

Verbesserungen:

- Schneide Teilbäume ab wenn möglich  
 ↳ z.T. Einsparungen, bleibt aber <sup>idR.</sup> exponentiell

Später: "Branch and Bound"

- Betrachte nicht Rückwärtsrekursion  
 (d.h. probiere stufenweise aus, reduziere dabei größere Teilinstanzen auf kleinere Teilinstanzen)

Sondern Vorwärtsrekursion:

Bau größere Lösungen aus kleineren Lösungen auf!

# 1.3 Dynamisches Programmieren

## Prinzip: 1.3.1 Subset Sum

7, 13, 17, 20, 29, 31, 31, 35, 57

→  
aufsteigend!

Teste für Zahl  $x$ , ob sie mit den ersten  $i$  Summanden erzeugt werden kann!

$$S(x, i) := \begin{cases} 0 & \text{falls } x \text{ nicht mit } z_1, \dots, z_i \text{ erzeugt werden kann} \\ 1 & \text{falls } x \text{ erzeugt werden kann} \end{cases}$$

$$S(x, 0) = 0 \quad \text{für alle } x > 0 \quad (\text{aber } 1 \text{ für } x = 0)$$

$$S(x, 1) = \begin{cases} 1 & \text{für } x = 7 \\ 0 & \text{sonst} \end{cases}$$

$$S(x, 2) = \begin{cases} 1 & \text{für } x = 7, 13, 20 \\ 0 & \text{sonst} \end{cases}$$

$$S(x, 3) = \begin{cases} 1 & \text{für } x = 7, 13, 20, 17, 24, 30, 37 \\ 0 & \text{sonst} \end{cases}$$

usw.!

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	1	0	...														0
2	1						1						1								1		0
3	...																						

Update-Regel:

- (1)  $\mathcal{F}(x, 0) = 0$  für alle  $x \in \{1, \dots, Z\}$   
 $\mathcal{F}(0, 0) = 1$ 
) Initialisierung
- (2)  $\mathcal{F}(x, i-1) = 1 \Rightarrow \mathcal{F}(x, i) = 1$ 
← geht vorher, geht immer noch (ohne  $z_i$ )
- $\mathcal{F}(x-z_i, i-1) = 1 \Rightarrow \mathcal{F}(x, i) = 1$ 
← geht mit  $z_i$  weit  $x-z_i$  mit  $\{1, \dots, i-1\}$  geht

ALGORITHMUS 1.11 (Dynamic Programming für Subset Sum)

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

Output: Boolesche Funktion

$$\begin{aligned}
 \mathcal{G}: \{1, \dots, Z\} \times \{1, \dots, n\} &\rightarrow \{0, 1\} \\
 (x, i) &\mapsto \mathcal{G}(x, i)
 \end{aligned}$$

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

①  $\mathcal{G}(0, 0) = 1$  ;  
 FOR  $(x=1)$  TO  $Z$   
 $\mathcal{G}(x, 0) = 0$  ;

② FOR  $(i=1)$  TO  $n$  {  
 FOR  $(x=0)$  TO  $Z$  {  
 IF  $((\mathcal{G}(x, i-1) = 1) \text{ OR } (\mathcal{G}(x-z_i, i-1) = 1))$   
 $\mathcal{G}(x, i) = 1$   
 ELSE  
 $\mathcal{G}(x, i) = 0$   
 }  
 }

③ RETURN

SATZ 1.12

ALGORITHMUS 1.11 löst das Problem SUBSET SUM.  
Die Laufzeit ist  $O(2 \cdot n)$ .

Beweis:

Vollständige Induktion über  $n$ !

$n=0$  ist korrekt initialisiert.

Für den Induktionsschritt nehmen wir an, dass

$\mathcal{S}(x, i-1)$  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  $\mathcal{S}(x, i)$  korrekt ist.

