0 & \text{sonst} \end{cases}$$

(1) $\varphi(0,0) = 1$;
FOR $(x=1)$ TO z
$\varphi(x,0) = 0$;

(2) FOR $(i=1)$ TO n {
FOR $(x=0)$ TO z {
IF $\left(\left(\varphi(x, i-1) = 1 \right) \text{ OR } \left(\varphi((x - z_i), i-1) = 1 \right) \right)$
$\varphi(x,i) = 1$
ELSE
$\varphi(x,i) = 0$
}
}

(3) RETURN

Satz 1.12

Algorithmus 1.11 löst das Problem Subset Sum.
Die Laufzeit ist $O(z \cdot n)$.

Beweis:

Vollständige Induktion über n!

$n=0$ ist korrekt initialisiert.

Für den Induktionsschritt nehmen wir an, dass $\varphi(x, i-1)$ für alle x korrekt ist.

Dann betrachten wir im Schritt (2) die Möglichkeiten für die Erzeugung von x mit $z_1, \dots, z_i$:

(I) z_i wird nicht verwendet
$\Rightarrow$ x kann mit $z_1, \dots, z_{i-1}$ erzeugt werden.

(II) z_i wird verwendet
$\Rightarrow$ $x - z_i$ kann mit $z_1, \dots, z_{i-1}$ erzeugt werden.

Genau das macht der Algorithmus in (2).

Daher gilt auch, dass $\varphi(x, i)$ korrekt ist.

□