$$P(x,i) = \begin{cases} P(x,i-1), \\ max\{P(x,i- \end{cases}$$

i \ x	0	1	2	3	3	5
0	1	0	0	0	0	0
1	1	0	0	0	0	0
2	1	0	0	0	+13	
3						

2 Dynamic Programming

Algorithmen und Datenstrukturen 2
Sommer 2024

Prof. Dr. Sándor Fekete

Beispiel

Beispiel 1.10.

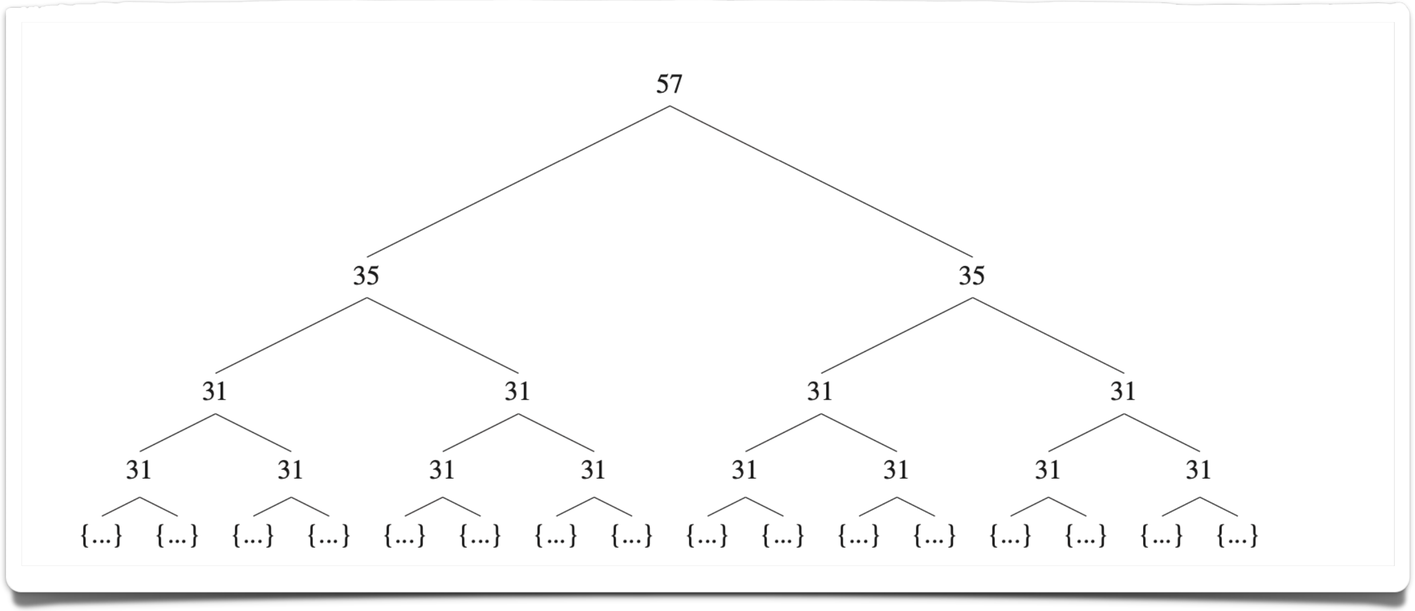
Partition für $\{z_1, ..., z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$

Gesamtsumme: 240

Finde Aufteilung in zwei gleiche Teilmengen!

Enumerationsprinzip

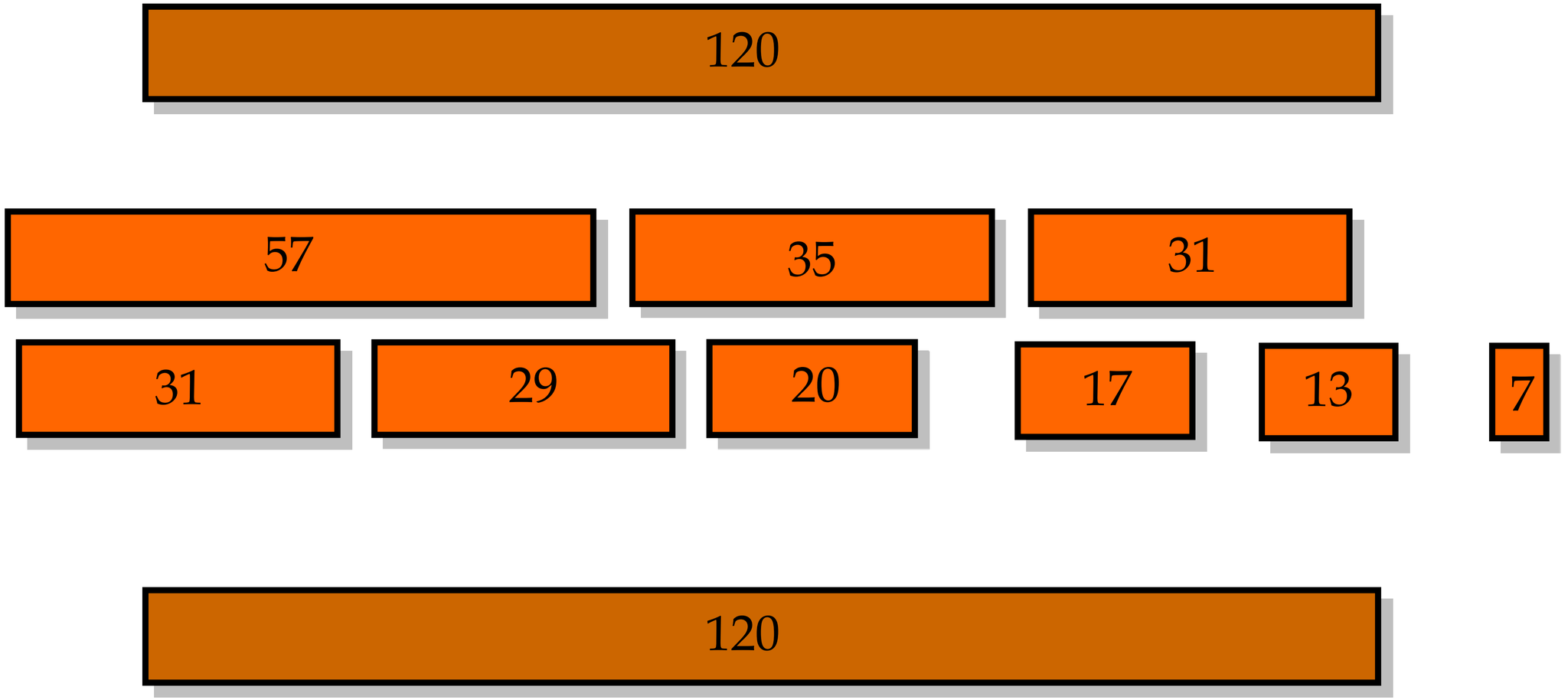
Dynamic Programming



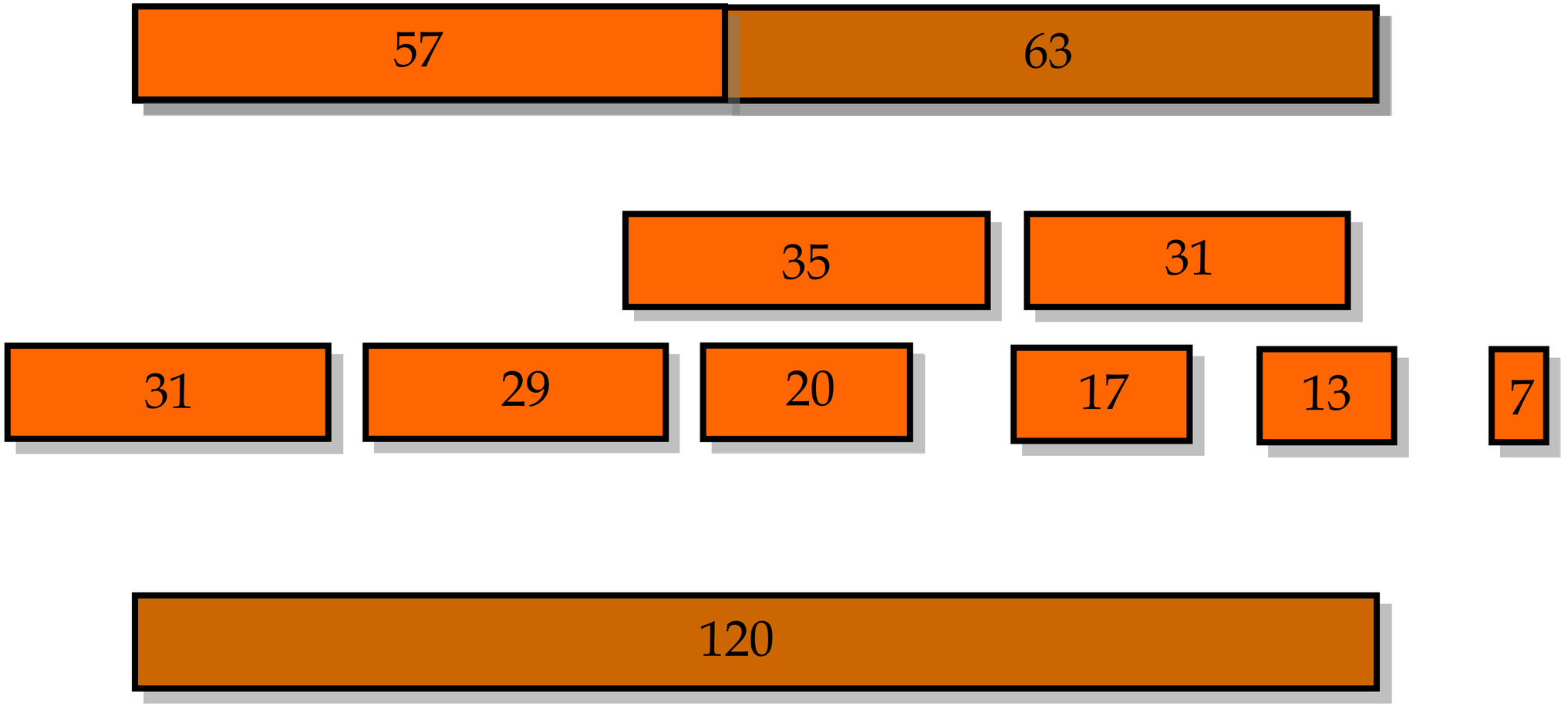
Branch and Bound

- Exponentiell viele Fälle!
- Wie geht das systematisch?
- Wo kann man Arbeit sparen?

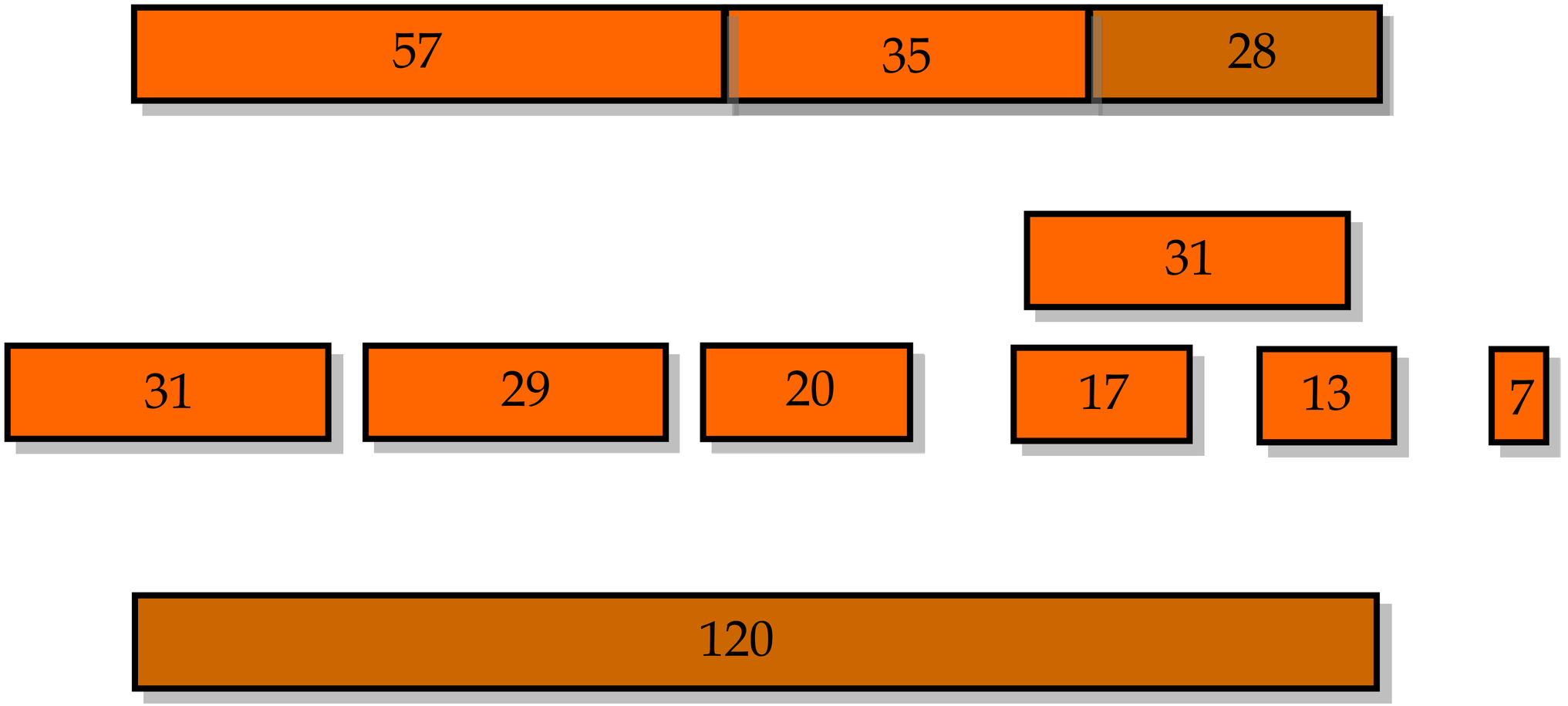
Keine Lösung?!



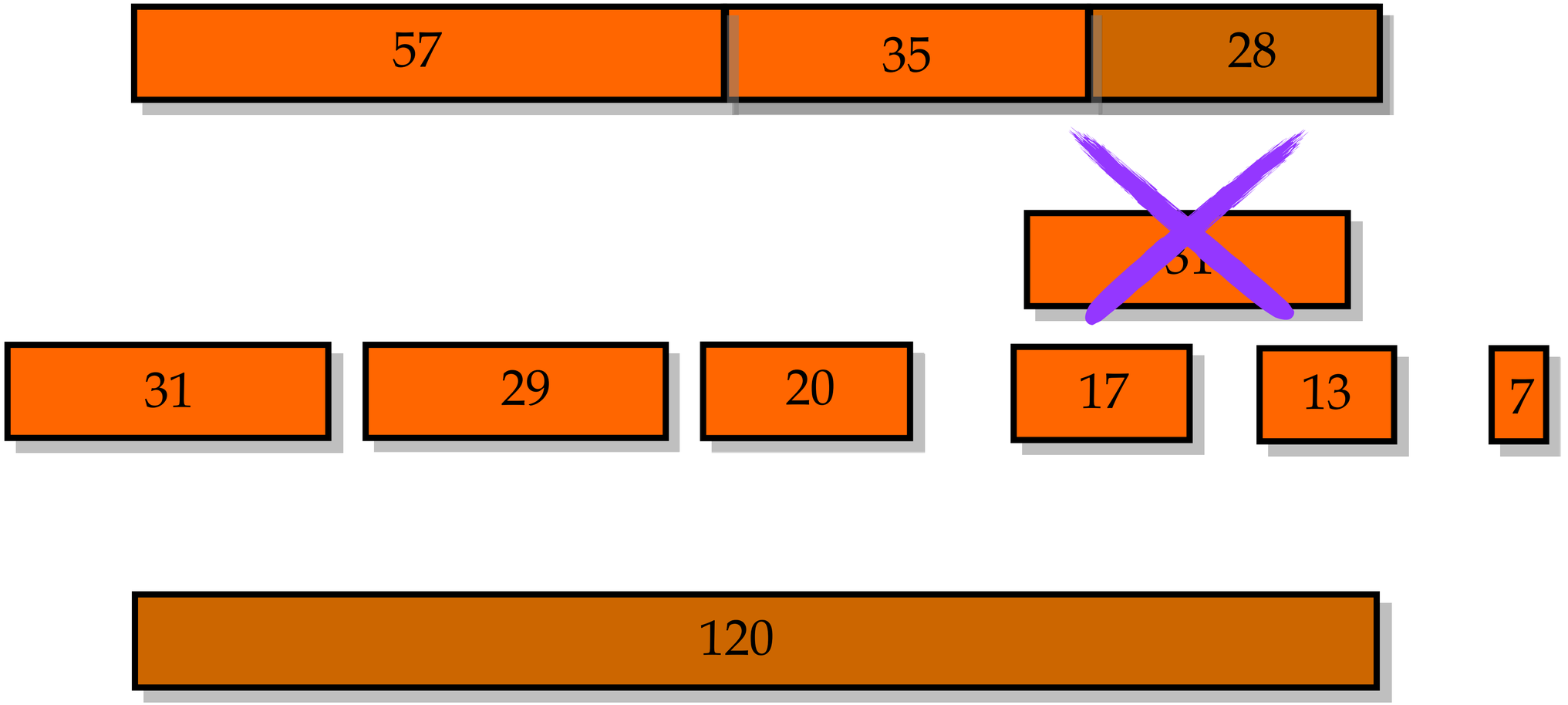
Keine Lösung?!



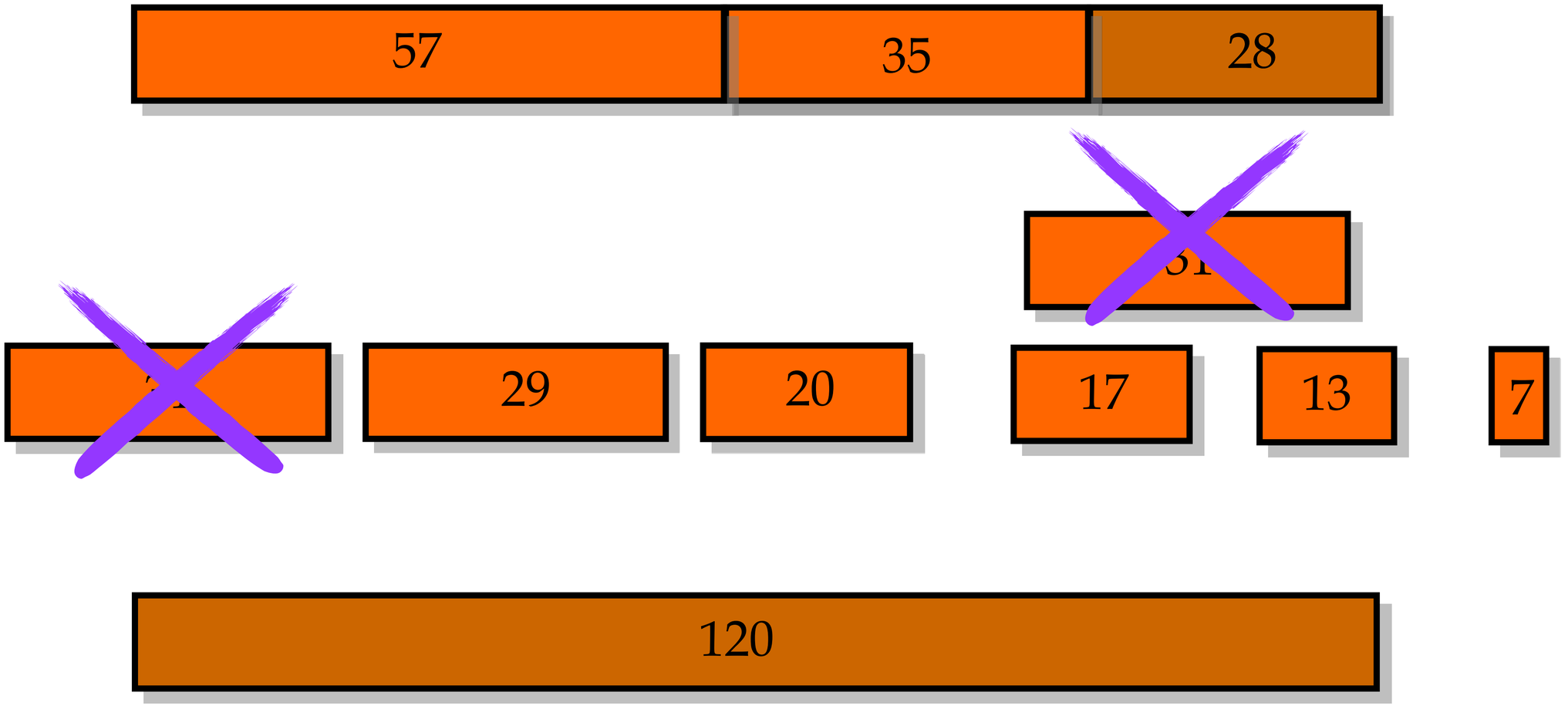
Keine Lösung?!



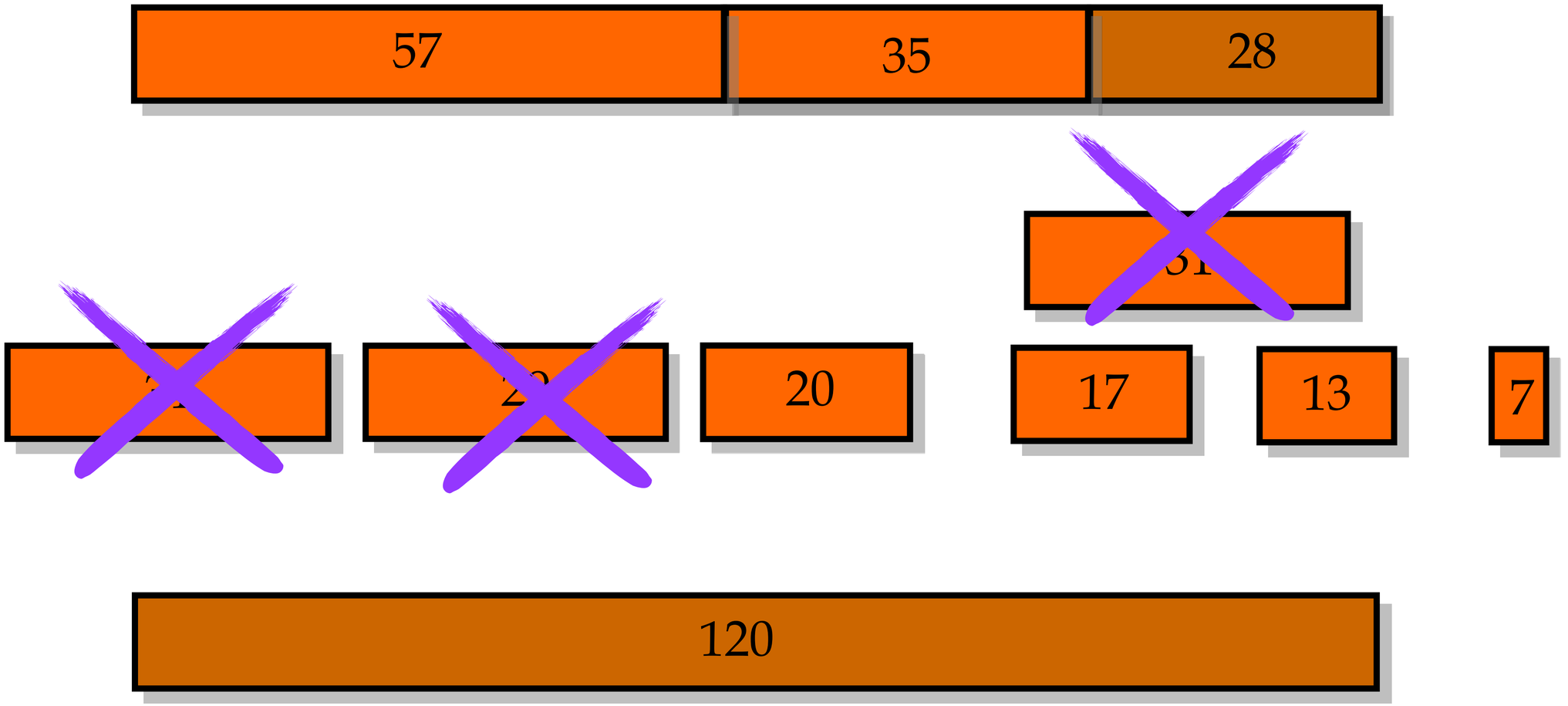
Keine Lösung?!



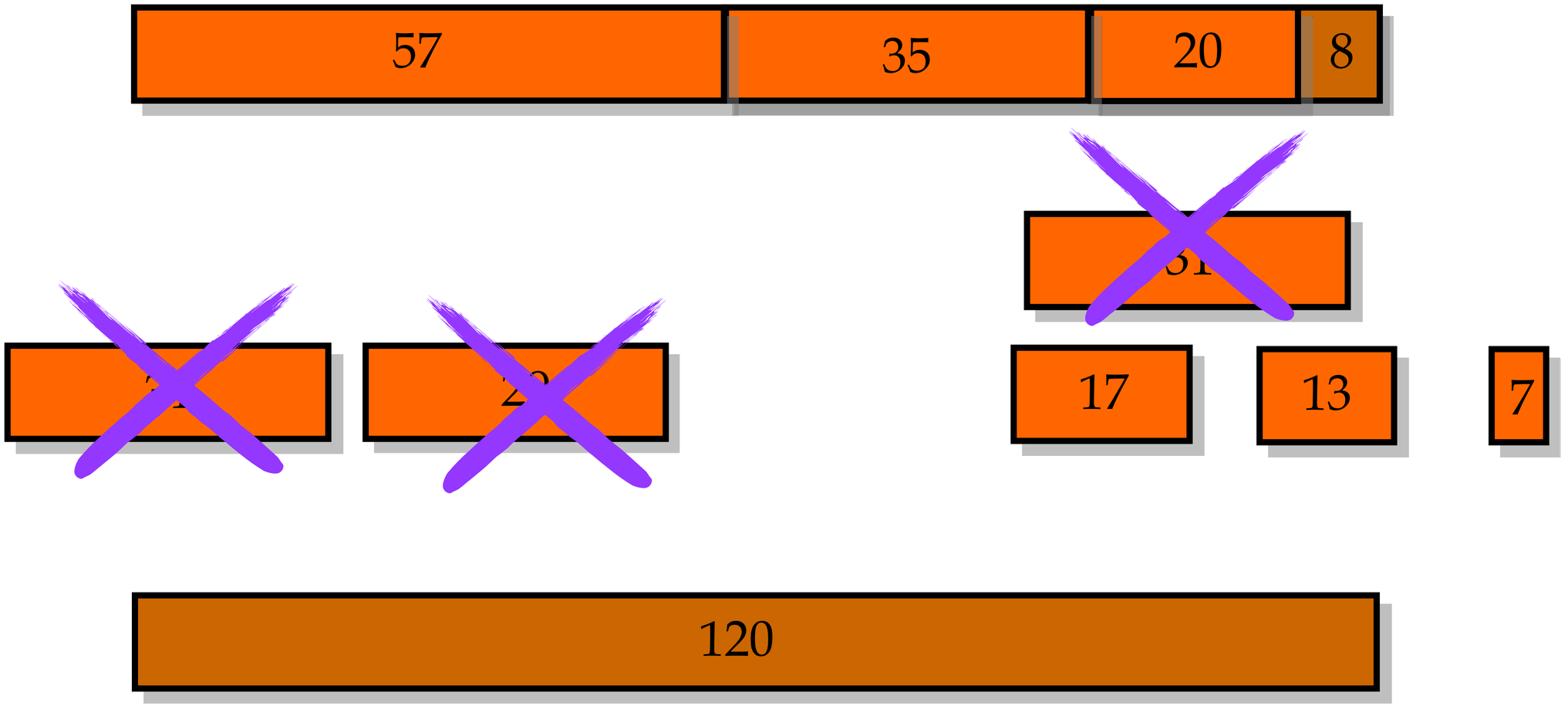
Keine Lösung?!



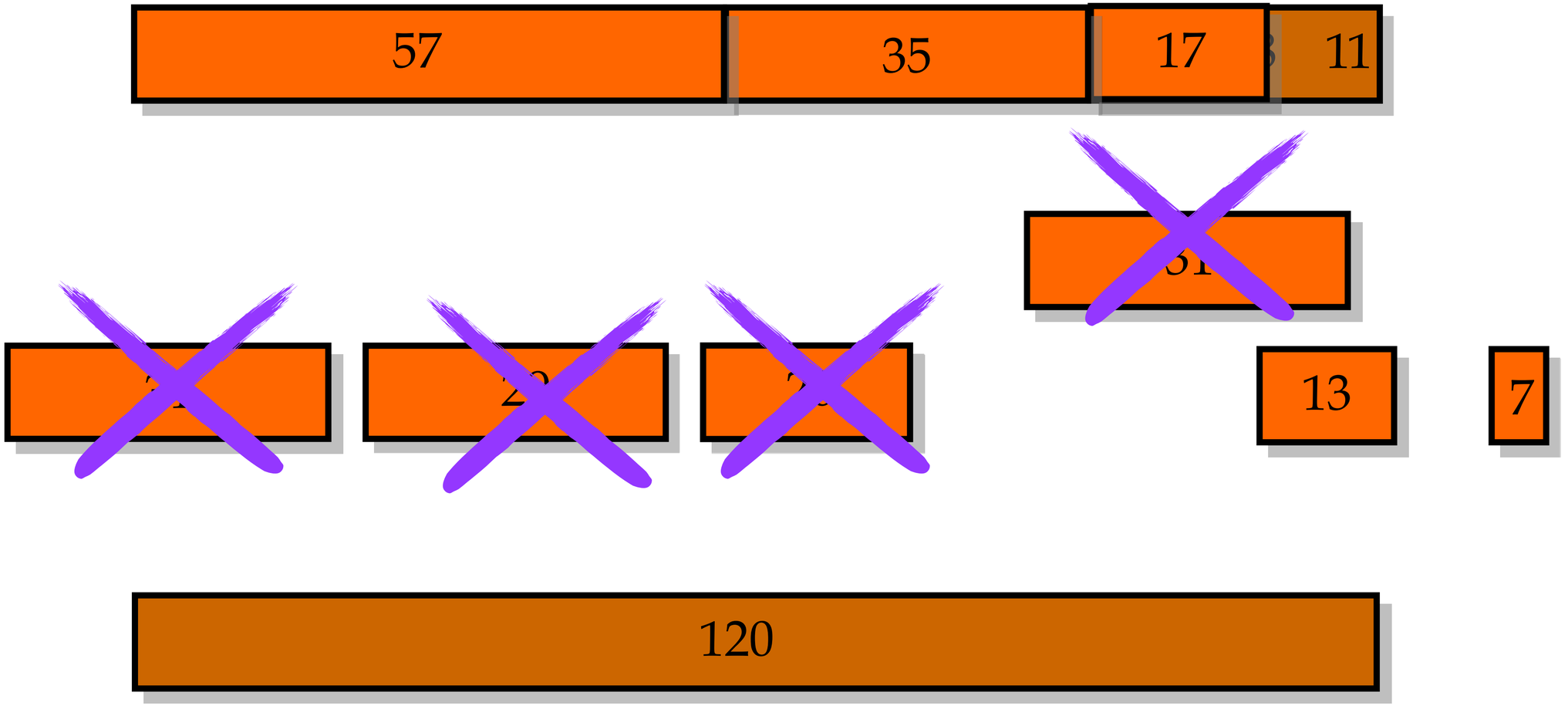
Keine Lösung?!



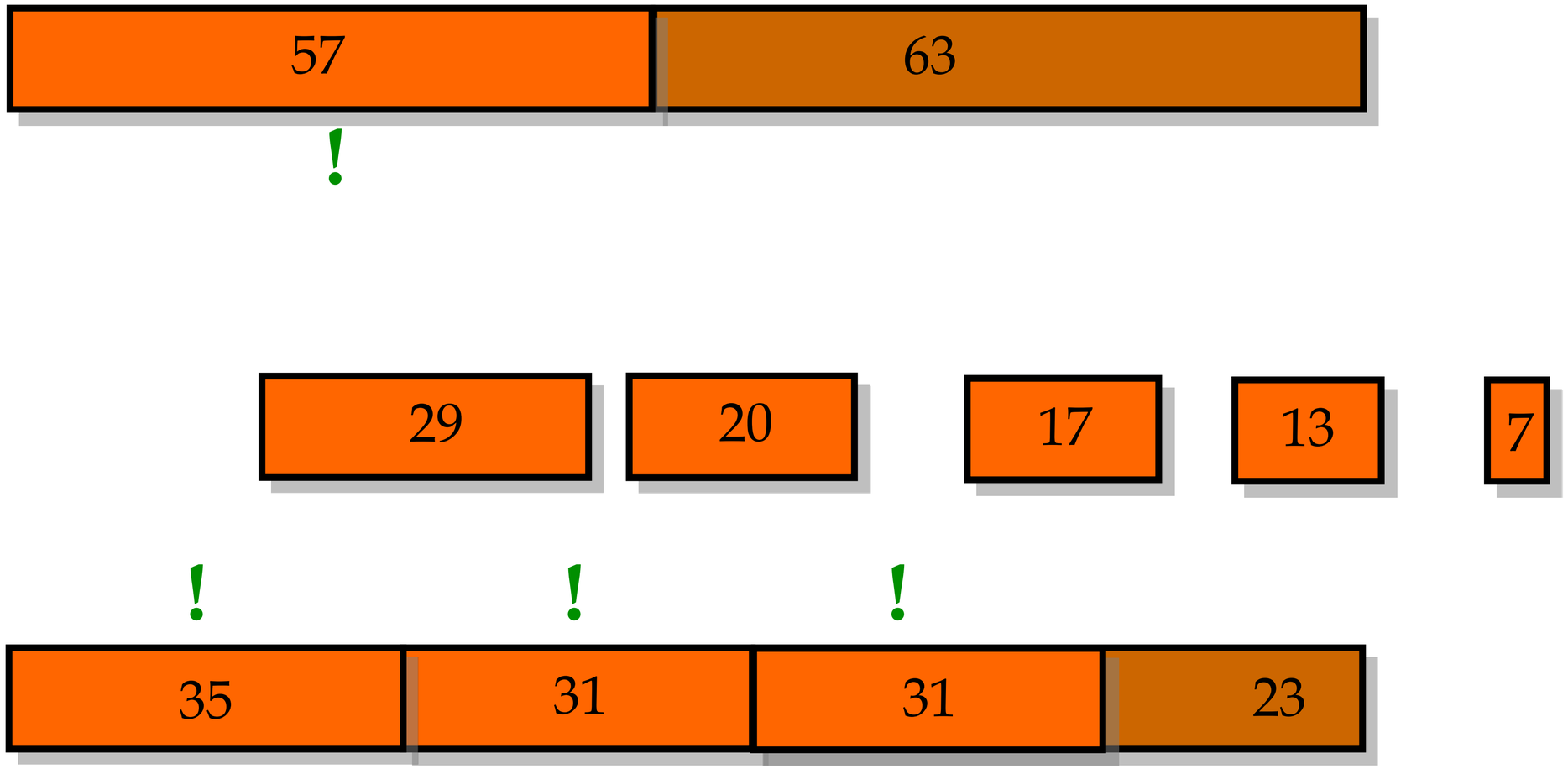
Keine Lösung?!



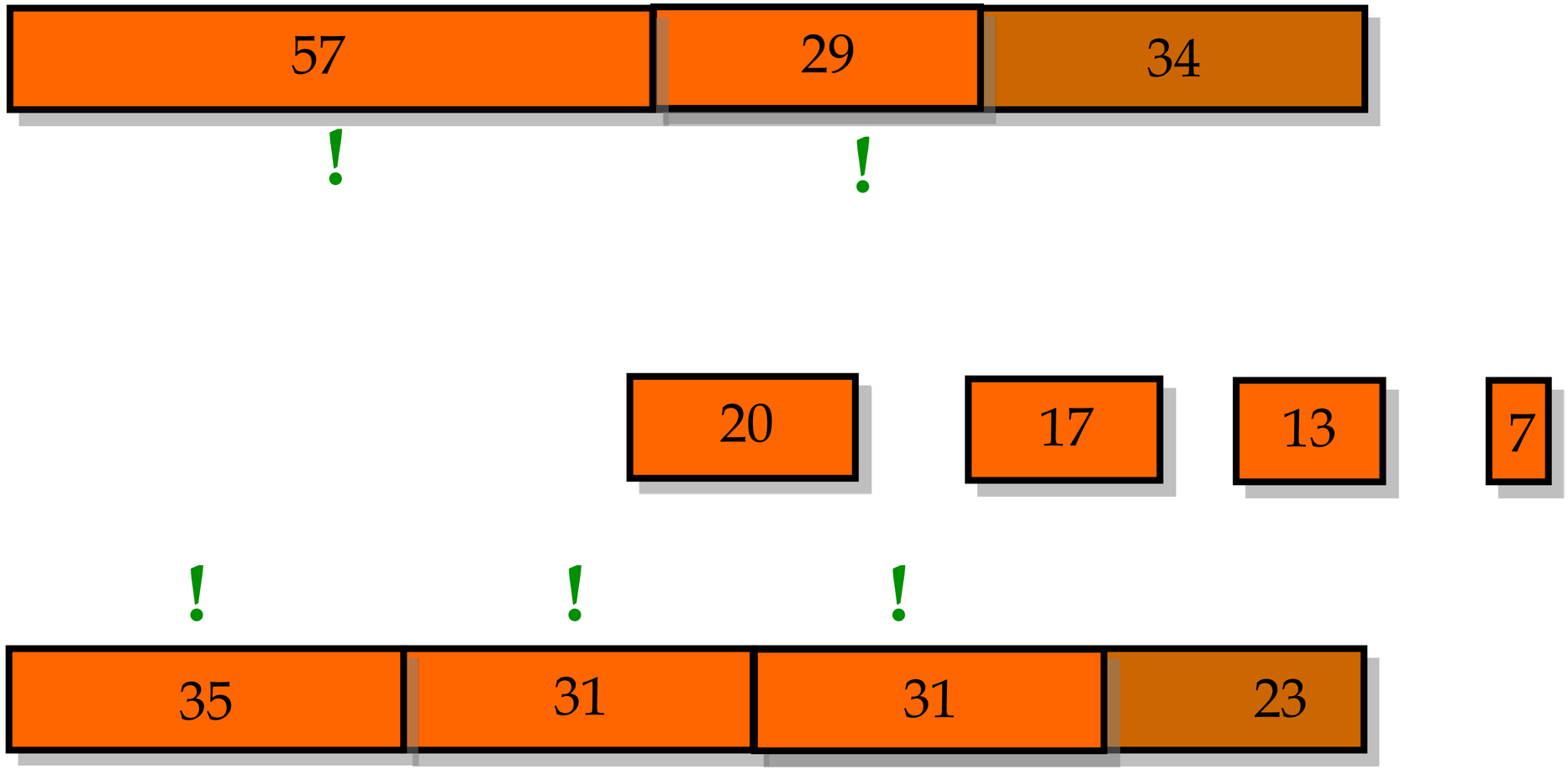
Keine Lösung?!



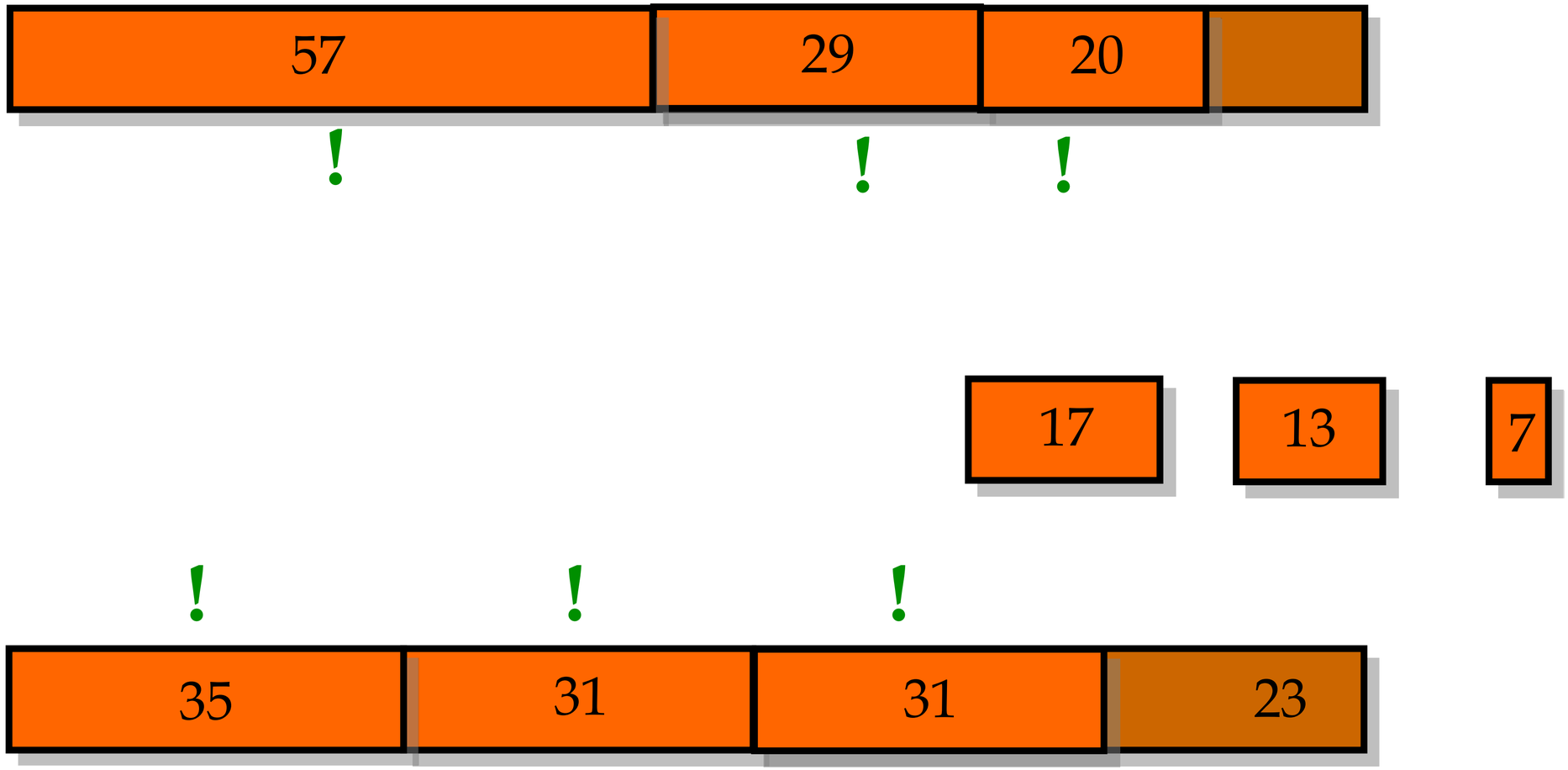
Keine Lösung?!



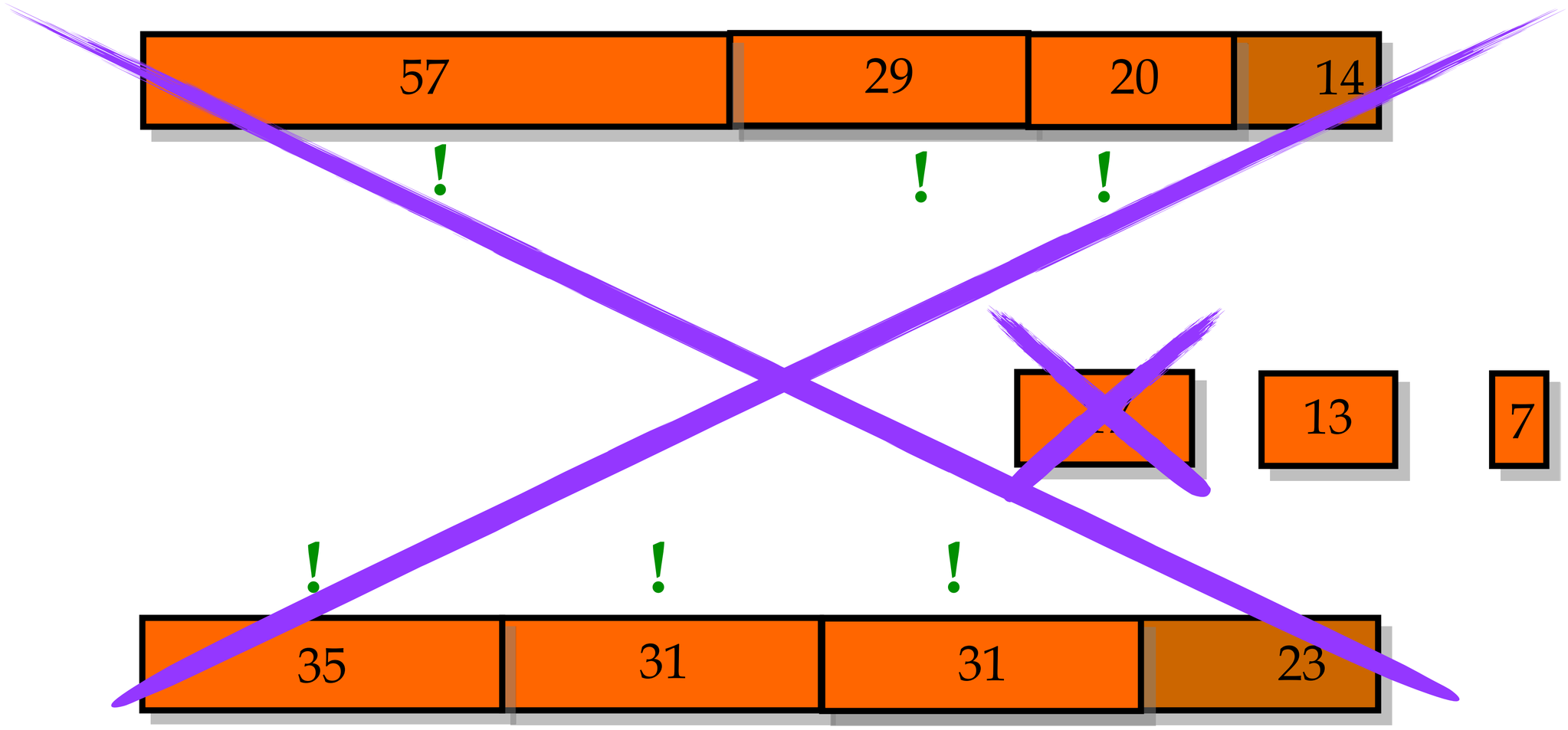
Keine Lösung?!



Keine Lösung?!

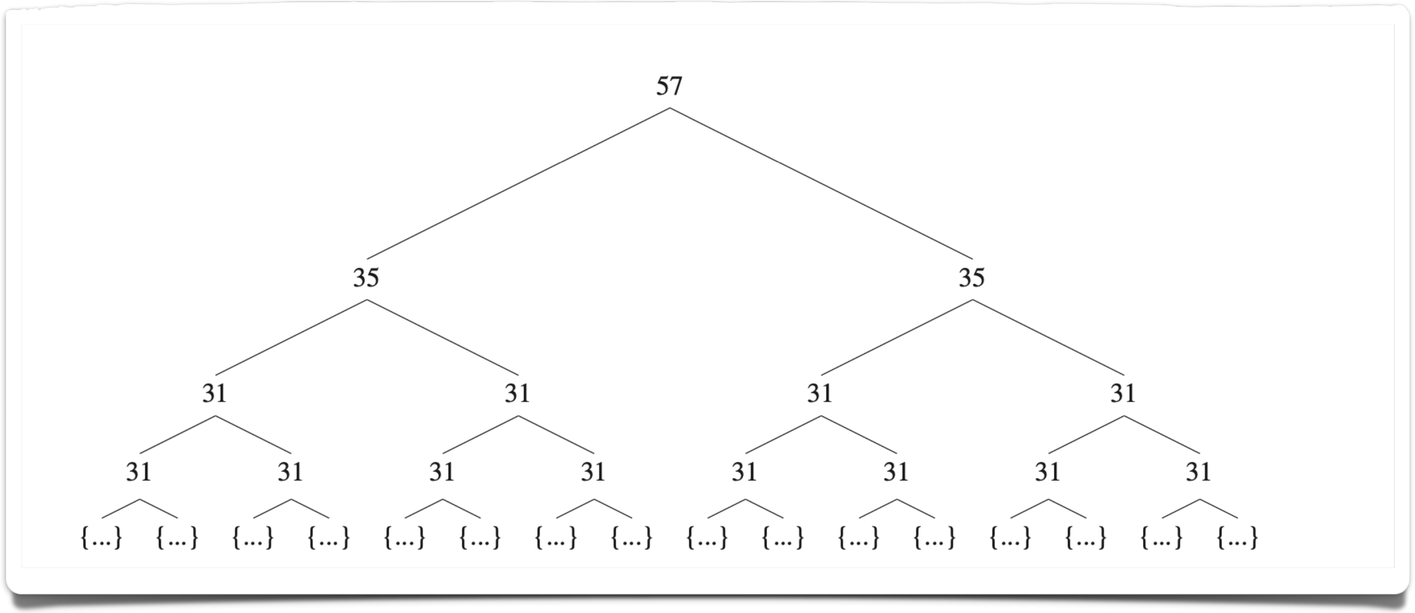


Keine Lösung?!



Ideen

Dynamic Programming



- Bestimme erreichbare Zahlen
- Vorwärts: Nimm Zahlen hinzu

- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: Schließe Zahlen aus

Beispiel

Beispiel 2.1

$$\{z_1, \ldots, z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$$

Bestimme alle Zahlen, die als Teilsummen erreichbar sind!

- Bestimme erreichbare Zahlen
- Vorwärts: Nimm Zahlen hinzu

Beispiel

$$\{z_1, ..., z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$$

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

i \ x	0	1	2	3	3	5	6	7	…	13	14	15	16	17	18	19	20	…	240
0																			
1																			
2																			
3																			

Beispiel

$$\{z_1, ..., z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$$

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

i \ x	0	1	2	3	3	5	6	7	…	13	14	15	16	17	18	19	20	…	240
0																			
1																			
2																			
3																			

Beispiel

$$\{z_1, \ldots, z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$$

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

i \ x	0	1	2	3	3	5	6	7	…	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
1																			
2																			
3																			

Beispiel

$$\{z_1, \ldots, z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$$

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

i \ x	0	1	2	3	3	5	6	7	…	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
1	1	0	0	0	0	0	0	1	…	0	0	0	0	0	0	0	0	…	0
2																			
3																			

+7

Beispiel

$$\{z_1, \ldots, z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$$

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

i \ x	0	1	2	3	3	5	6	7	…	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
1	1	0	0	0	0	0	0	1	…	0	0	0	+13	0	0	0	0	…	0
2	1	0	0	0	+13	0	1	…	1	0	0	0	0	0	0	1	…	0	
3																			

Beispiel

$$\{z_1, \ldots, z_9\} = \{7, 13, 17, 20, 29, 31, 31, 35, 57\}$$

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

24 30 37

i \ x	0	1	2	3	3	5	6	7	…	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
1	1	0	0	0	0	0	0	1	…	0	0	0	0	0	0	0	0	…	0
2	1	0	0	0	0	0	0	1	…	1	0	0	0	0	0	0	1	…	0
3	1	0	0	0	0	0	0	1	…	1	0	+17	0	1	0	0	1	…	0

Allgemeiner

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

$$\mathcal{S}(x,0) = 0, \text{ für alle } x \in \{1, ..., Z\}; \mathcal{S}(0,0) = 1$$

i \ x	0	1	2	3	3	5	6	7	…	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
1	1	0	0	0	0	0	0	1	…	0	0	0	+13	0	0	0	0	…	0
2	1	0	0	0	+13	0	1	…	1	0	0	0	0	0	0	1	…	0	
3																			

$$\mathcal{S}(x, i-1) = 1 \Rightarrow \mathcal{S}(x,i) = 1$$

$$\mathcal{S}(x - z_i, i-1) = 1 \Rightarrow \mathcal{S}(x,i) = 1$$

Algorithmus

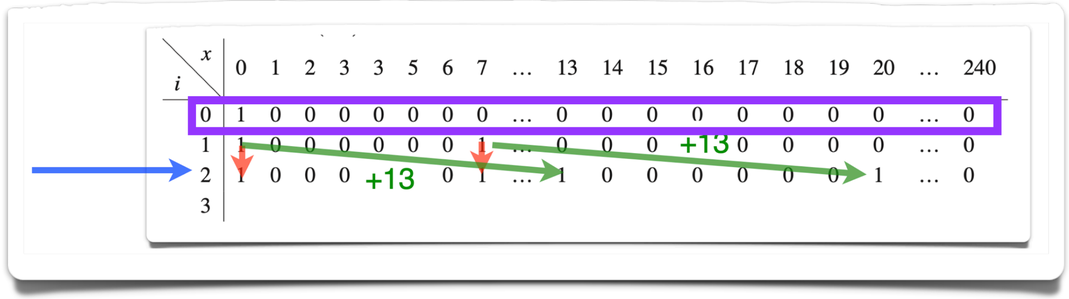
Algorithmus 2.2 . (Dynamic Programming für SUBSET SUM).

Eingabe: *Zahlen* $z_1, ..., z_n$ *mit* $\sum_{i=1}^{n} z_i = Z$

Ausgabe: *Boolesche Funktion* $\mathscr{S} : \{0, ..., Z\} \times \{0, ..., n\} \to \{0, 1\}$

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

```
1:  S(0,0) := 1
2:  for (x=1) to Z do
3:      S(x,0) := 0;
4:  for (i = 1) to n do
5:      for (x = 0) to z_i − 1 do
6:          S(x,i) := S(x,i−1);
7:      for (x = z_i) to Z do
8:          if ( S(x,i−1) = 1 OR (S((x−z_i),i−1) = 1) then
9:              S(x,i) := 1;
10:         else
11:             S(x,i) := 0;
12: return
```



Korrektheit

Satz 2.3 *Algorithmus 2.2 löst das Problem* SUBSET SUM. *Die Laufzeit ist* $O(Z \cdot n)$.

Beweis. Induktion über die Anzahl der Elemente n.

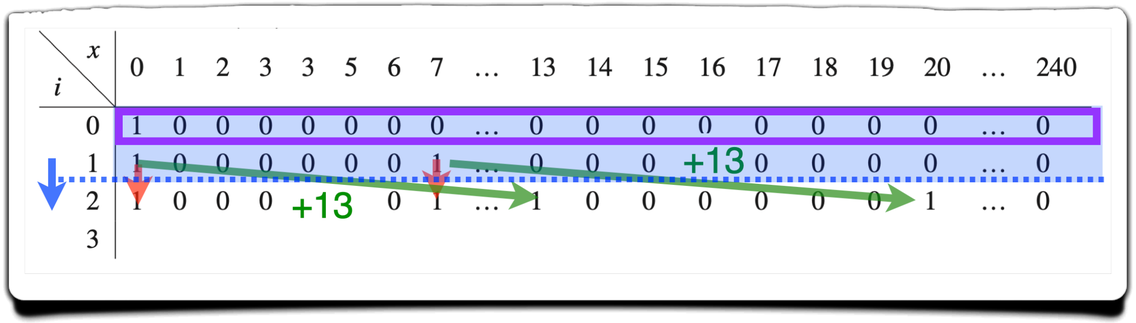
Induktionsanfang: Initialisierung korrekt für $n=0$.

Induktionssannahme: $\mathcal{S}(x, i-1)$ *korrekt für alle i.*

Induktionsschritt: Erzeugung von x mit $z_1, ..., z_i$.

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

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



Laufzeit

Algorithmus 2.2 . (Dynamic Programming für SUBSET SUM).

Eingabe: *Zahlen* $z_1, ..., z_n$ *mit* $\sum_{i=1}^{n} z_i = Z$

Ausgabe: *Boolesche Funktion* $\mathscr{S} : \{0, ..., Z\} \times \{0, ..., n\} \to \{0, 1\}$

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

```
1:  S(0,0) := 1
2:  for (x=1) to Z do
3:      S(x,0) := 0;
4:  for (i = 1) to n do
5:      for (x = 0) to z_i − 1 do
6:          S(x,i) := S(x,i−1);
7:      for (x = z_i) to Z do
8:          if ((S(x,i−1) = 1) OR (S((x−z_i),i−1) = 1)) then
9:              S(x,i) := 1;
10:         else
11:             S(x,i) := 0;
12: return
```

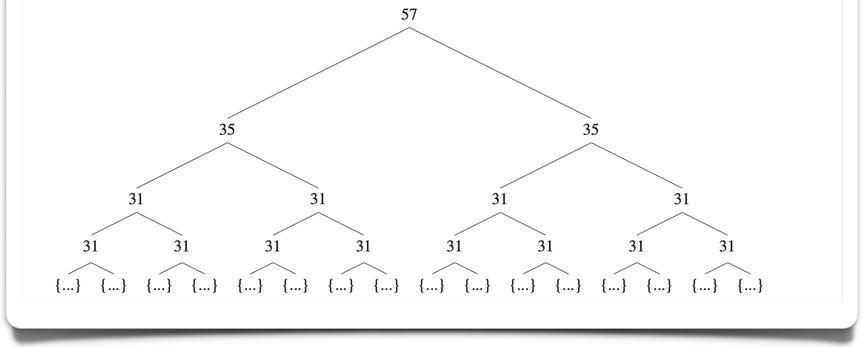
$O(nZ)$

Effizienz?!

$O(nZ)$

i \ x	0	1	2	3	3	5	6	7	…	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
1																			
2																			
3																			

2^n



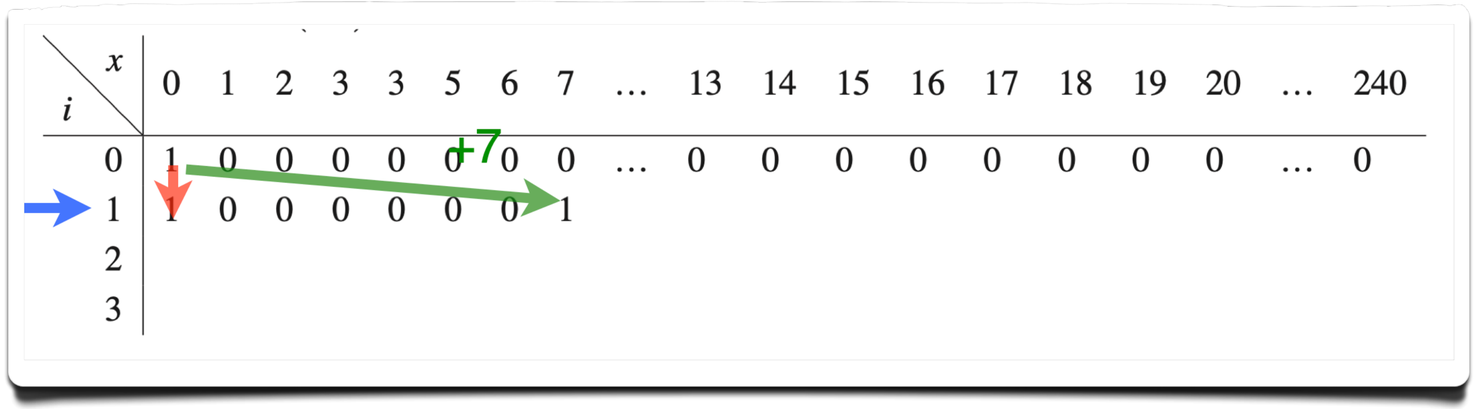
1

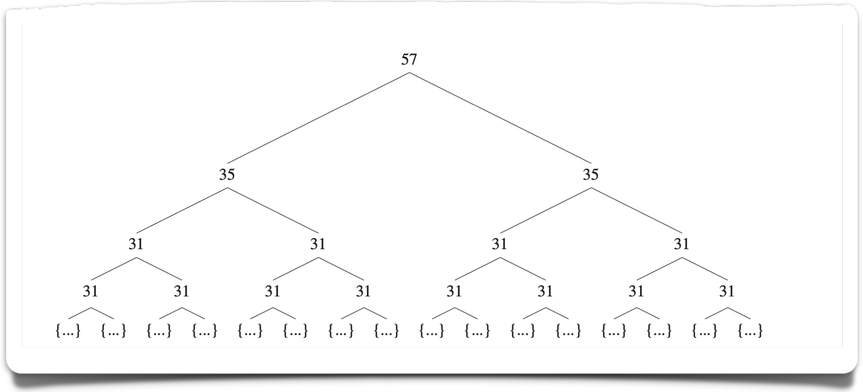
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

$O(nZ)$

2^n





2

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

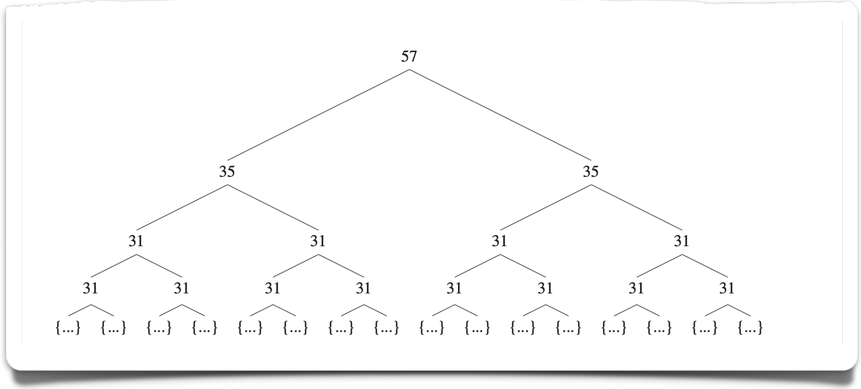
Effizienz?!

$O(nZ)$

i \ x	0	1	2	3	3	5	6	7	…	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
1	1	0	0	0	0	0	0	1	…	0	0	0	+13	0	0	0	0	…	0
2	1	0	0	0	+13	0	1	…	1	0	0	0	0	0	0	1	…	0	
3																			

4

2^n



- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

$O(nZ)$

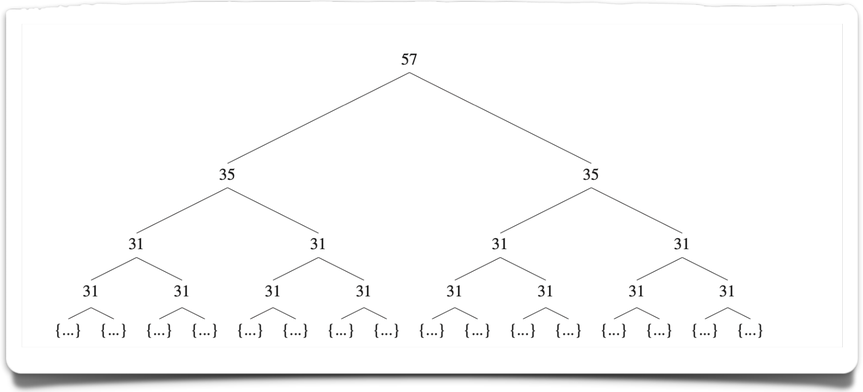
i \ x	0	1	2	3	3	5	6	7	…	13	14	15	16	17	18	19	20	…	240
0	1	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	…	0
1	1	0	0	0	0	0	0	1	…	0	0	0	0	0	0	0	0	…	0
2	1	0	0	0	0	0	0	1	…	1	0	0	0	0	0	0	1	…	0
3	1	0	0	0	0	0	0	1	…	1	0	0	0	1	0	0	1	…	0

24 30 37

+17

8

2^n



- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

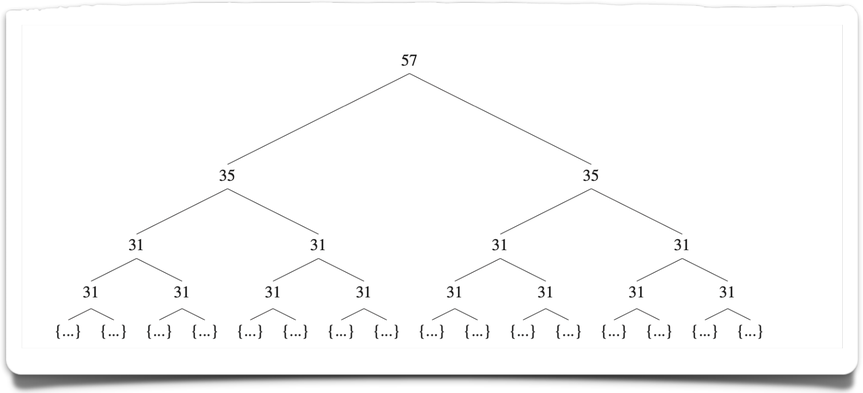
$O(nZ)$

2^n

i \ x	0	1	2	3	3	5	6	7	…	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
1	1	0	0	0	0	0	0	1	…	0	0	0	0	0	0	0	0	…	0
2	1	0	0	0	0	0	0	1	…	1	0	0	0	0	0	0	1	…	0
3	1	0	0	0	0	0	0	1	…	1	0	0	0	1	0	0	1	…	0

24 30 37

+17



16

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

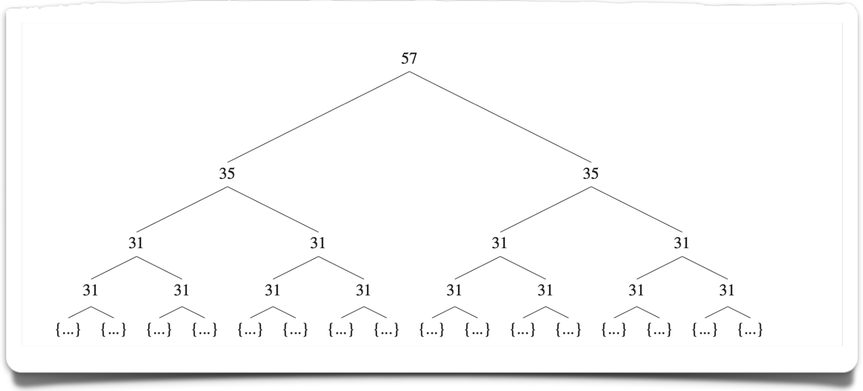
$O(nZ)$

i \ x	0	1	2	3	3	5	6	7	…	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
1	1	0	0	0	0	0	0	1	…	0	0	0	0	0	0	0	0	…	0
2	1	0	0	0	0	0	0	1	…	1	0	0	0	0	0	0	1	…	0
3	1	0	0	0	0	0	0	1	…	1	0	0	0	1	0	0	1	…	0

24 30 37

+17

2^n

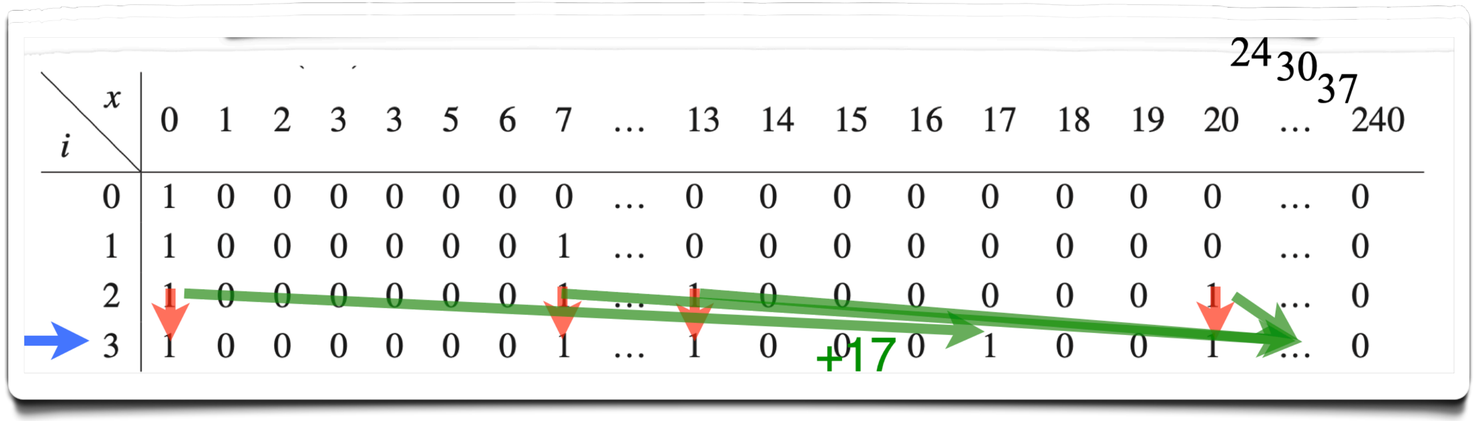


32

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

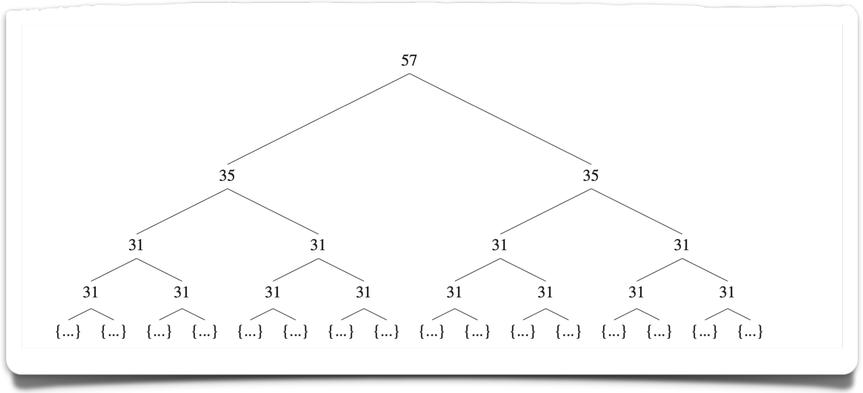
Effizienz?!

$O(nZ)$



64

2^n

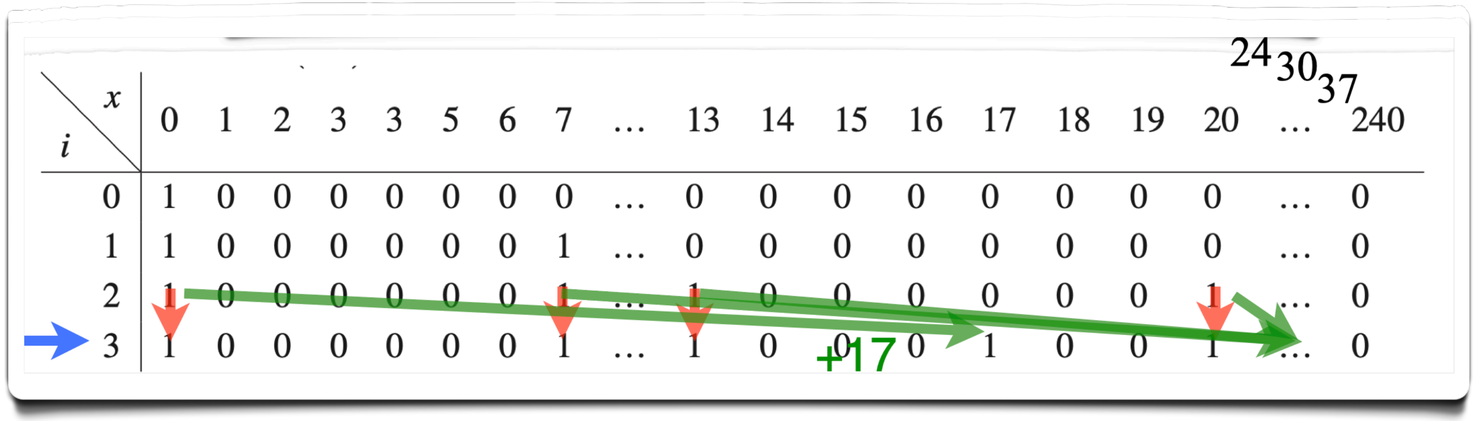


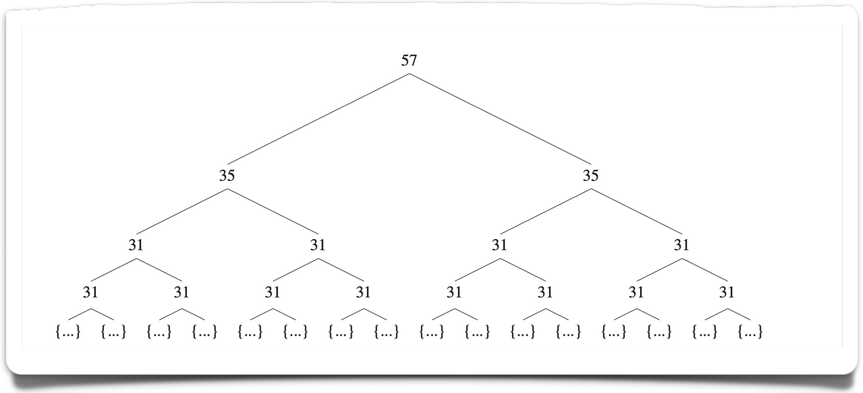
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

$O(nZ)$

2^n





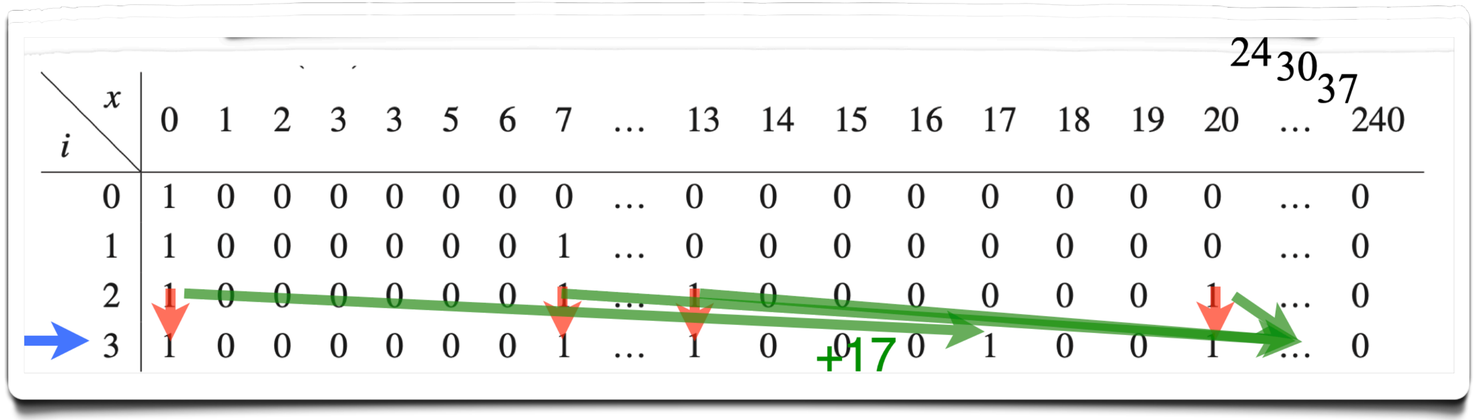
128

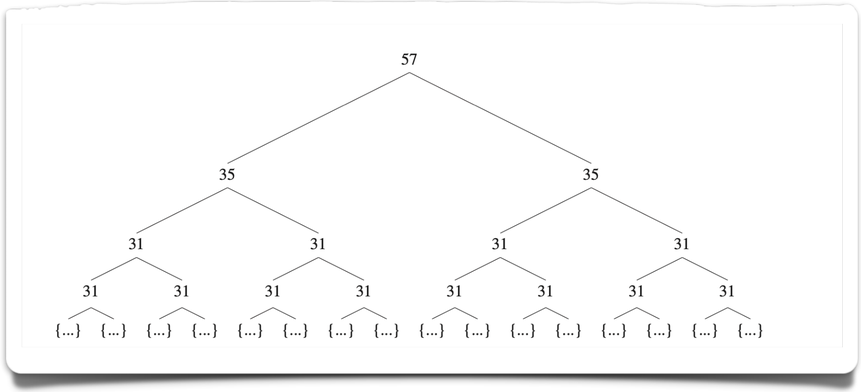
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

$O(nZ)$

2^n





256?!

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Z klein im Vergleich zu 2^n : viele Duplikate gespart

Z groß im Vergleich zu 2^n : viel unnötige Arbeit

Fragen

- Wie lässt sich das Prinzip verallgemeinern?
- Wie geht das für Knapsack?
- Wie geht das für andere Probleme?

Vielen Dank!

s.fekete@tu-bs.de