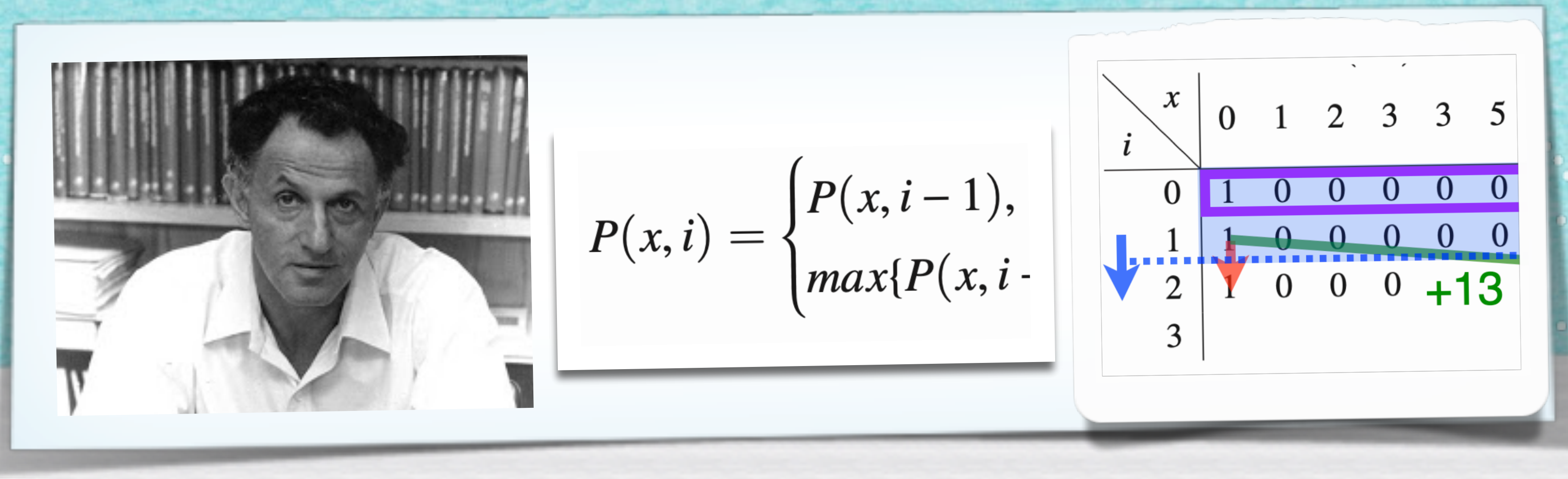


Algorithmen und Datenstrukturen 2

Sommer 2023

Prof. Dr. Sándor Fekete

Studentische Vertretung der Informatik an der TU

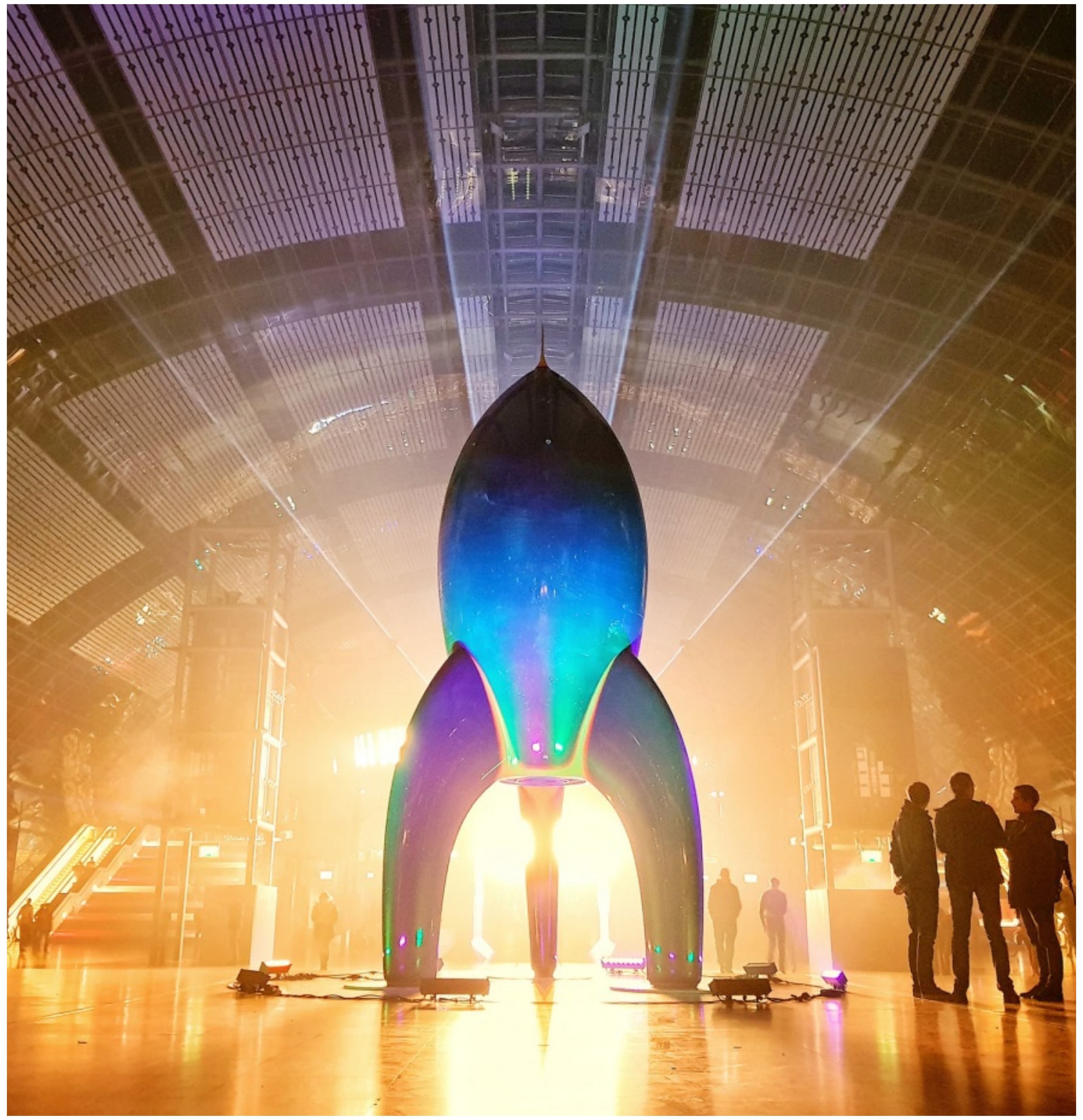
Vielfältige Service-Angebote
- Blog: fginfo.tu-bs.de
- Prüfungs- und Lernmaterialien
- Wiki & Knowledge Base

Socializing
- Fachgruppenraum (IZ 149/150)
- Ersti-Einführung und Betreuung
  - Du willst mithelfen? Einladung zur Orga kommt über cs-studs!
- Spiele-, Grillabende, LAN-Partys
- Exkursionen (z.B. Chaos Communication Congress)

Studentische Vertretung der Informatik an der TU

Hochschulpolitik

- Vertretung der studentischen Interessen in Hochschulgremien
- Mitwirken an Weiterentwicklung und Verbesserung des Studiums
- Ansprechpartner*innen bei Problemen
  - Kommt einfach vorbei!



Kontakt:

- Mail: fginfo@tu-bs.de
- Sprechstunde: Dienstags, 15:45
- FG-Treffen: Dienstags, 16:45

Was machen wir?

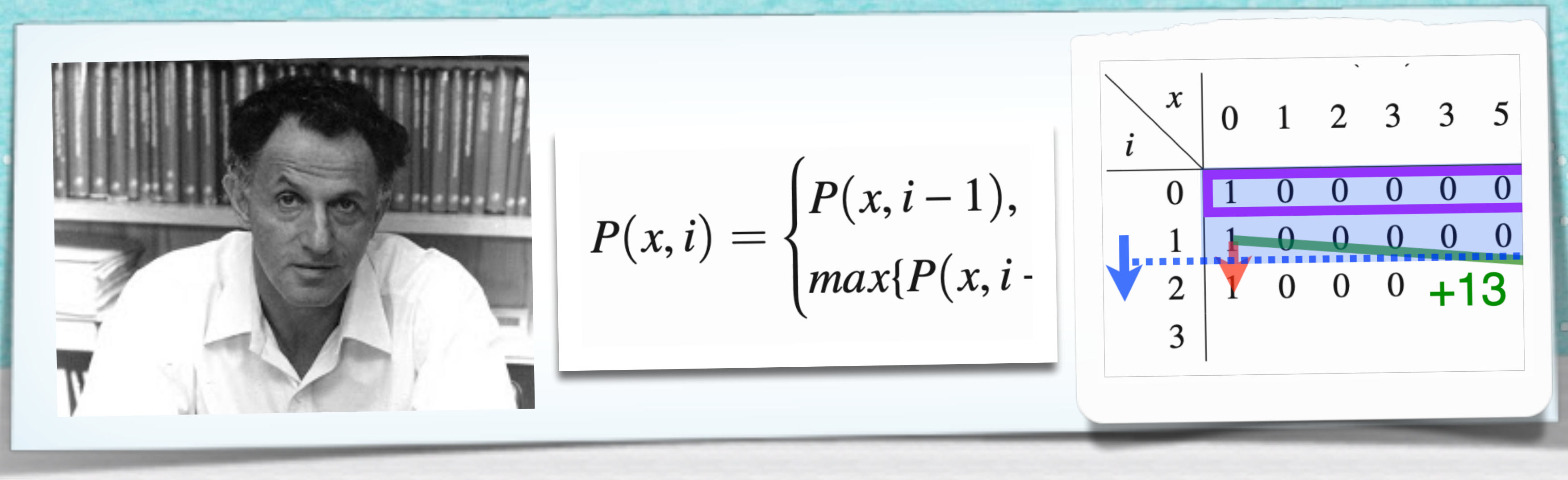
- coole **Veranstaltungen** planen
  - Erstsemester-Woche
  - Grillen, Spieleabende, monatliche Treffen, …
- Studierende **unterstützen** und vertreten
  - Du hast Fragen zu deinem Studiengang, einer Prüfung oder sonstigem? Kontaktiere uns einfach und wir helfen dir!
- Eigene, die Lehre ergänzende, Veranstaltungen anbieten

Hast du Lust mitzumachen?

Oder möchtest du einfach nur ein paar coole
Leute (uns ;)) kennenlernen, die das gleiche studieren wie du?

Melde dich einfach (völlig unverbindlich) bei uns und schau mal bei einem Treffen vorbei. **Wir freuen uns auf dich!**

Schreib uns einfach eine Mail an fgwinfo@tu-braunschweig.de
oder kontaktiere uns über Whatsapp, Instagram o.Ä.



2 Dynamic Programming

Algorithmen und Datenstrukturen 2
Sommer 2023

Prof. Dr. Sándor Fekete

Beispiel

Beispiel

Beispiel 1.10.

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

Gesamtsumme: 240

Beispiel

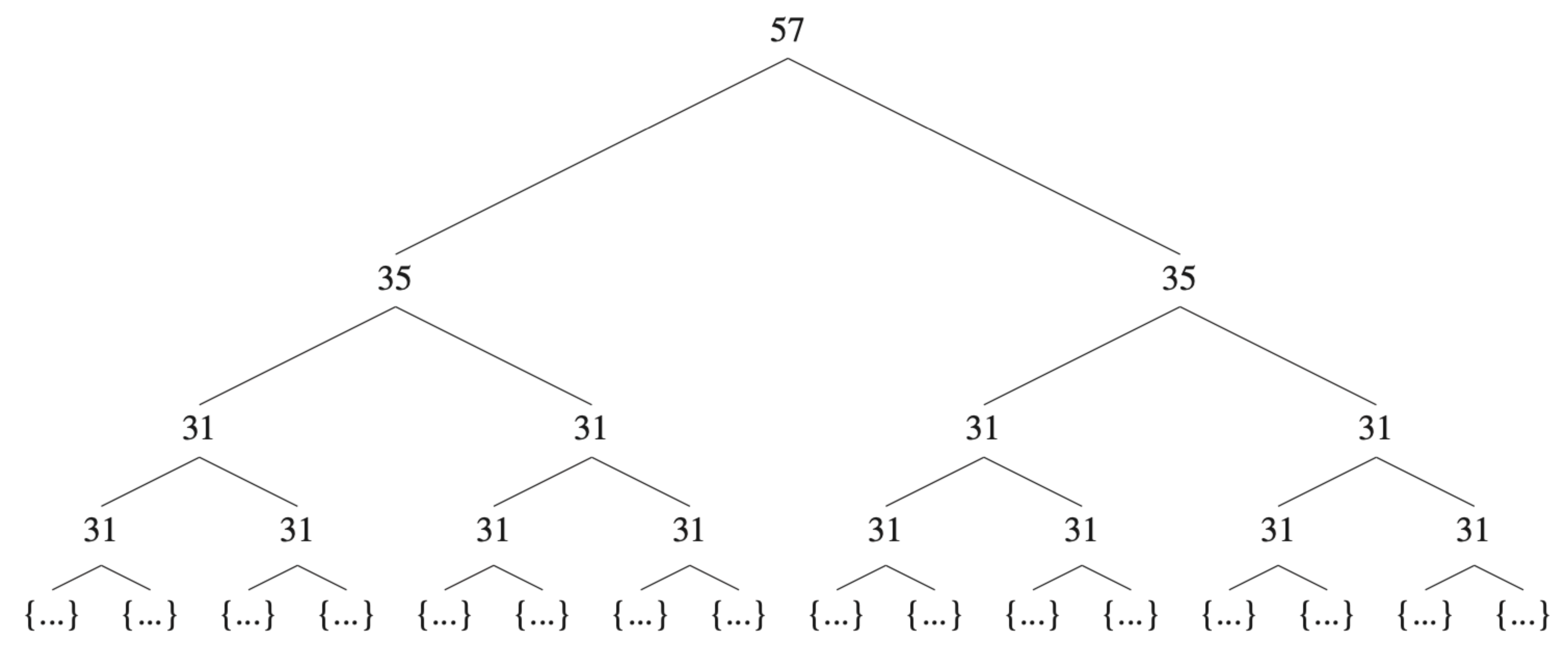
Beispiel 1.10.

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

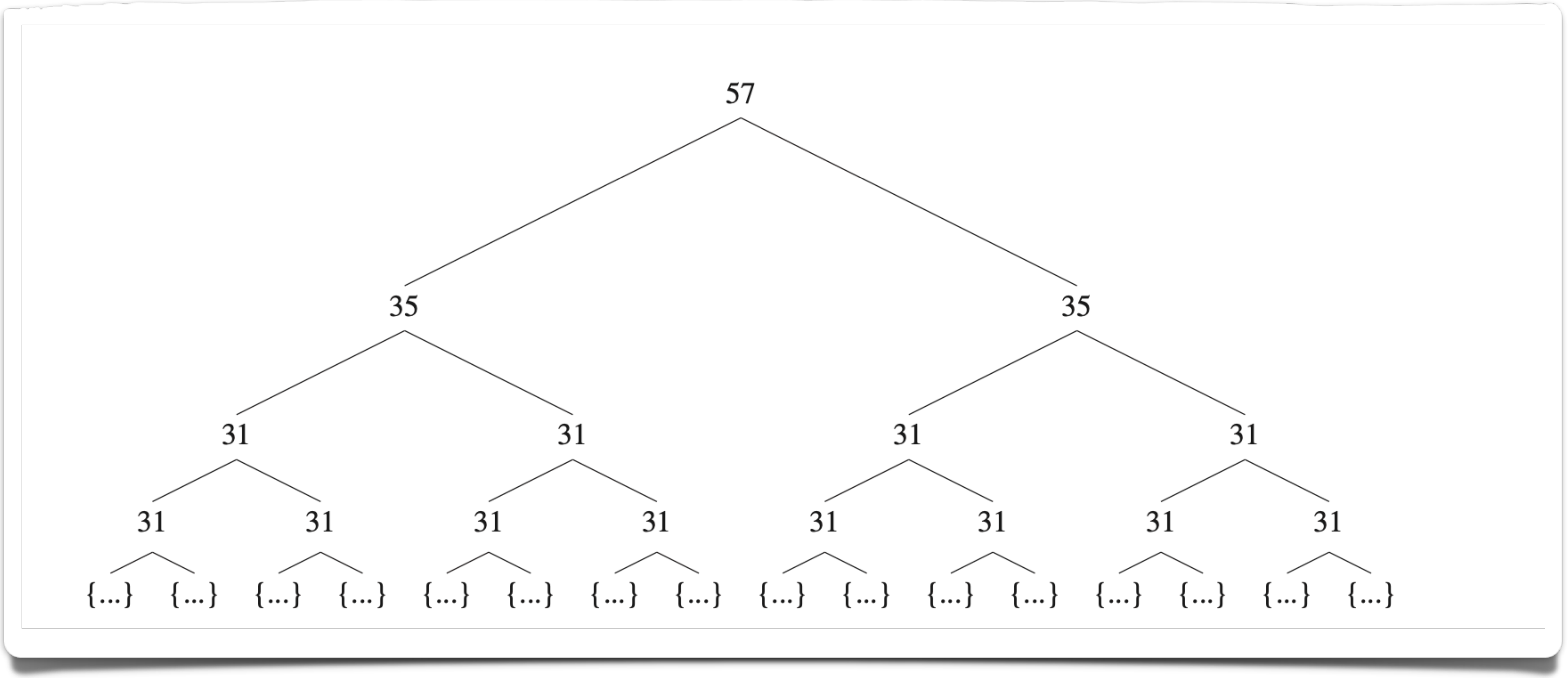
Gesamtsumme: 240

Finde Aufteilung in zwei gleiche Teilmengen!

Enumerationsprinzip

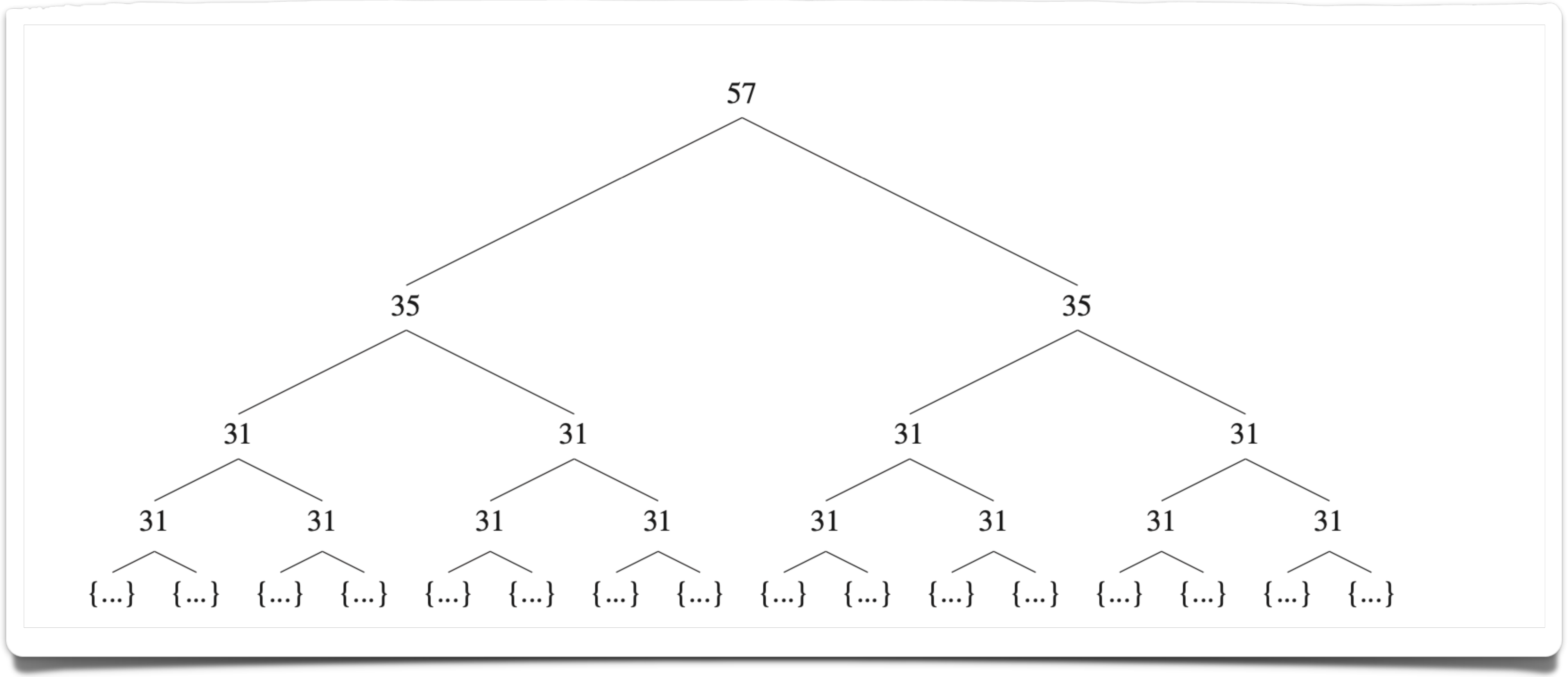


Enumerationsprinzip



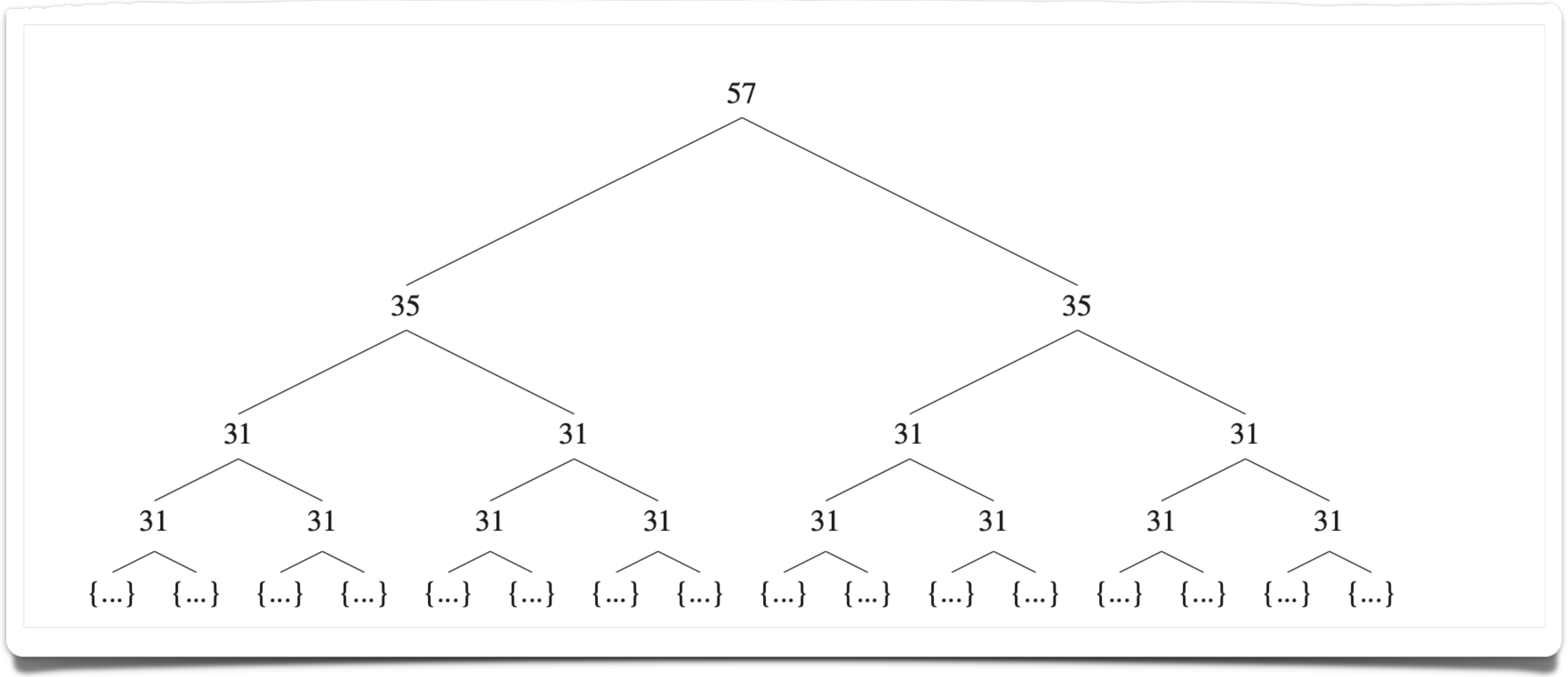
- Exponentiell viele Fälle!

Enumerationsprinzip



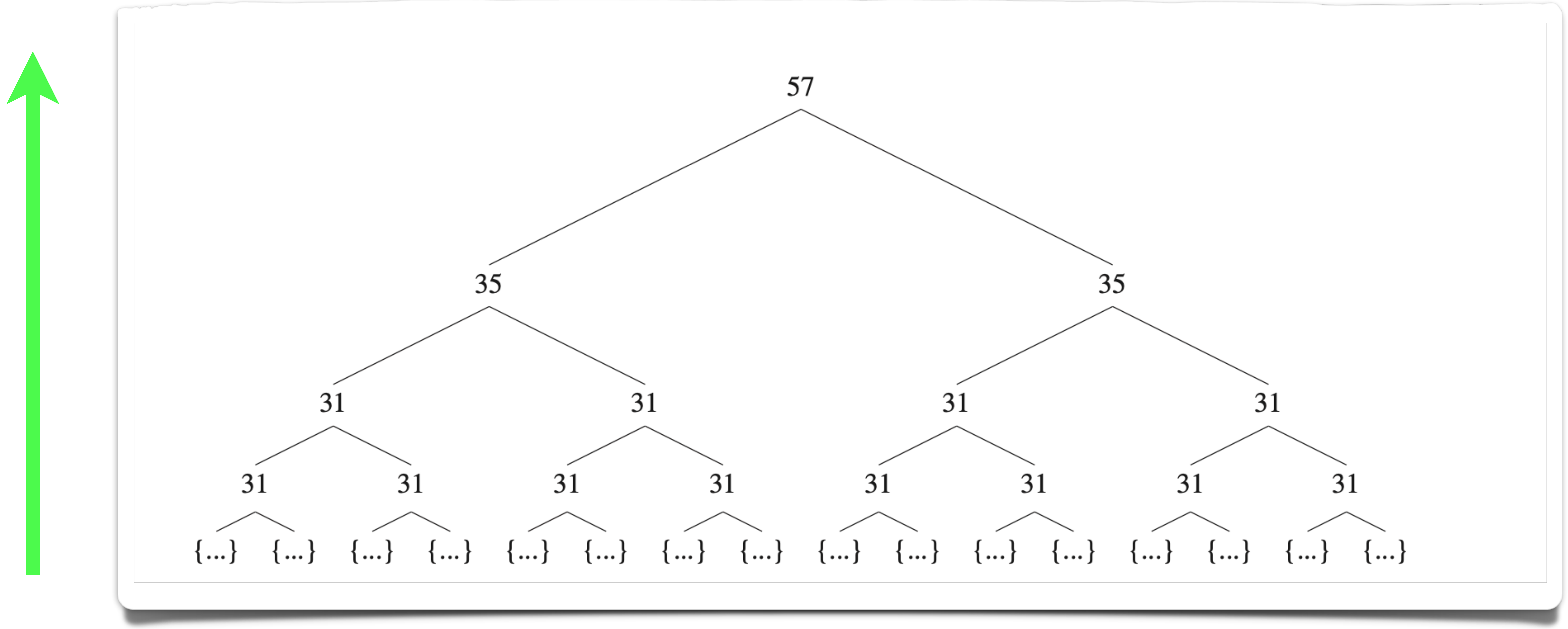
- Exponentiell viele Fälle!
- Wie geht das systematisch?

Enumerationsprinzip



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

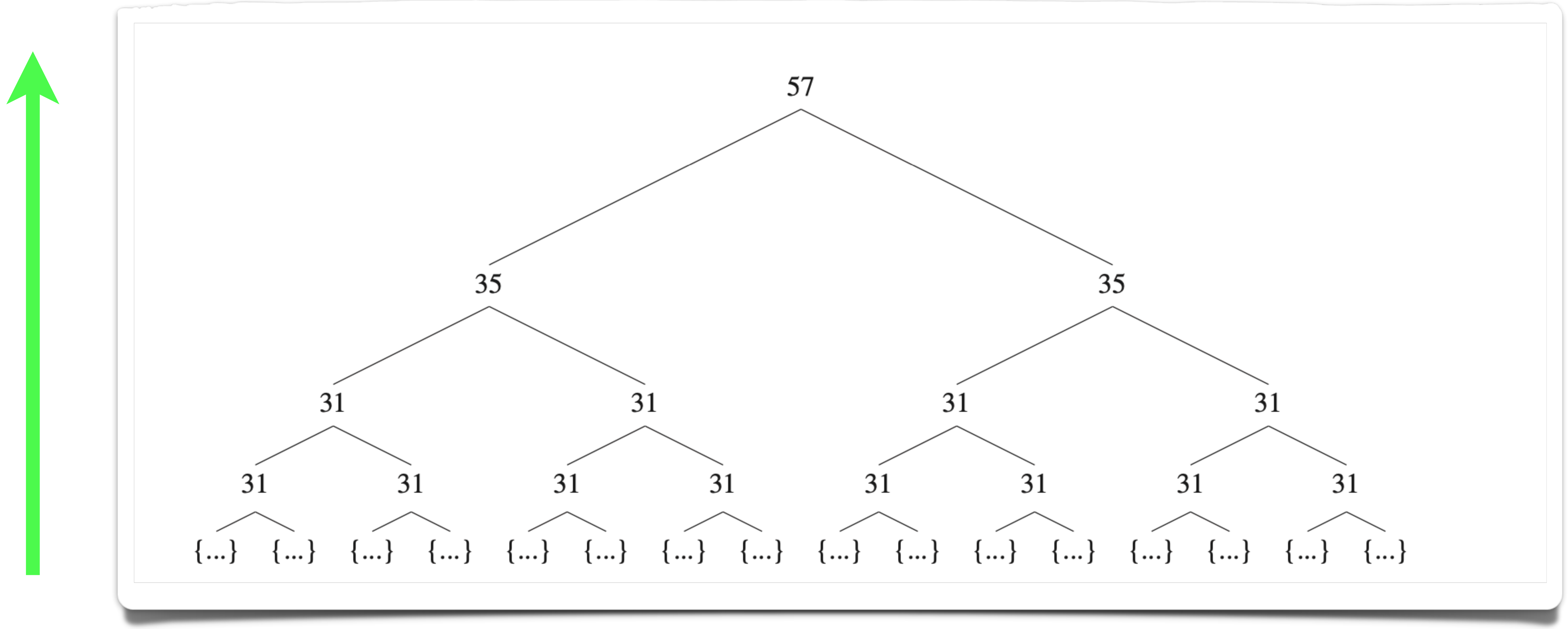
Enumerationsprinzip



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

Enumerationsprinzip

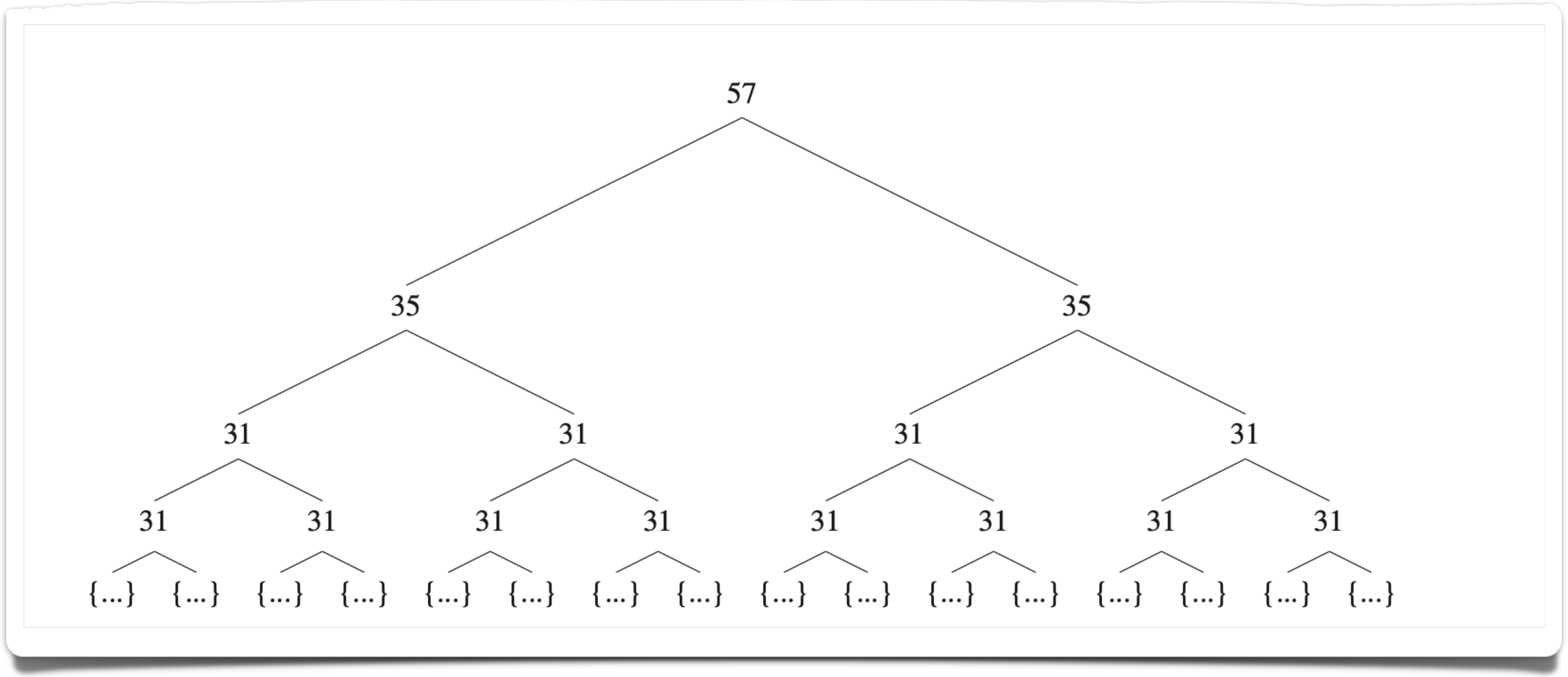
Dynamic Programming



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

Enumerationsprinzip

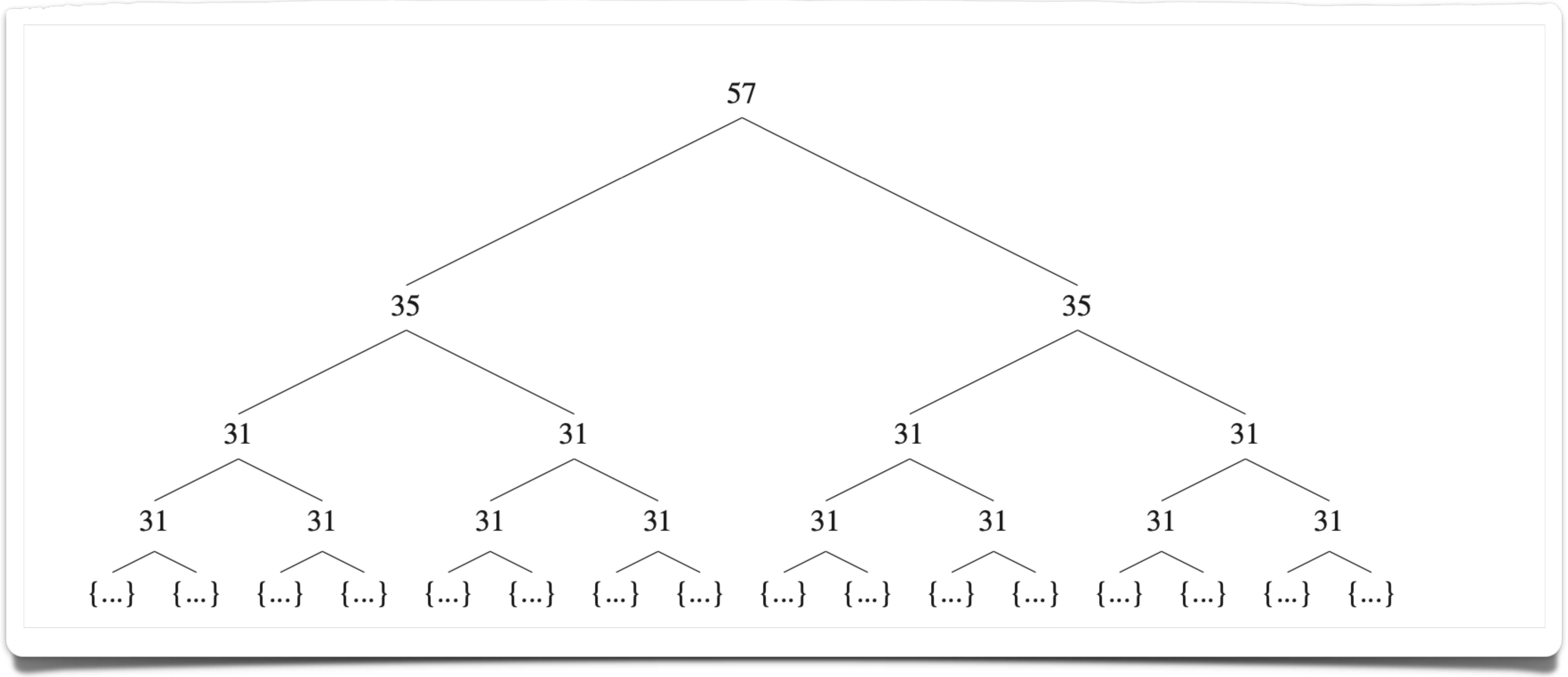
Dynamic Programming



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

Enumerationsprinzip

Dynamic Programming

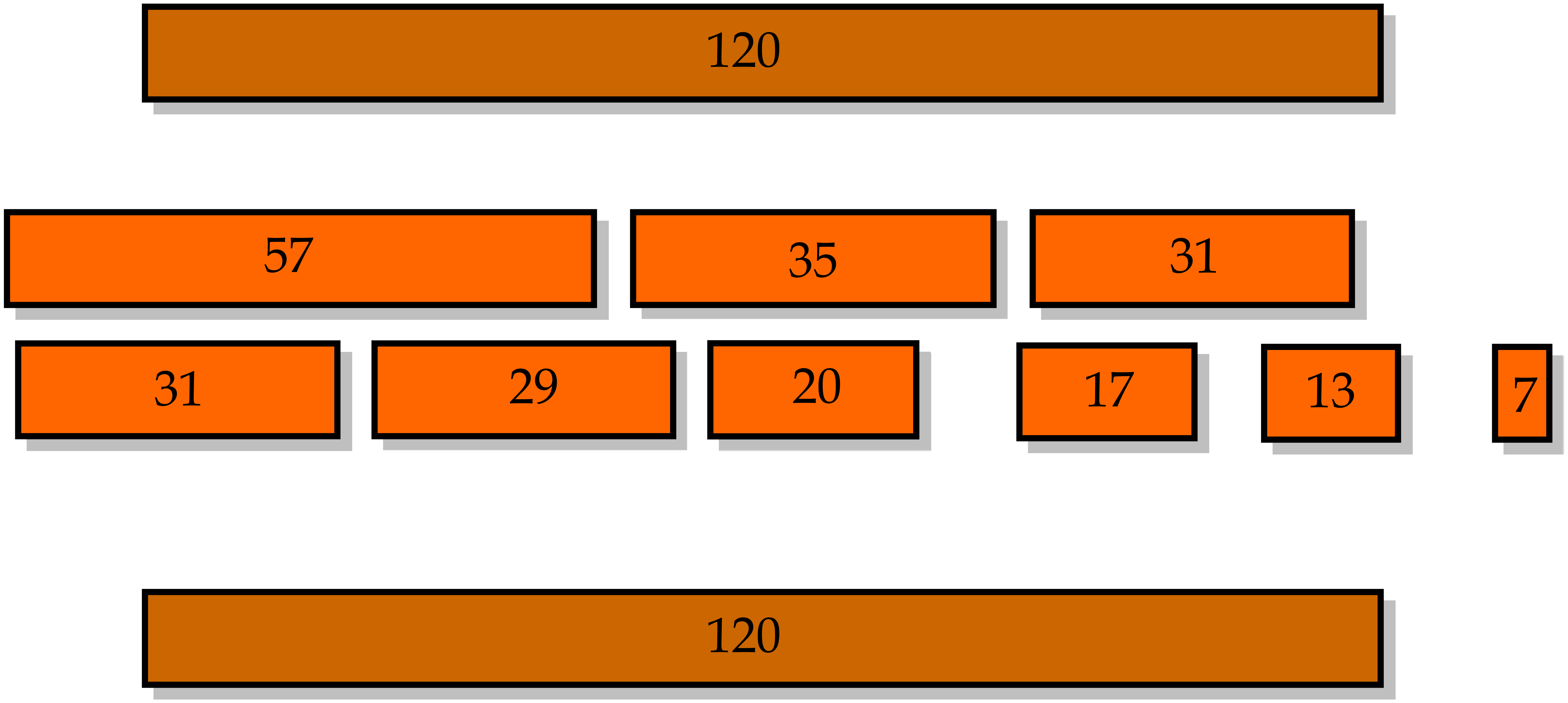


Branch and Bound

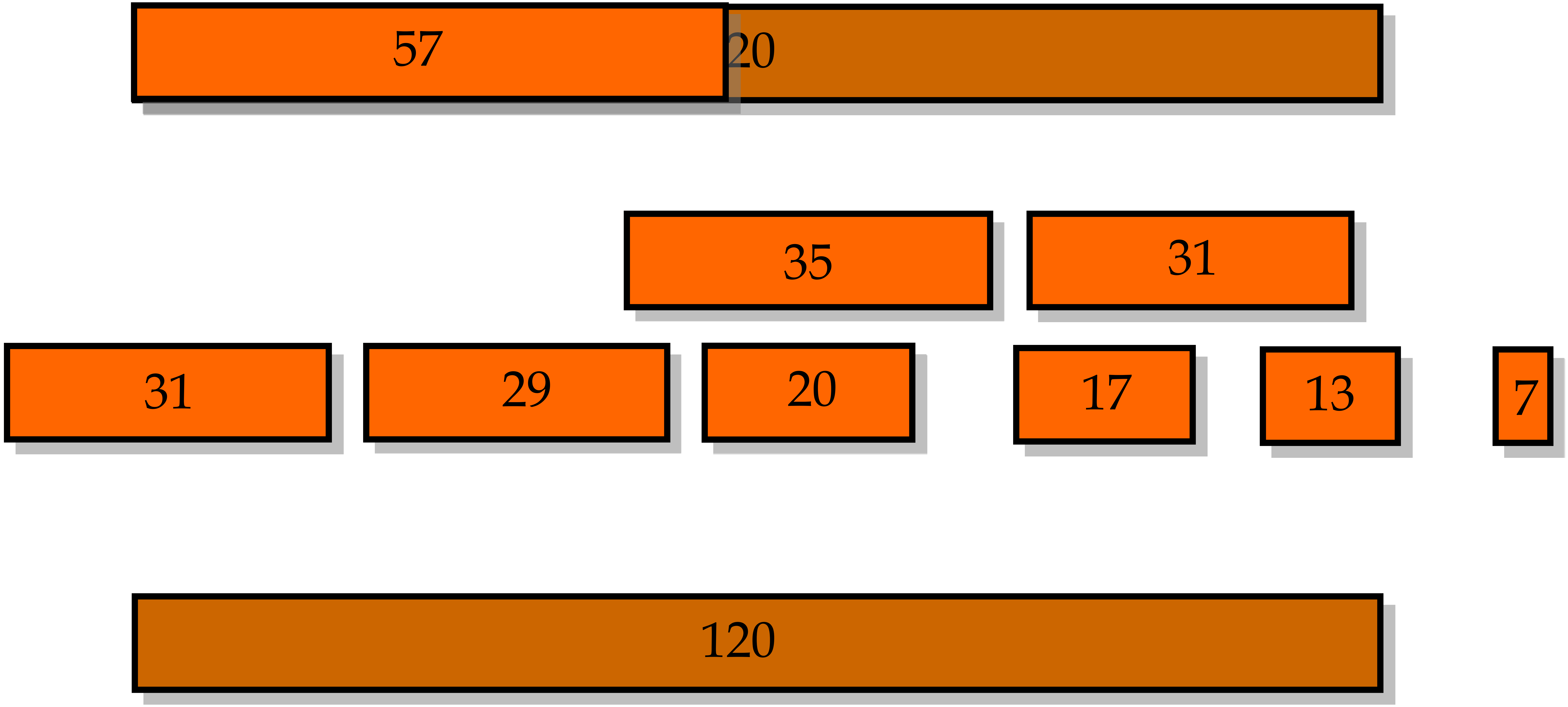
- Exponentiell viele Fälle!
- Wie geht das systematisch?
- Wo kann man Arbeit sparen?

Keine Lösung?!

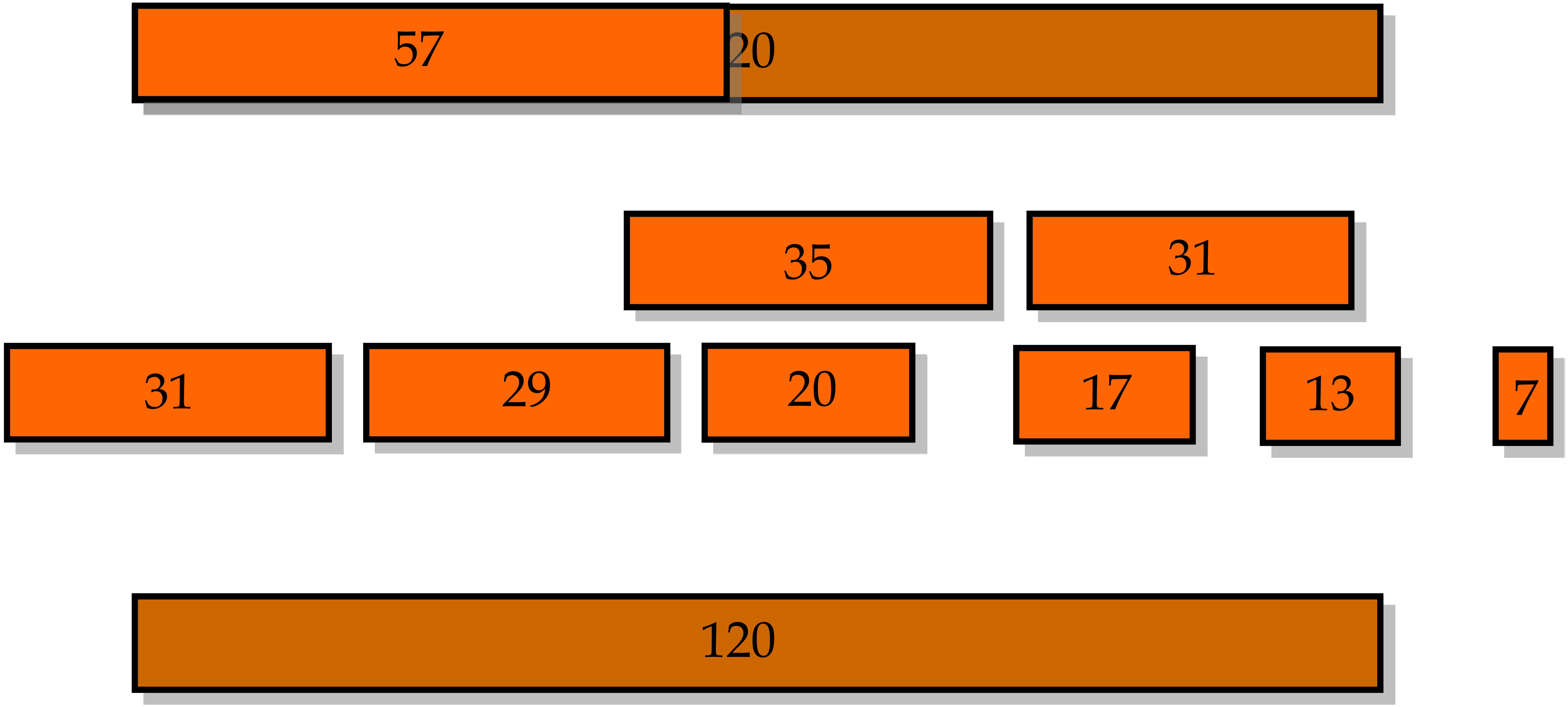
Keine Lösung?!



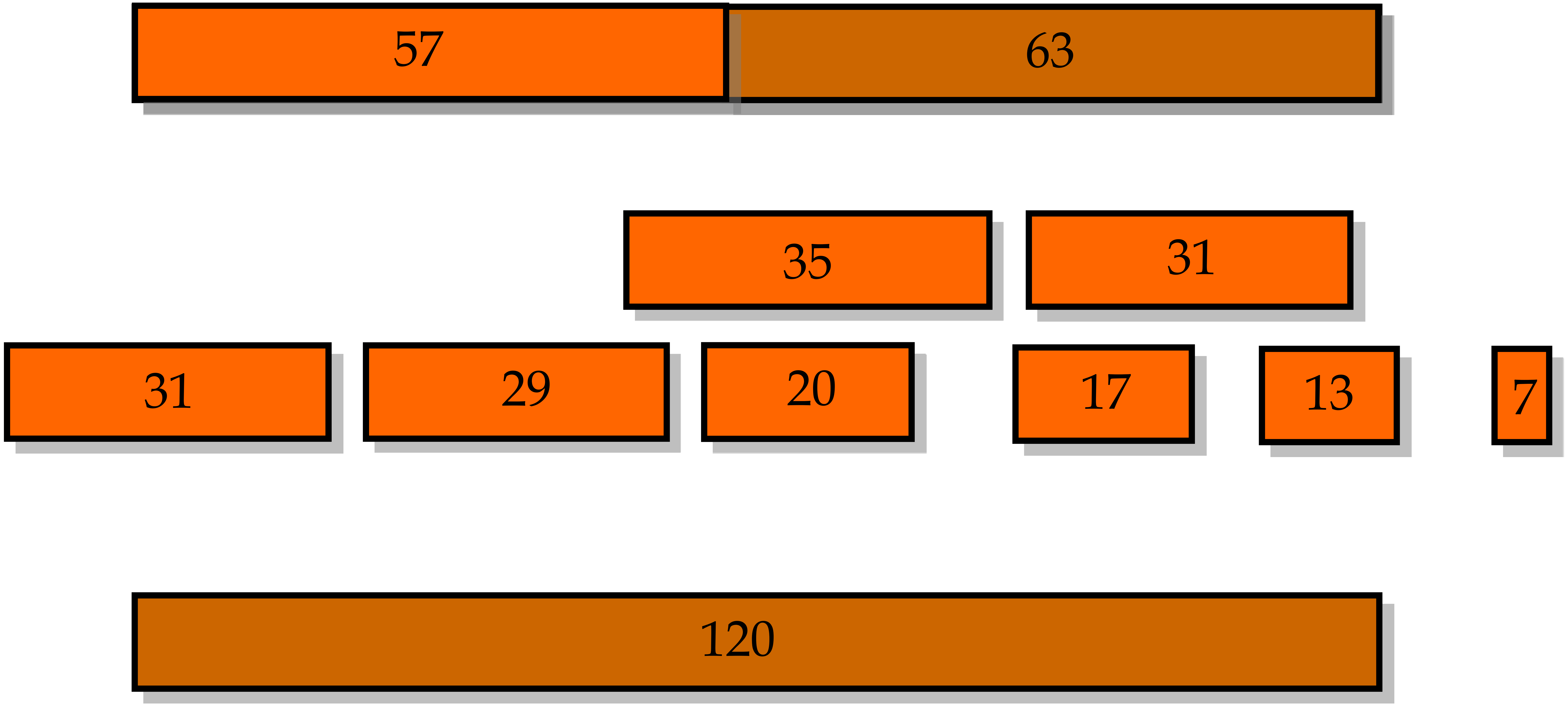
Keine Lösung?!



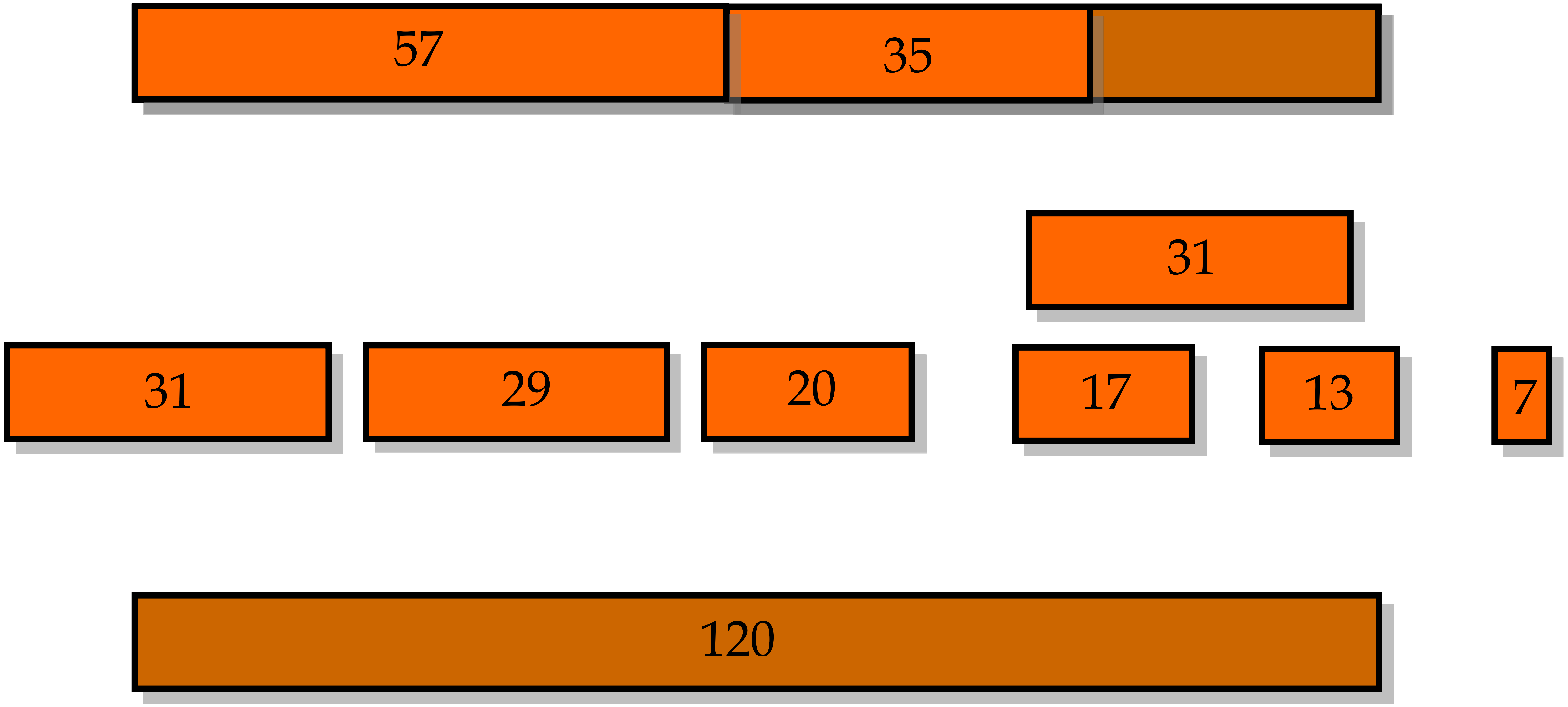
Keine Lösung?!



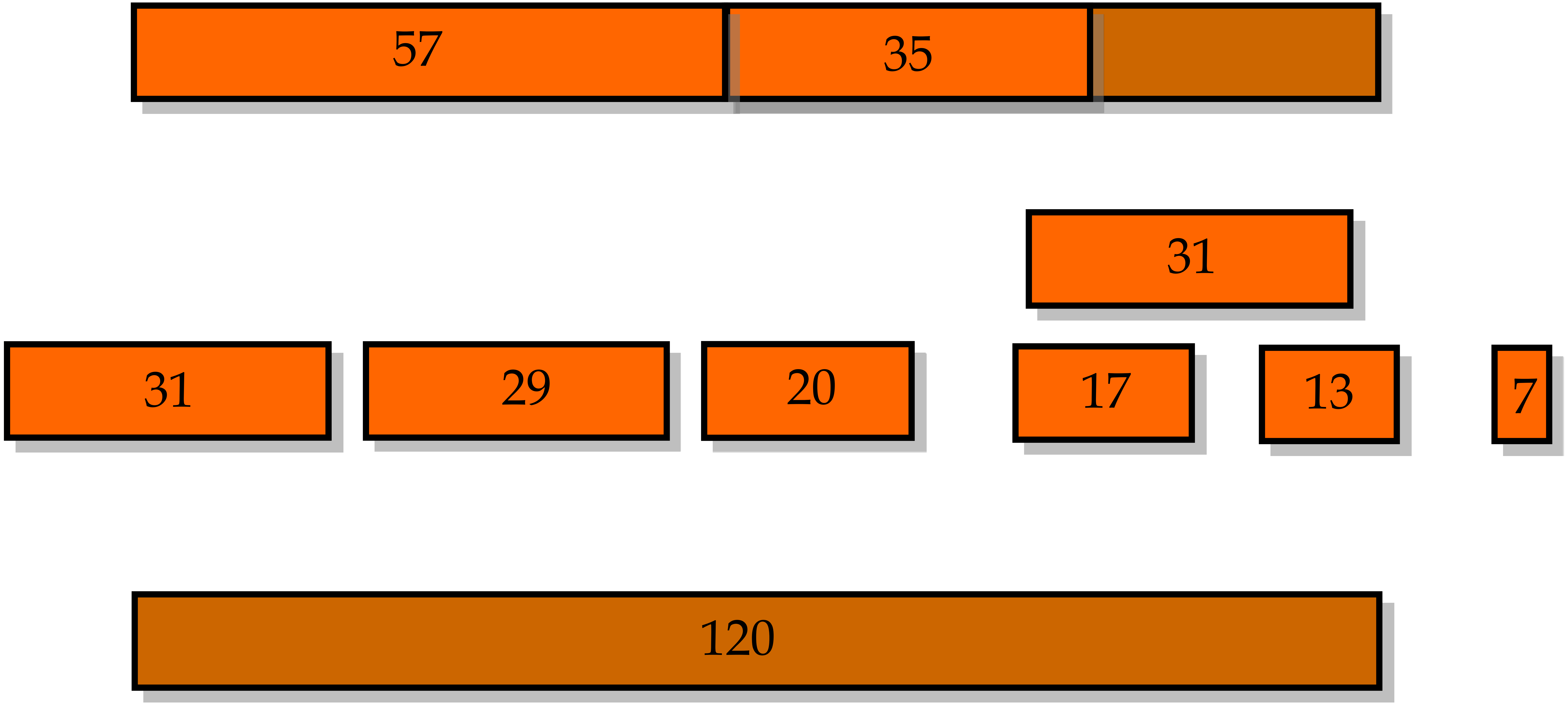
Keine Lösung?!



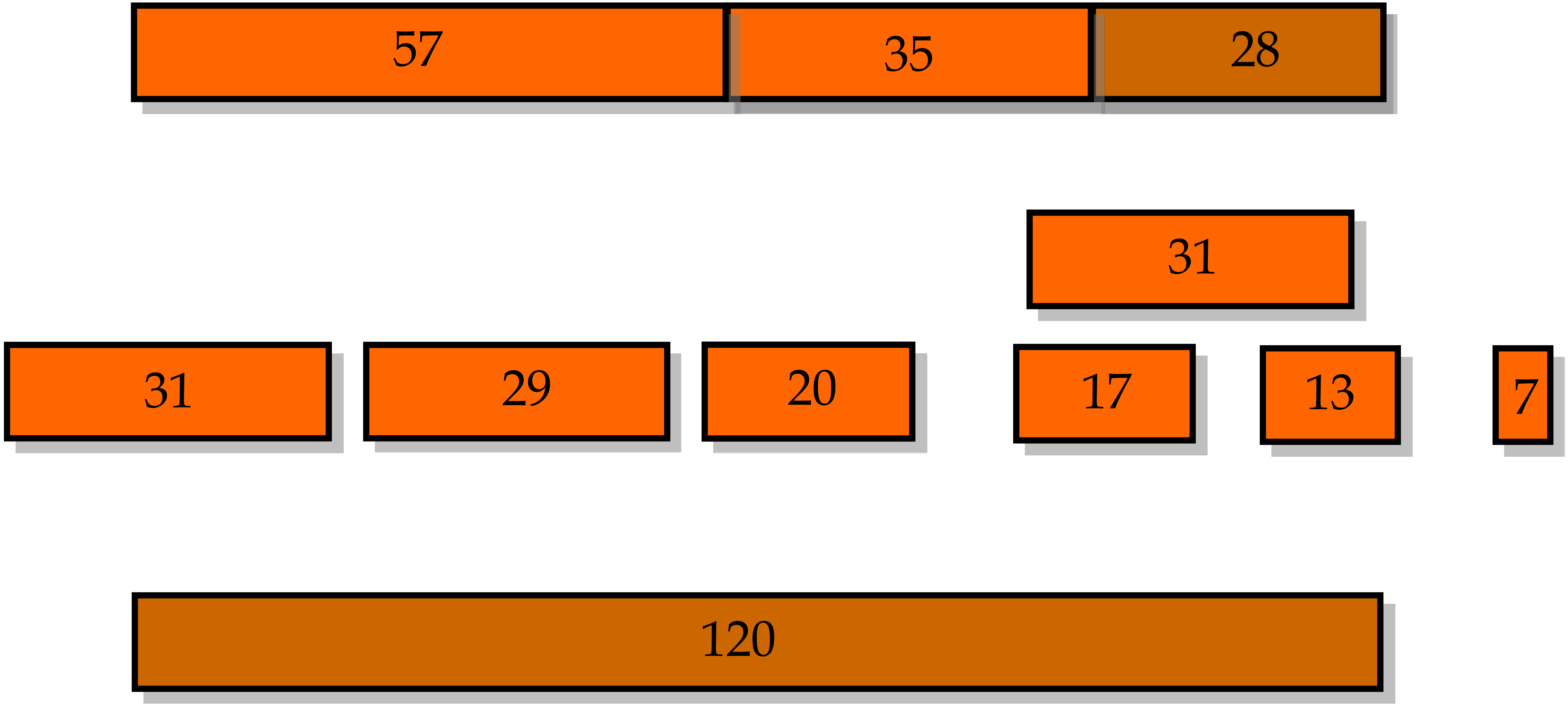
Keine Lösung?!



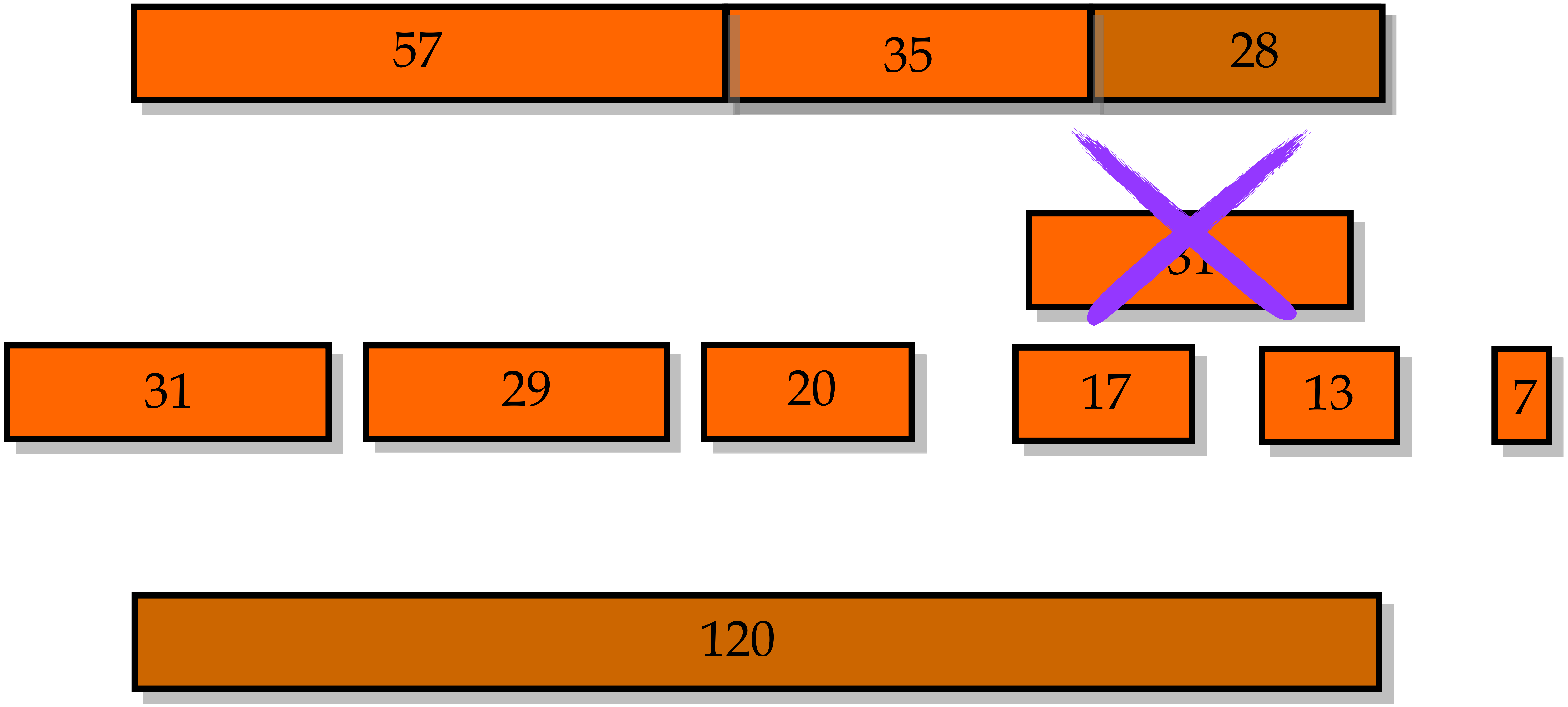
Keine Lösung?!



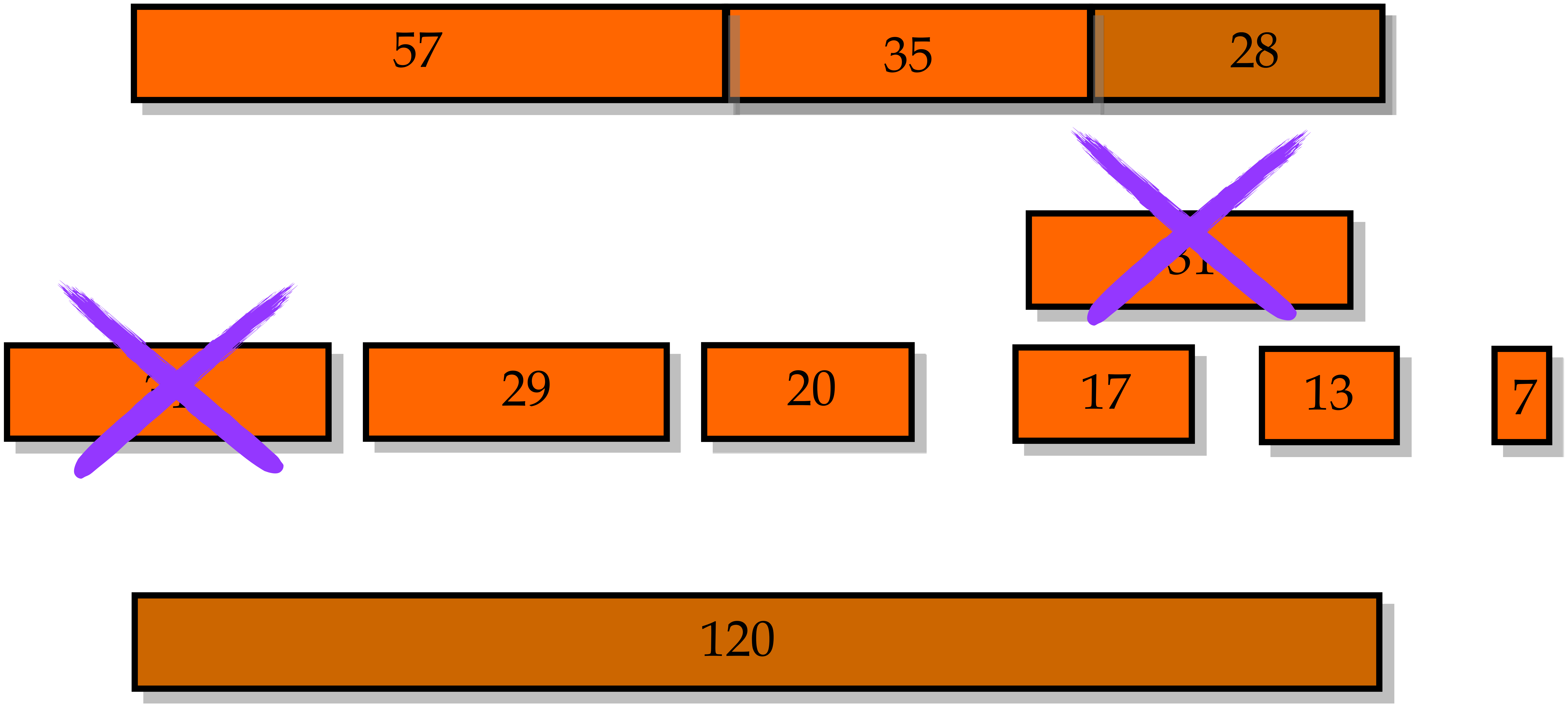
Keine Lösung?!



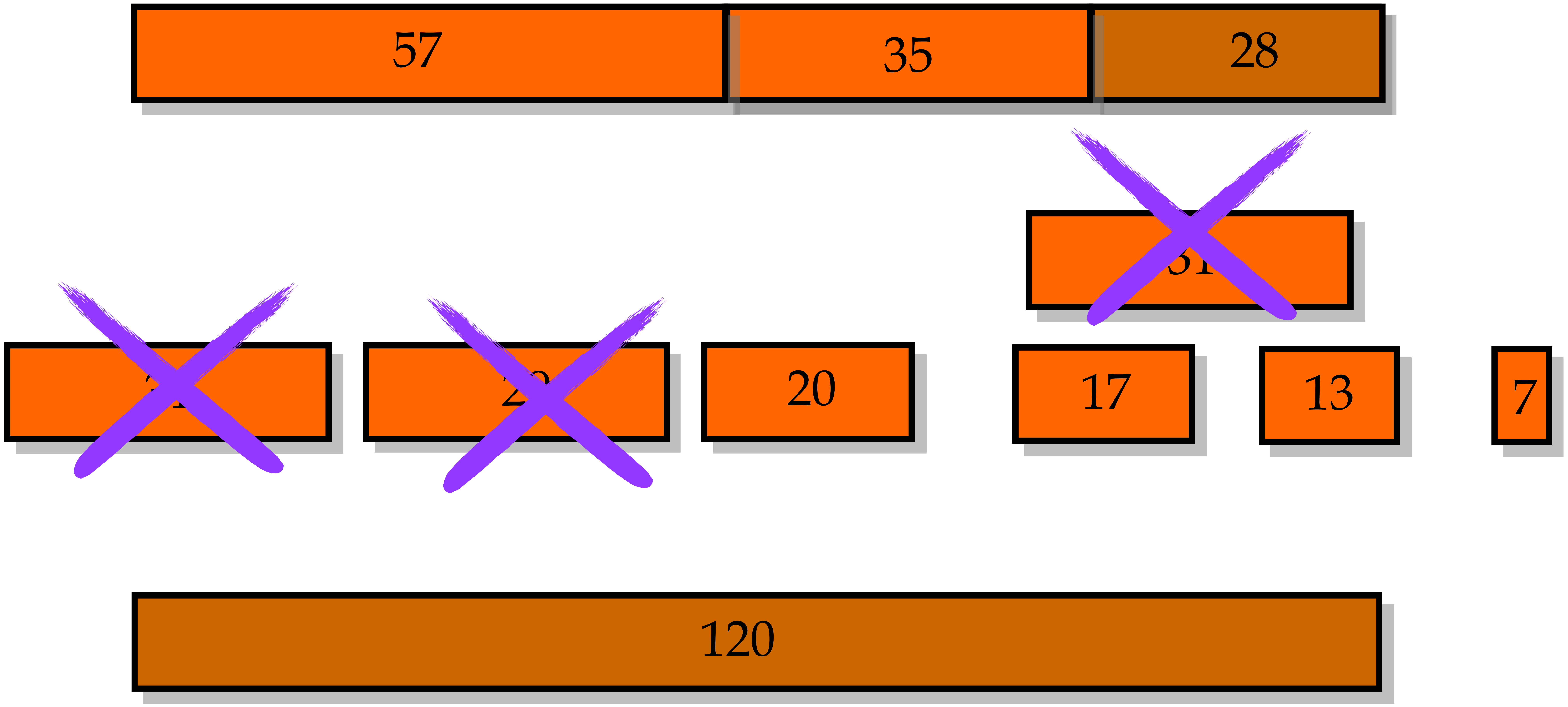
Keine Lösung?!



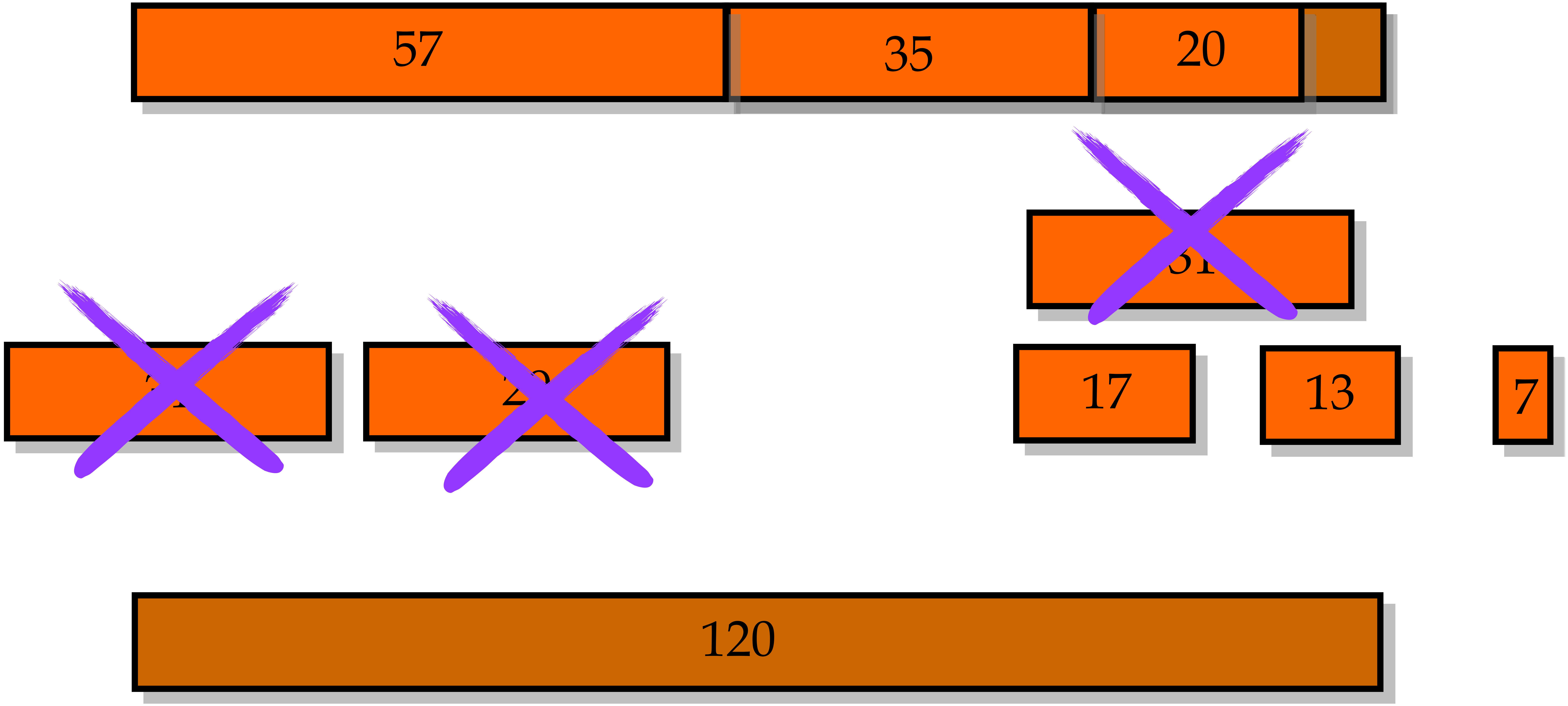
Keine Lösung?!



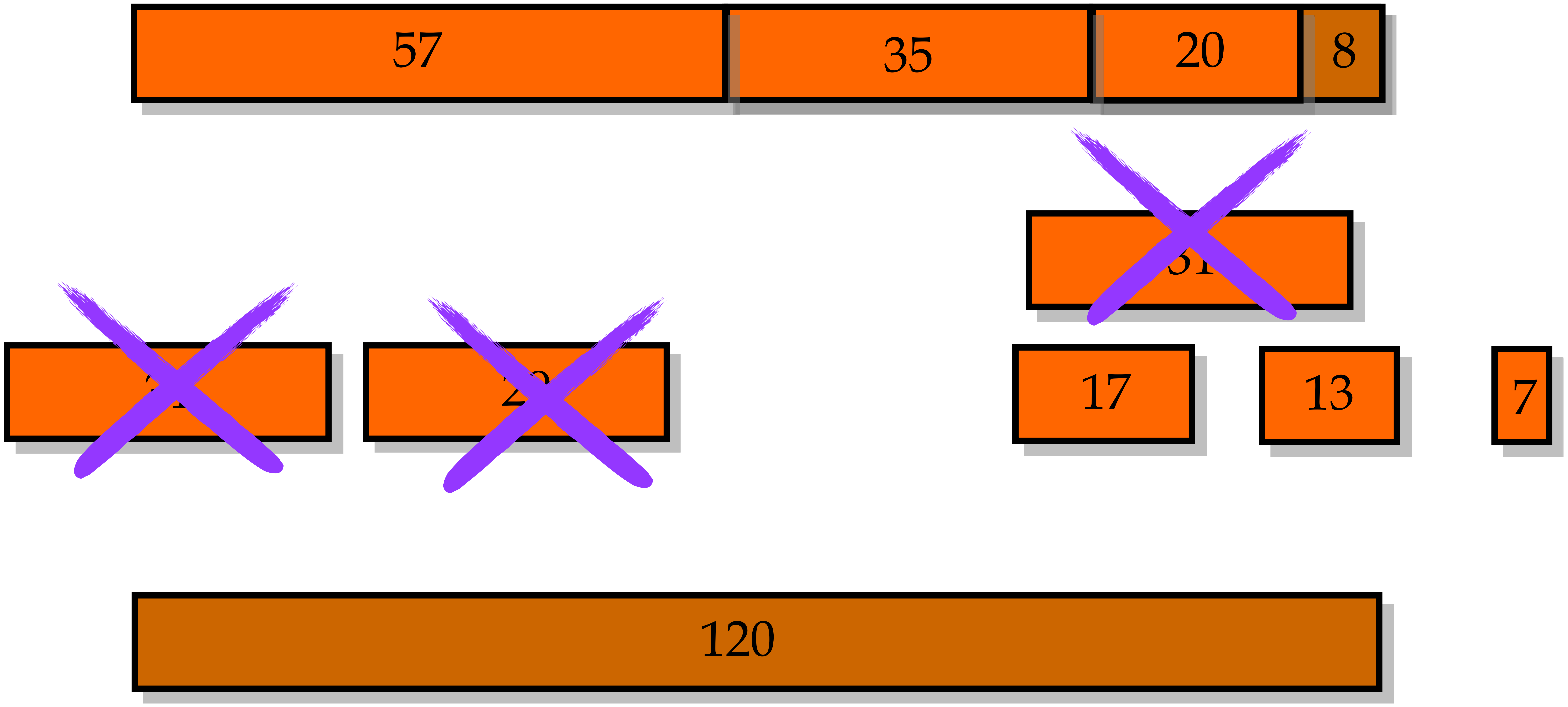
Keine Lösung?!



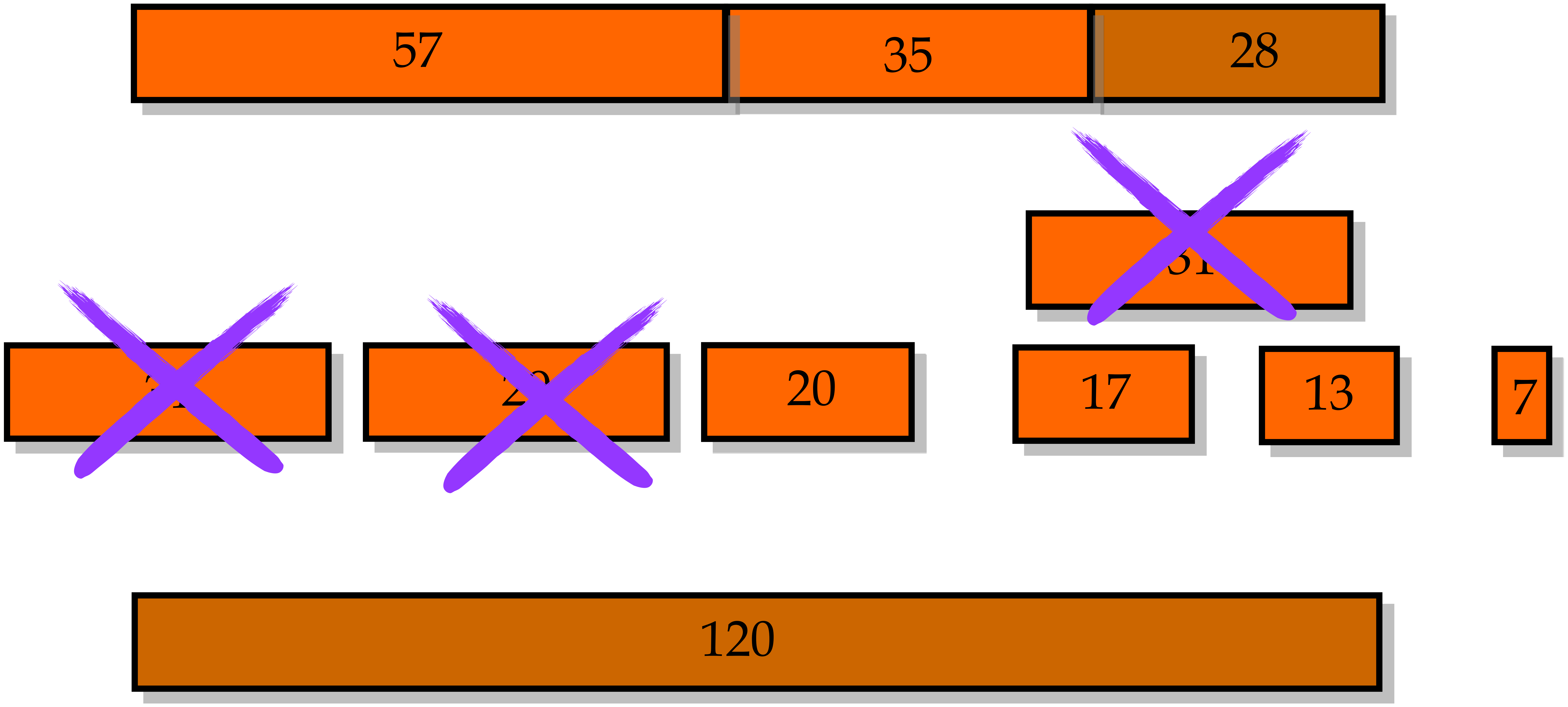
Keine Lösung?!



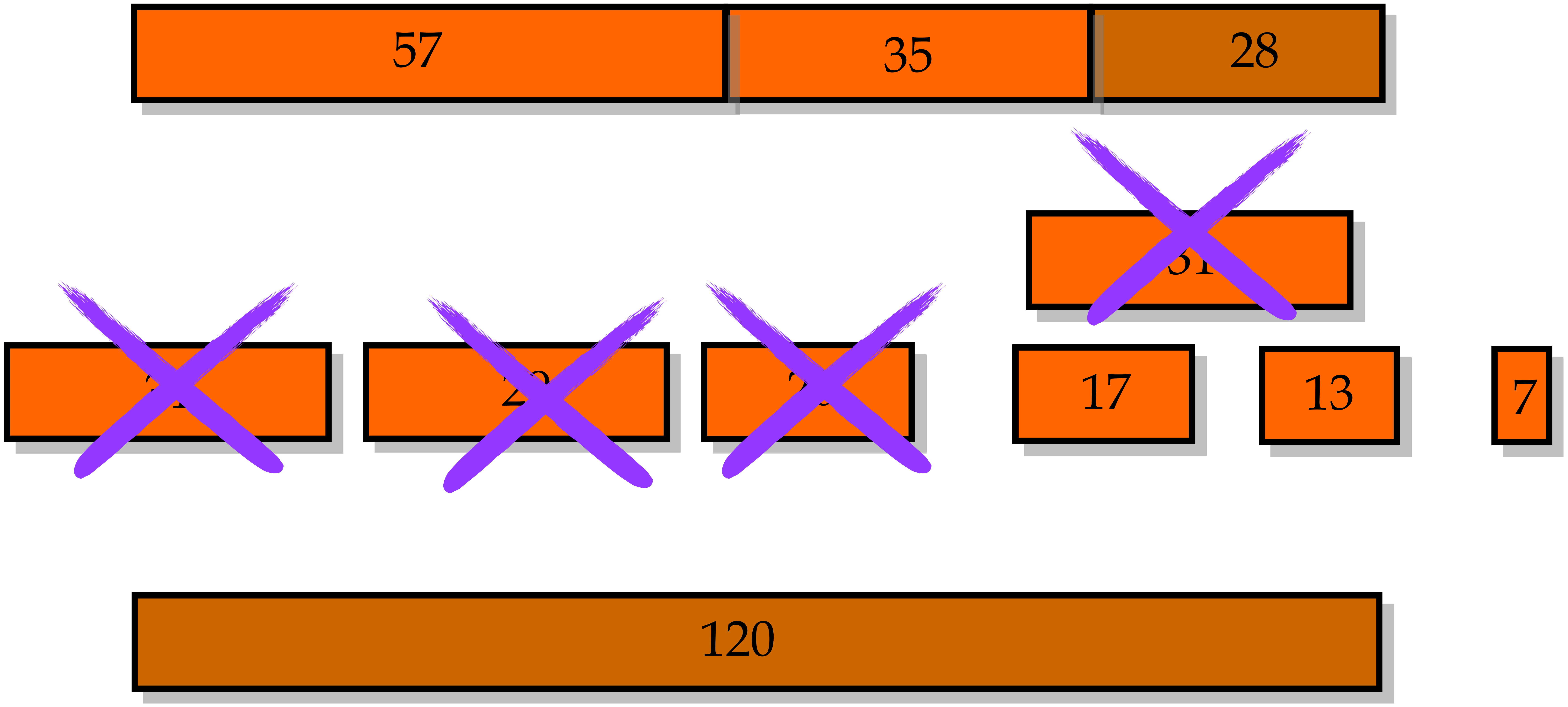
Keine Lösung?!



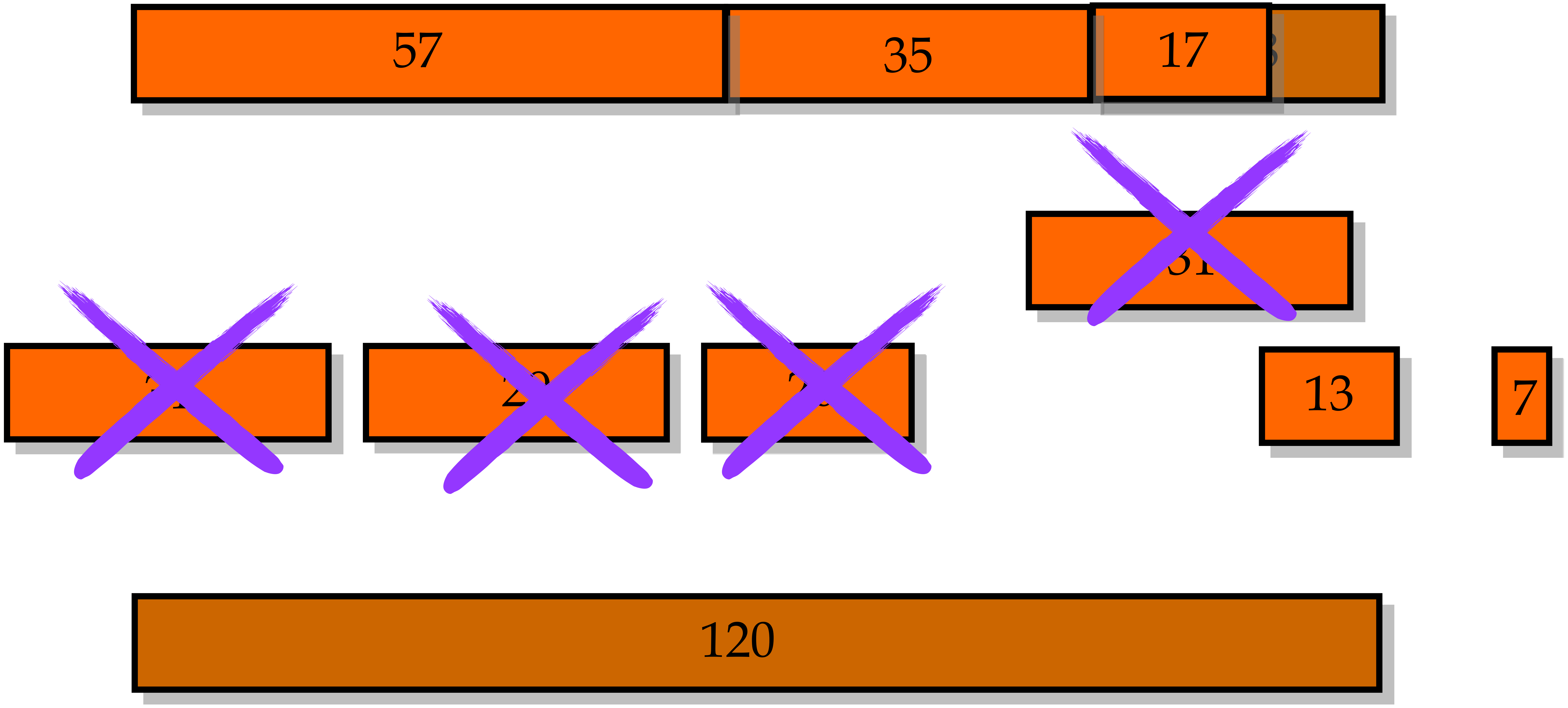
Keine Lösung?!



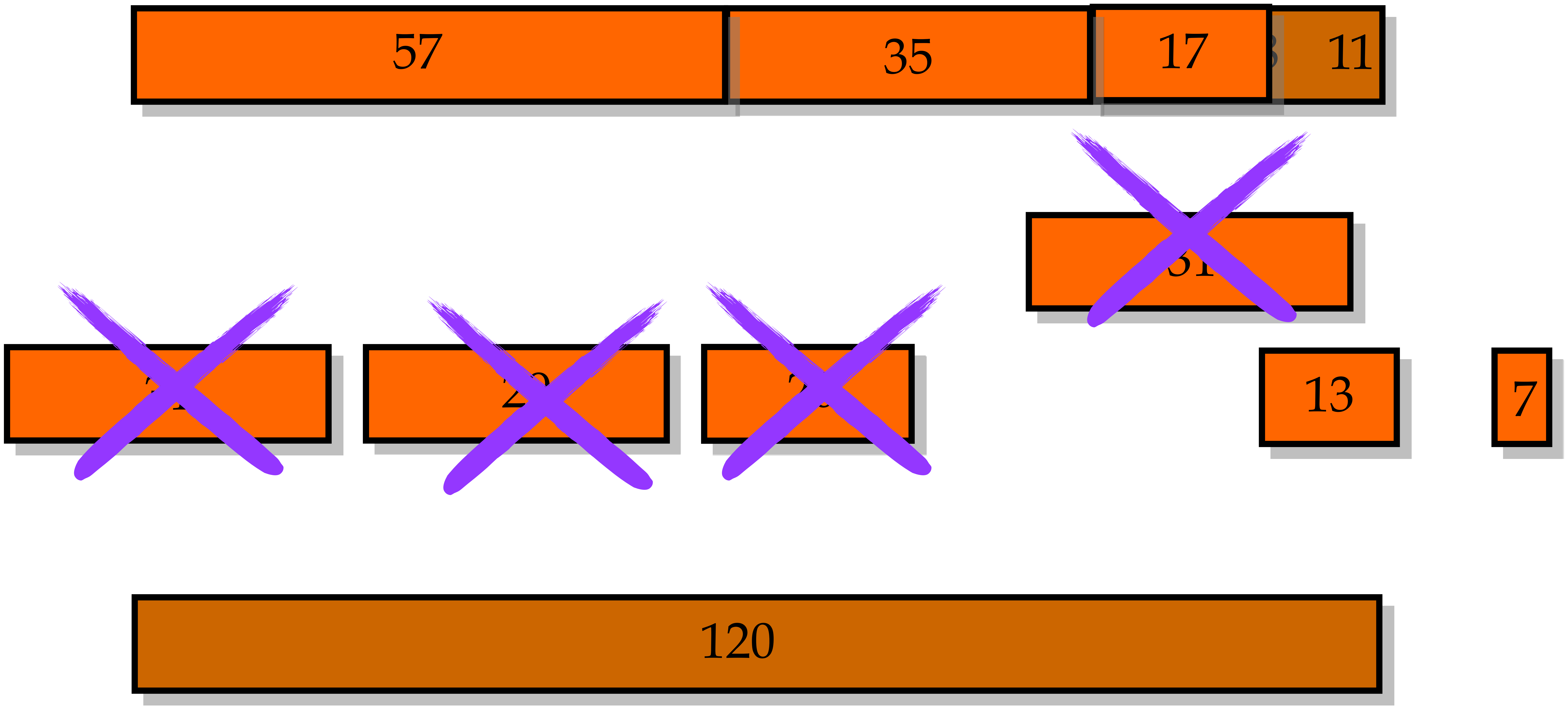
Keine Lösung?!



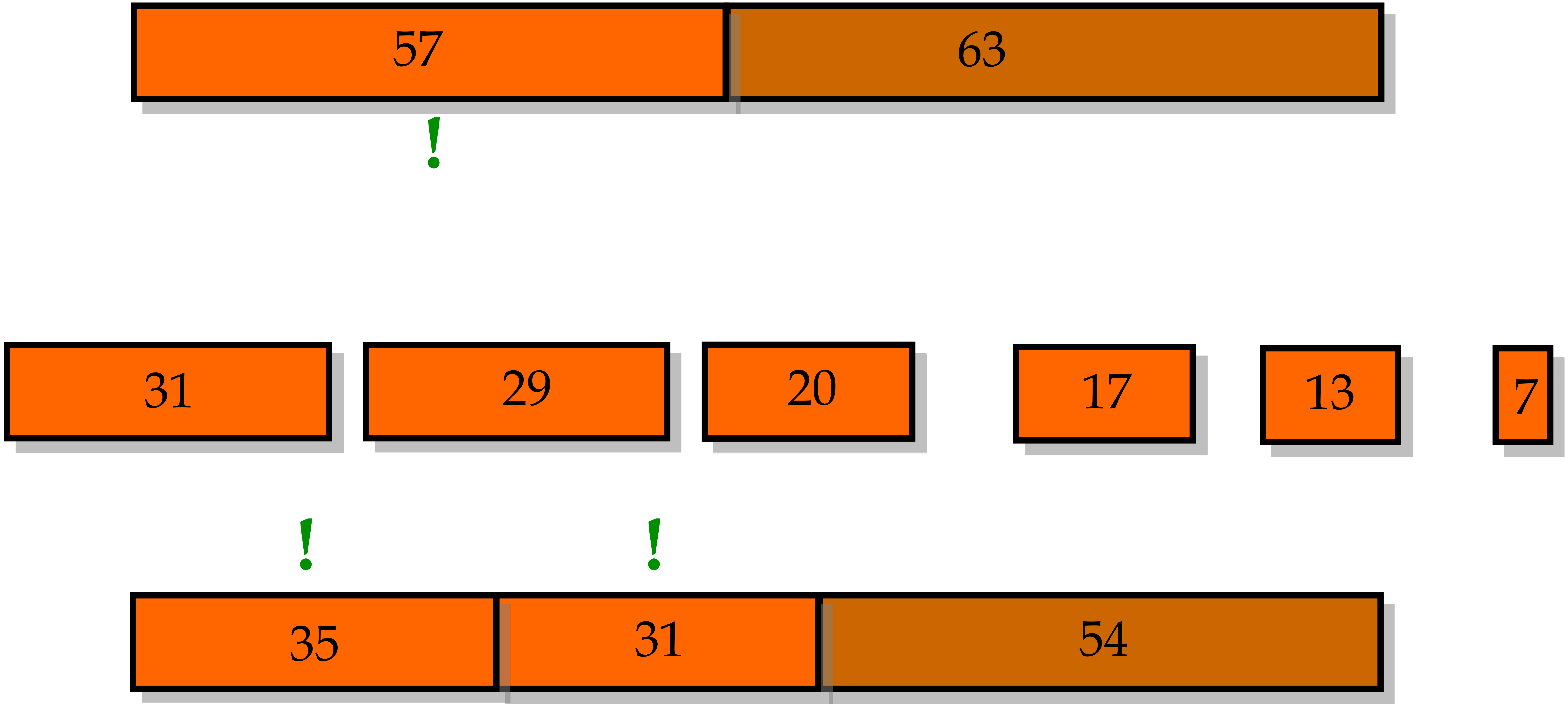
Keine Lösung?!



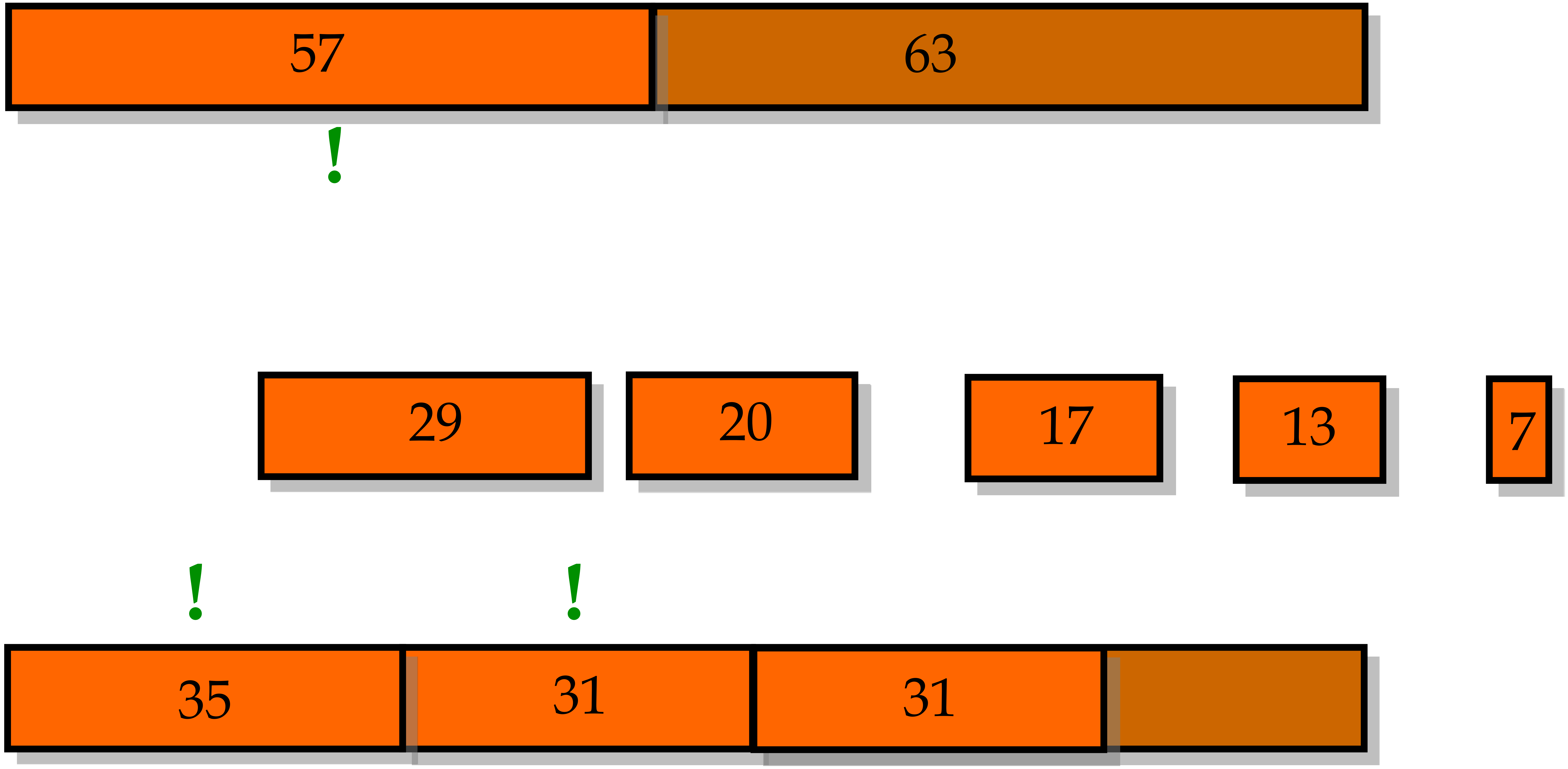
Keine Lösung?!



Keine Lösung?!

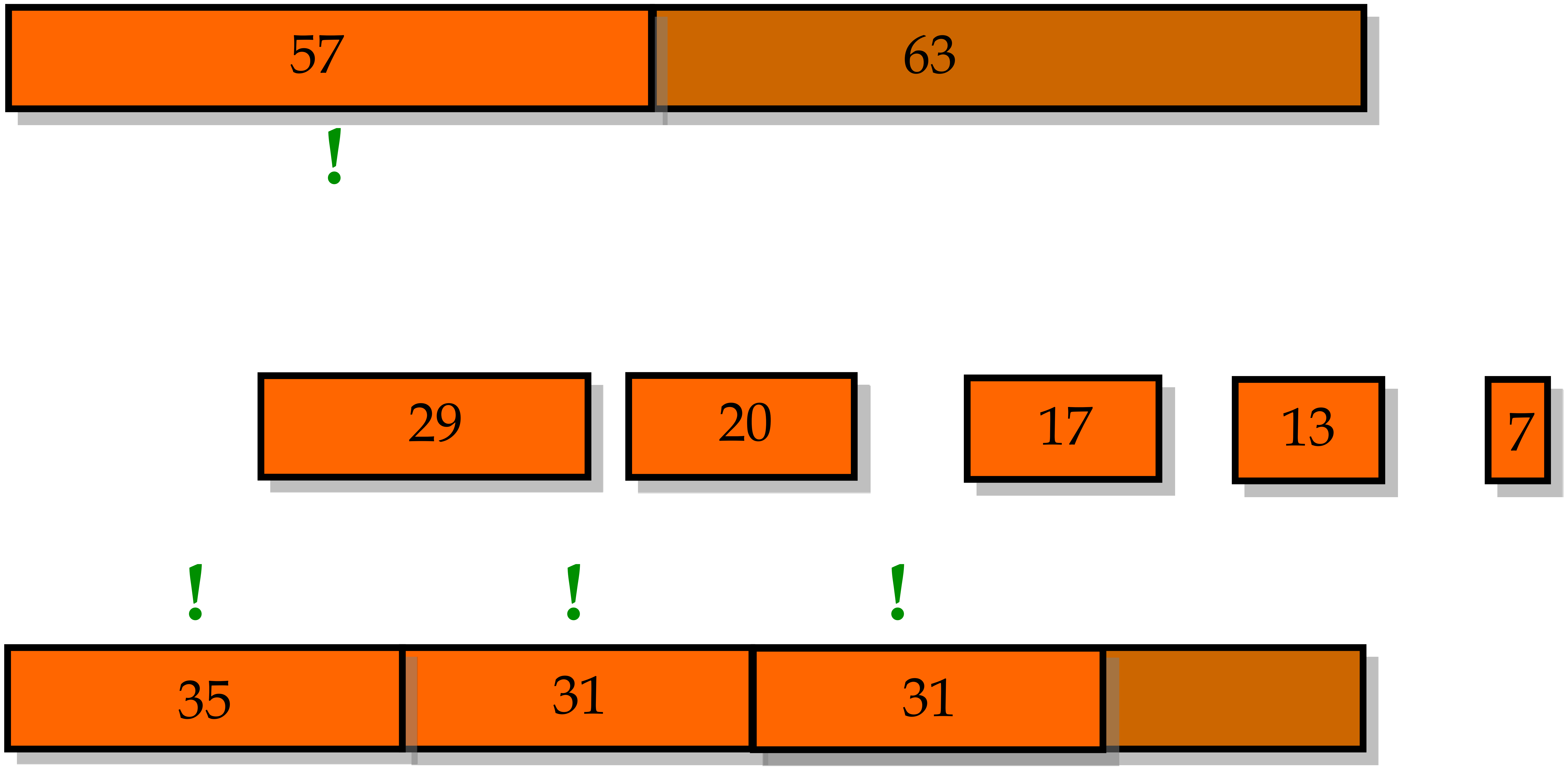


Keine Lösung?!

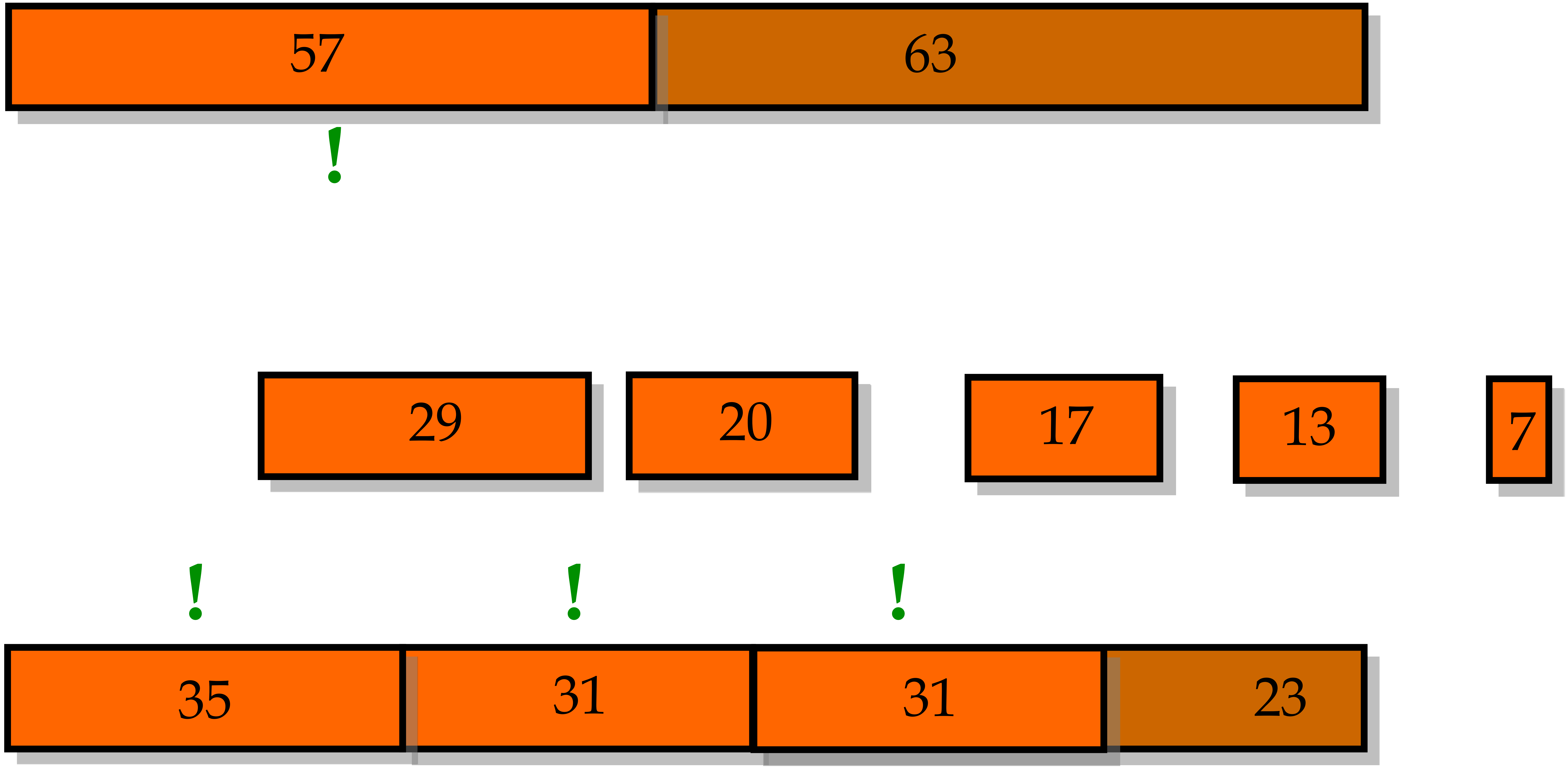


Technische Universität Braunschweig

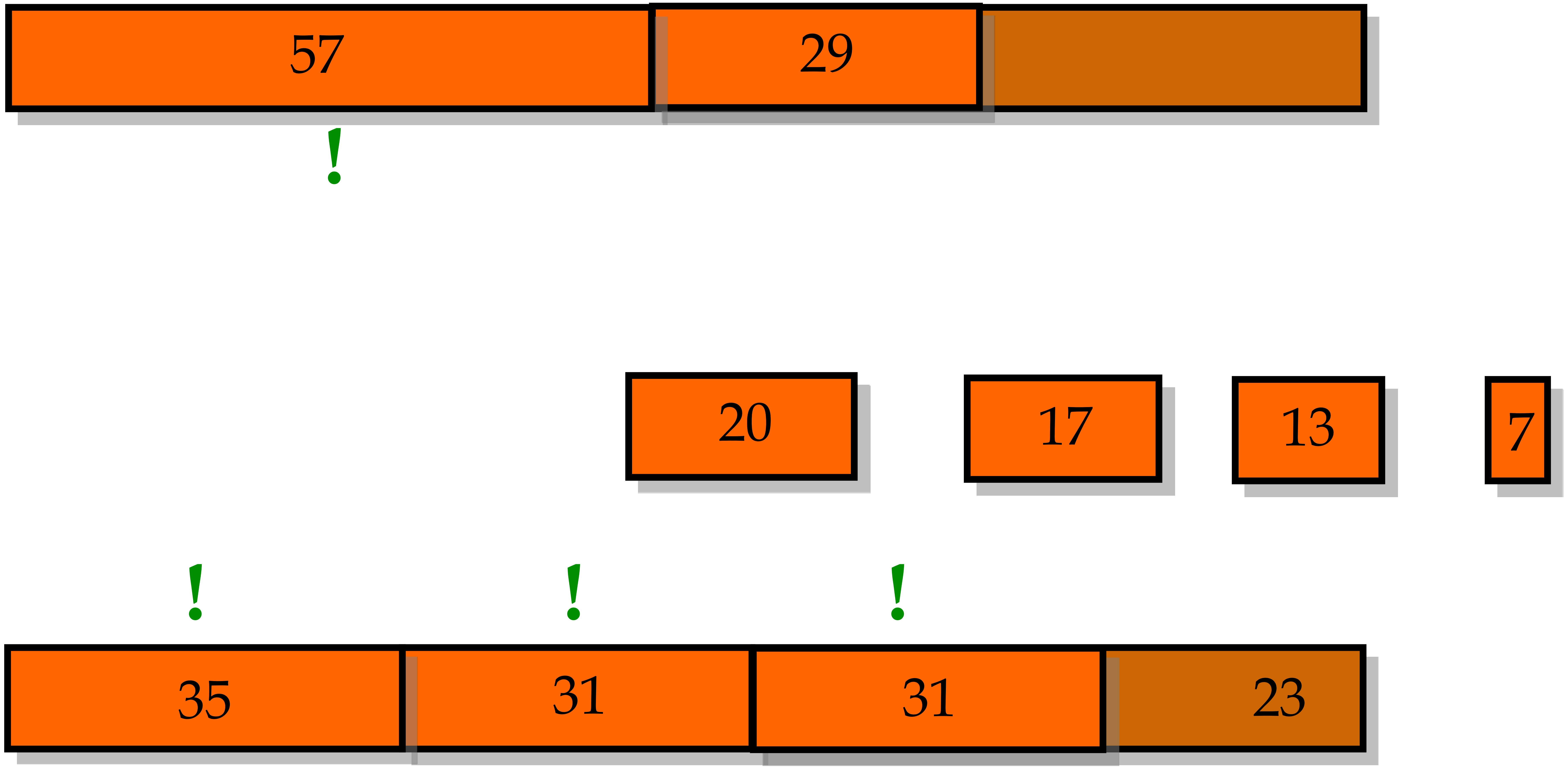
Keine Lösung?!



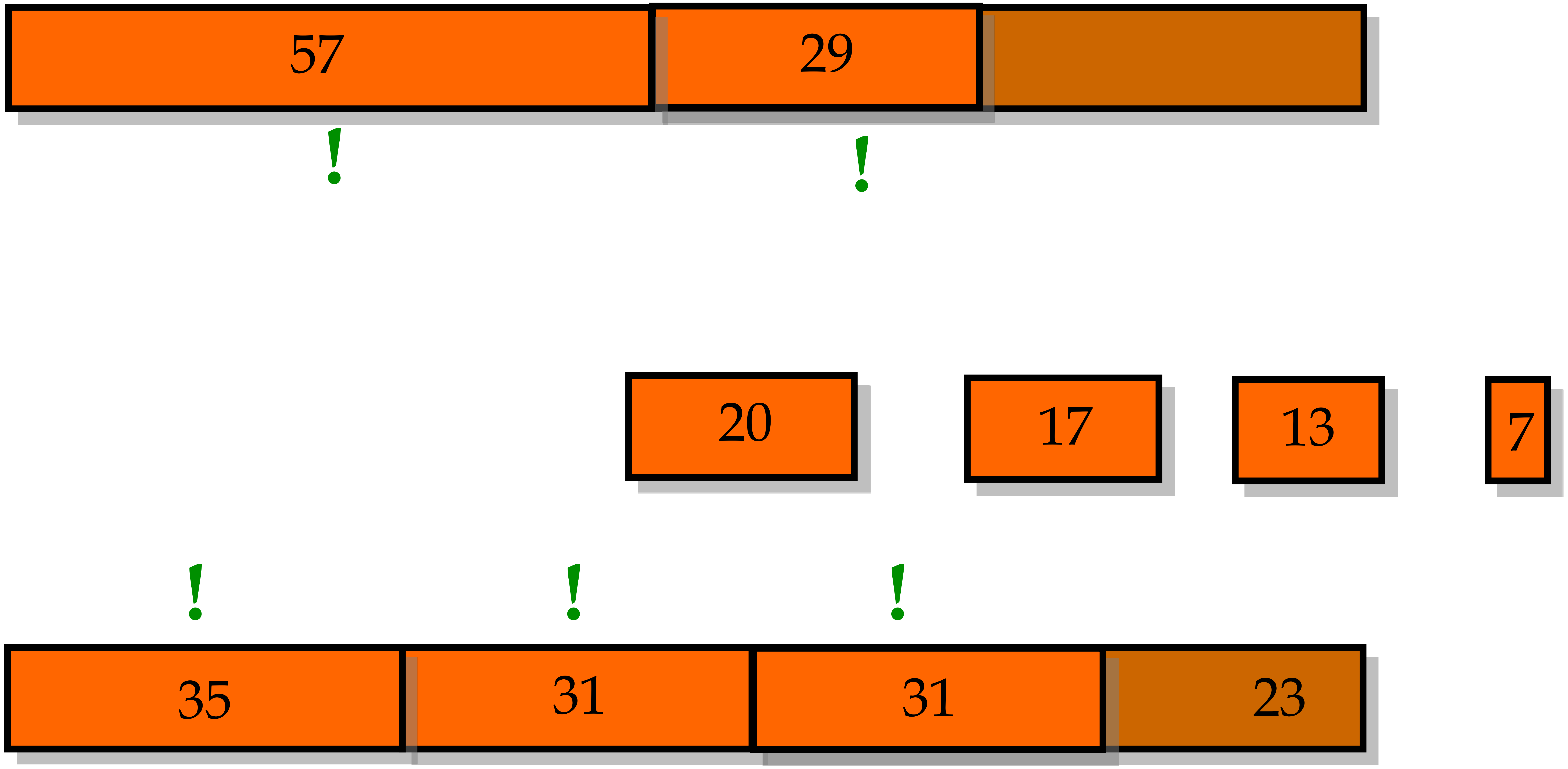
Keine Lösung?!



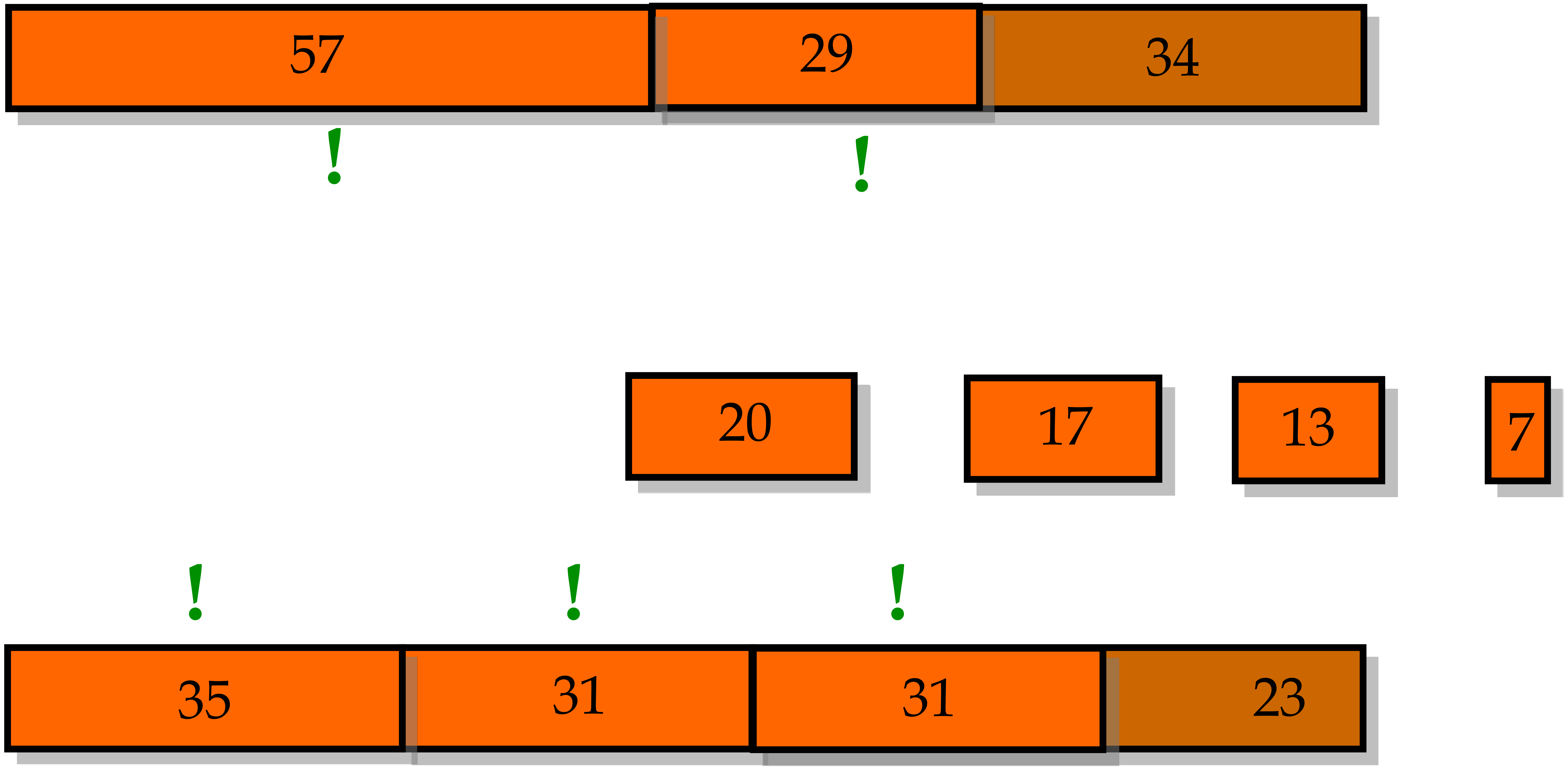
Keine Lösung?!



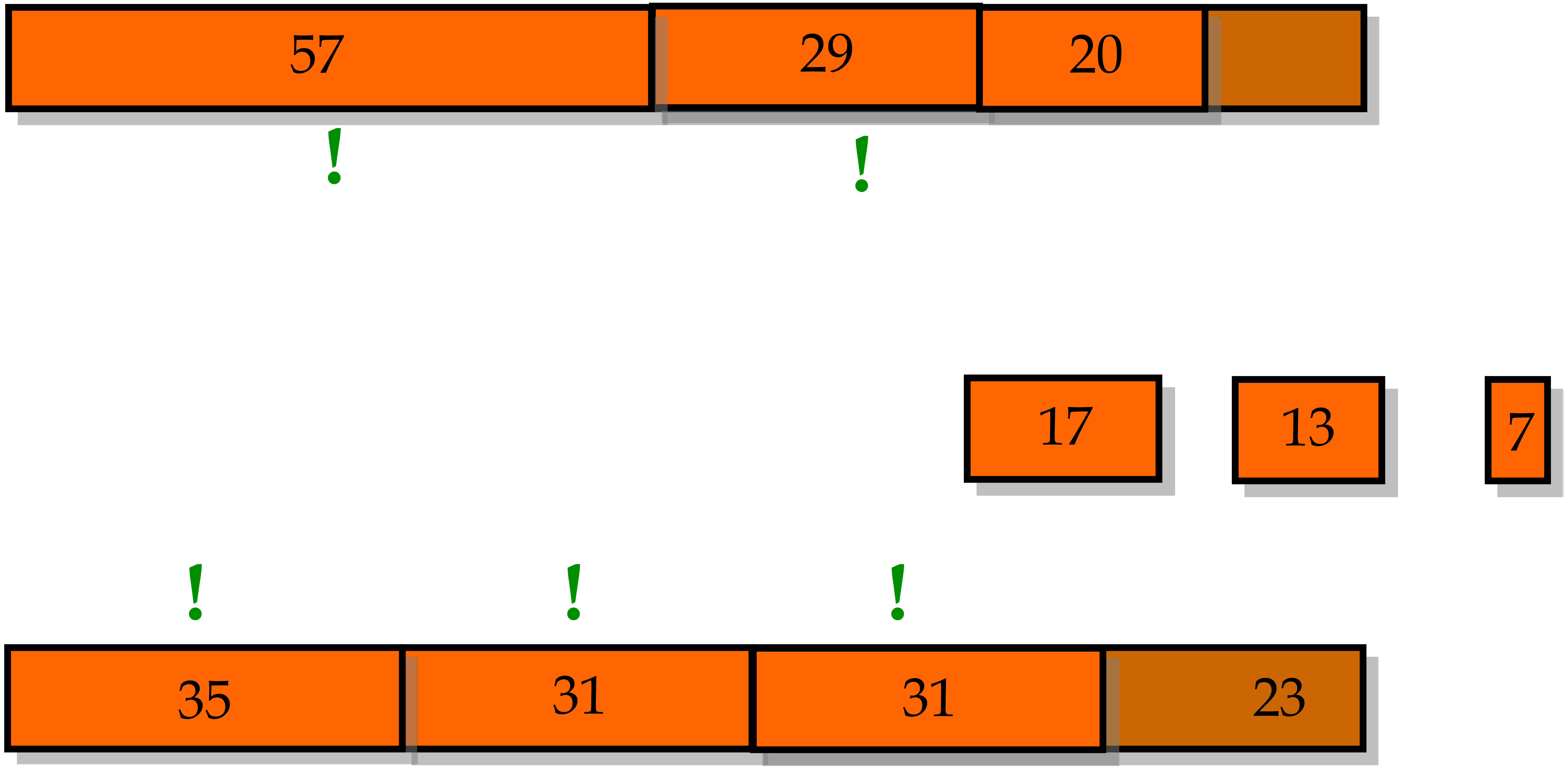
Keine Lösung?!



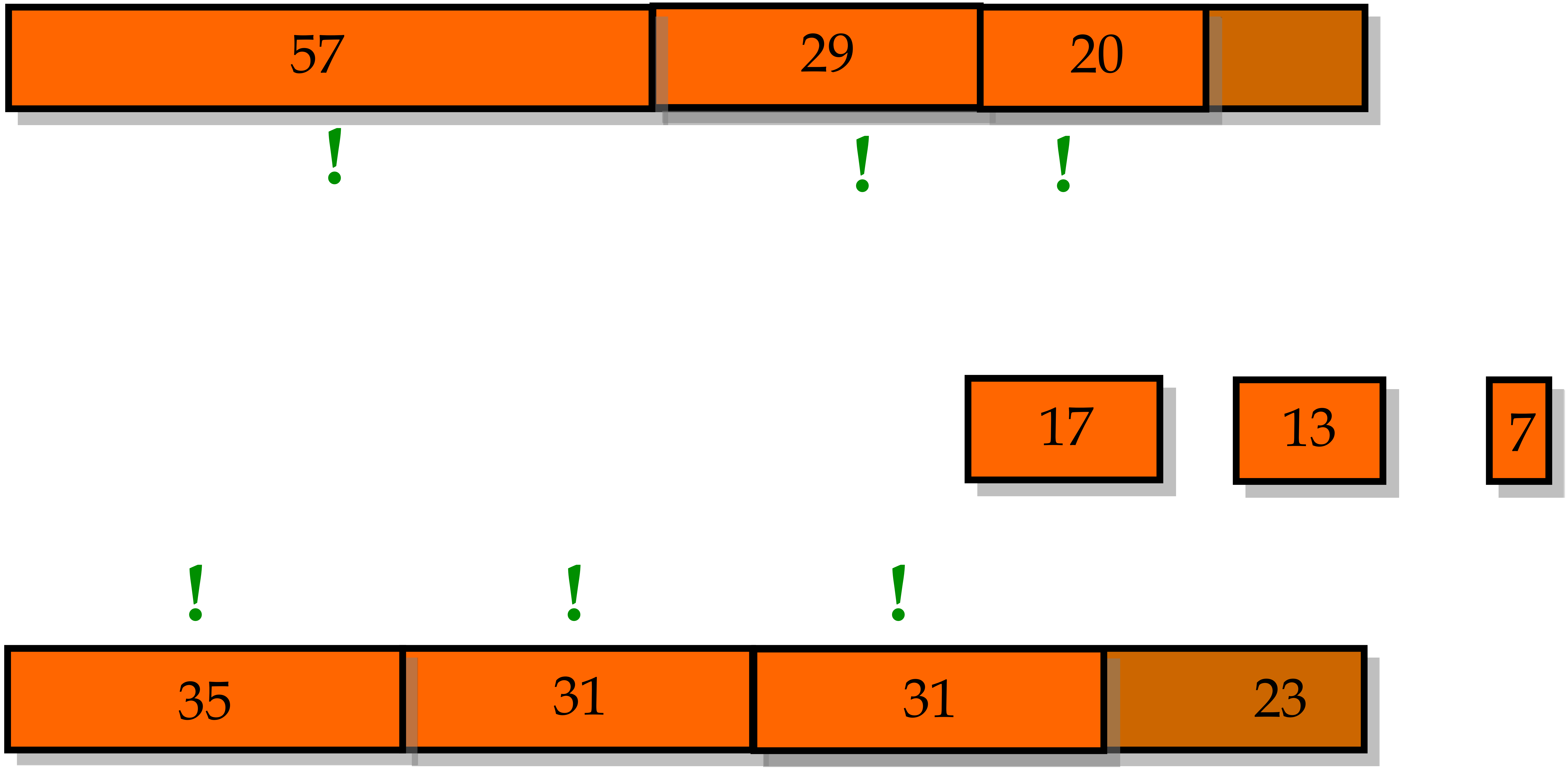
Keine Lösung?!



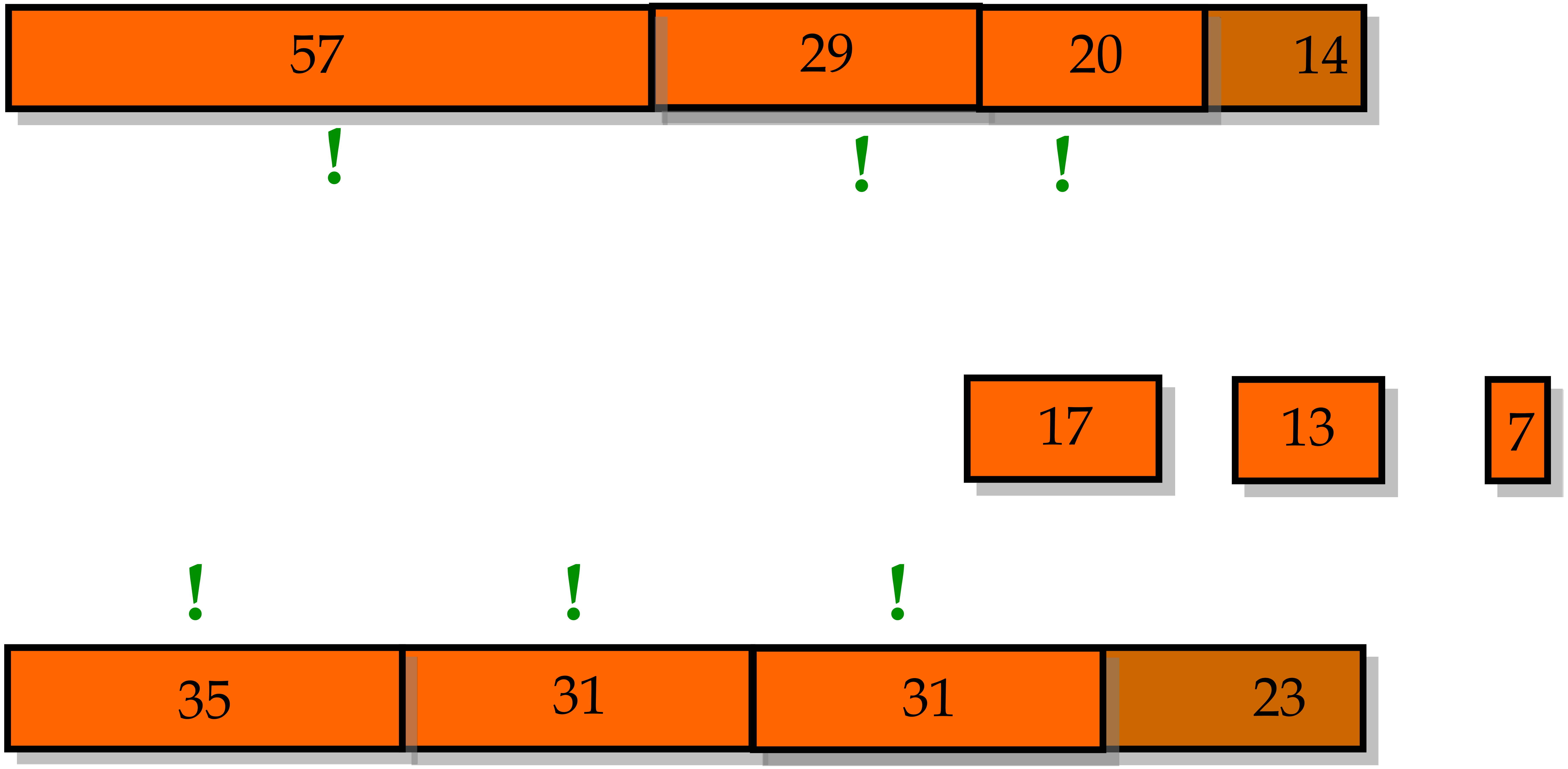
Keine Lösung?!



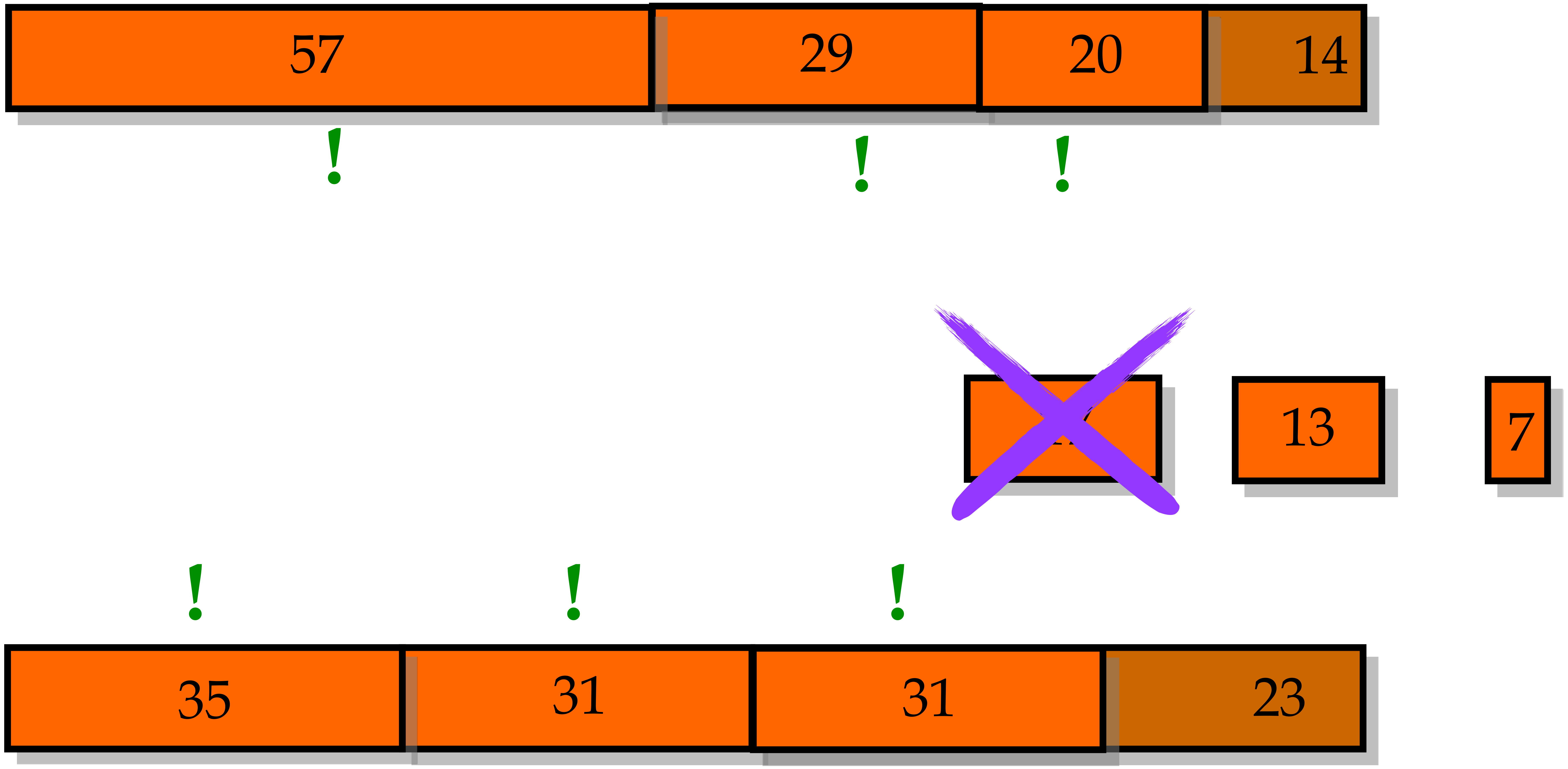
Keine Lösung?!



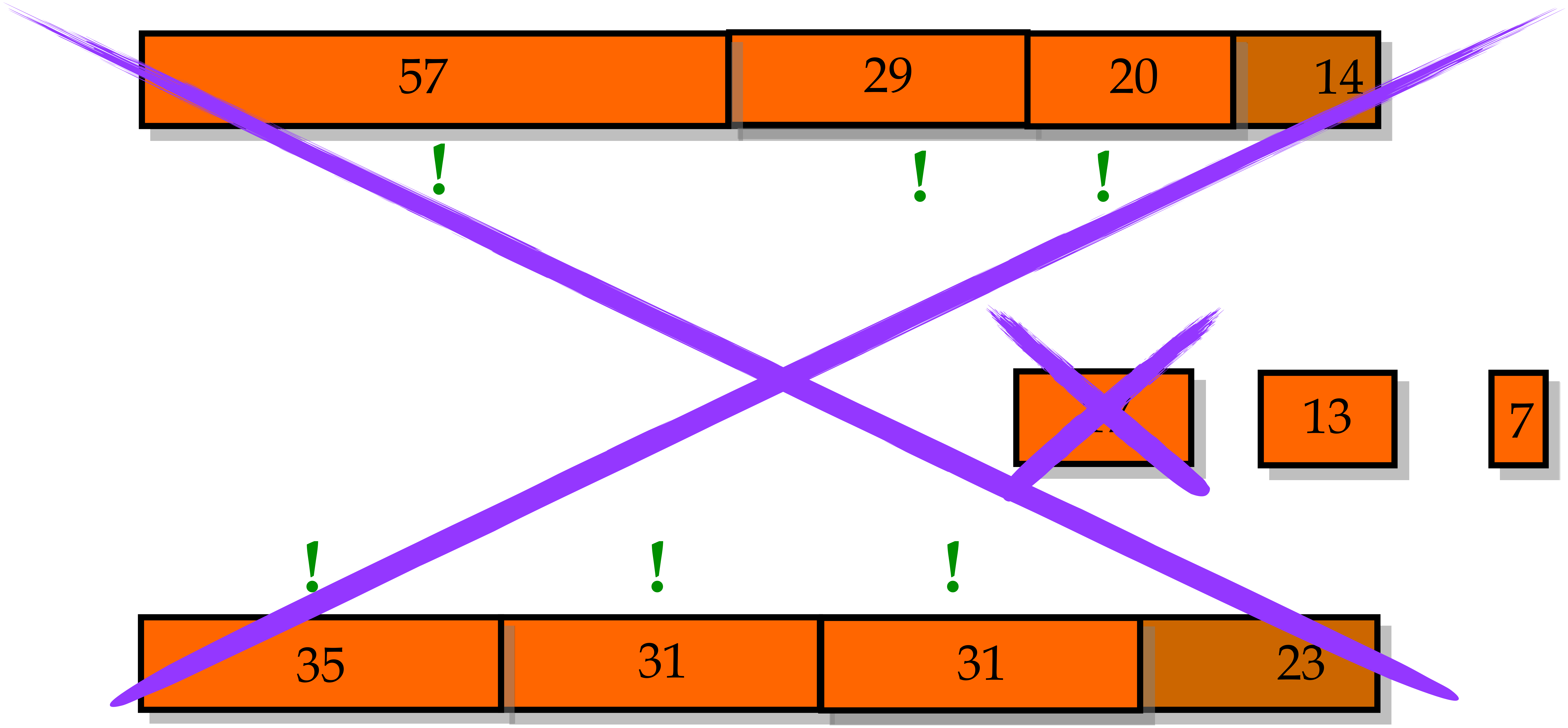
Keine Lösung?!



Keine Lösung?!

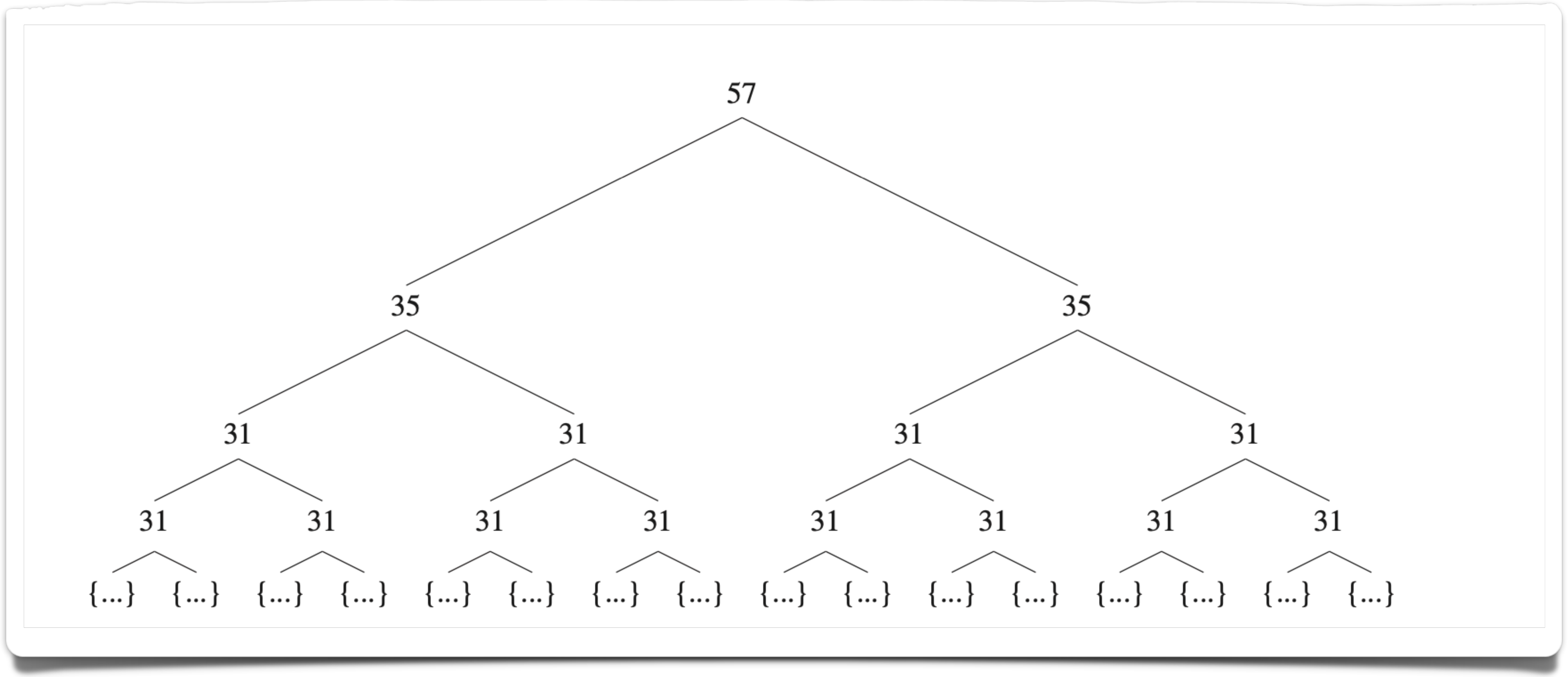


Keine Lösung?!

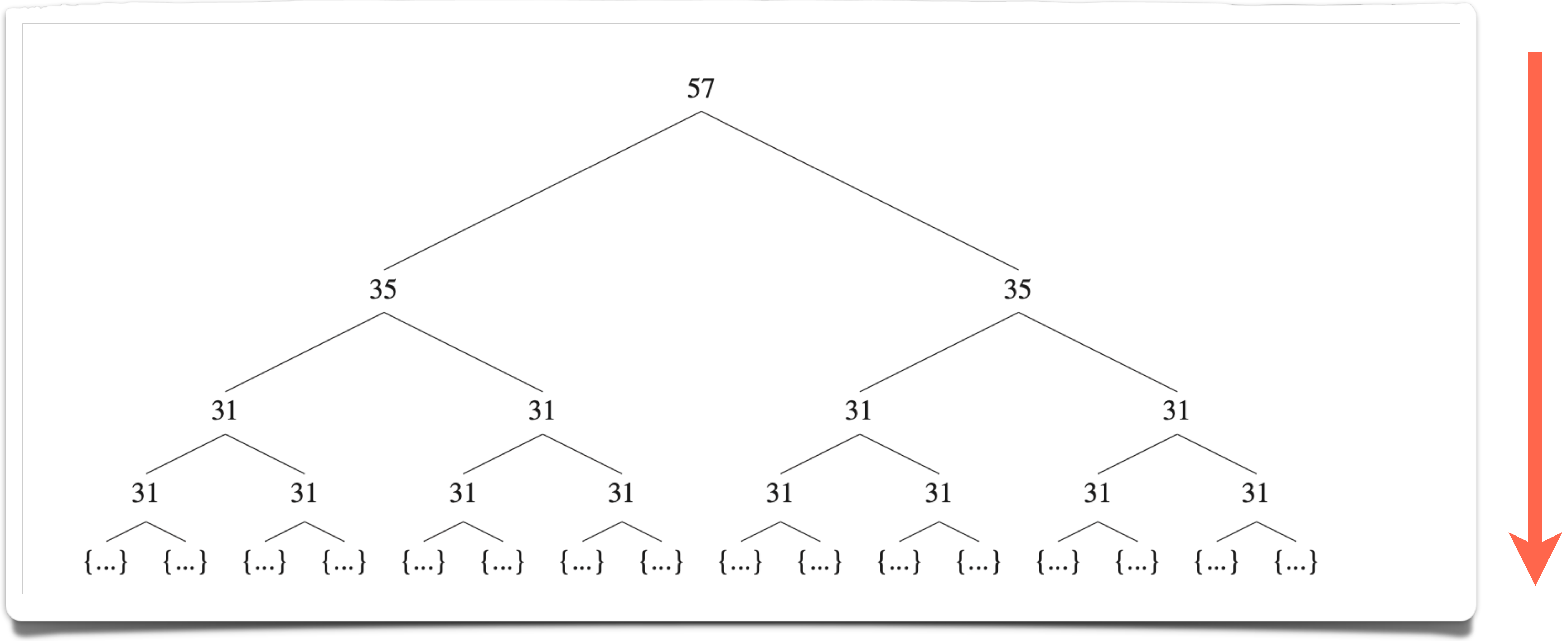


Ideen

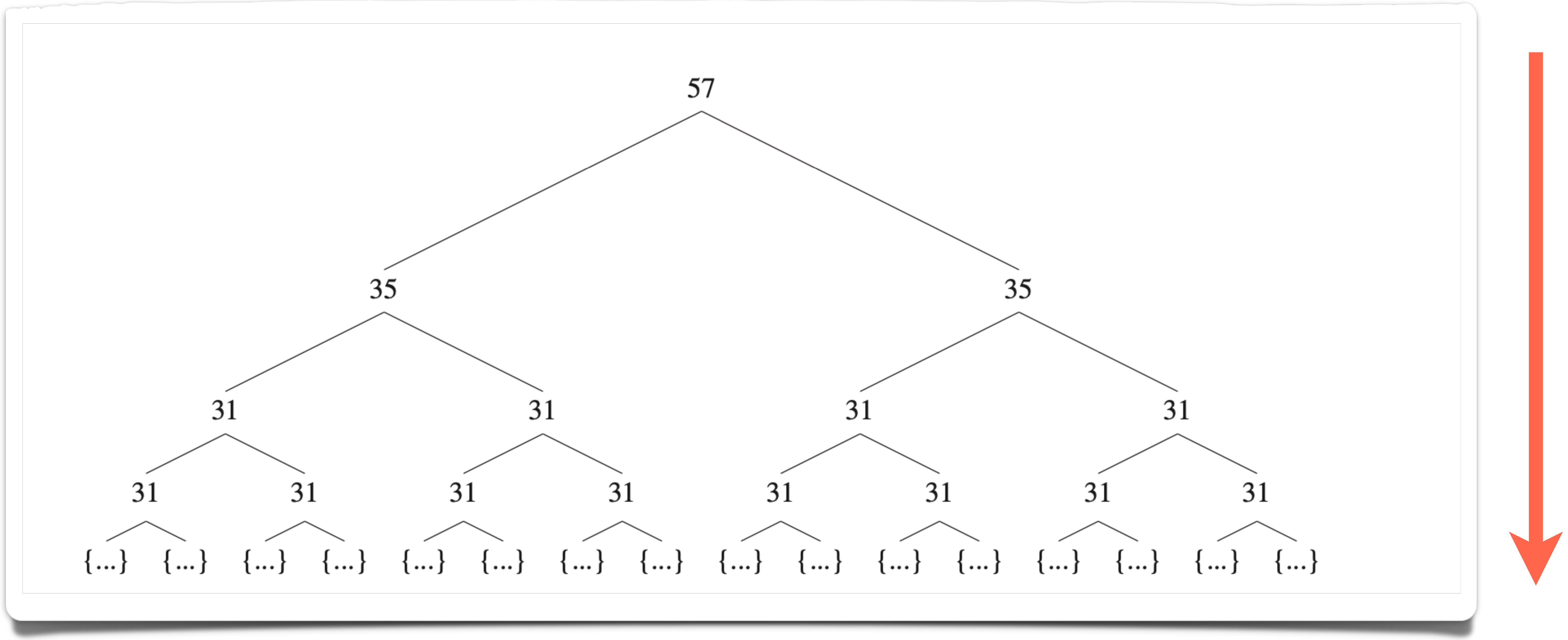
Ideen



Ideen

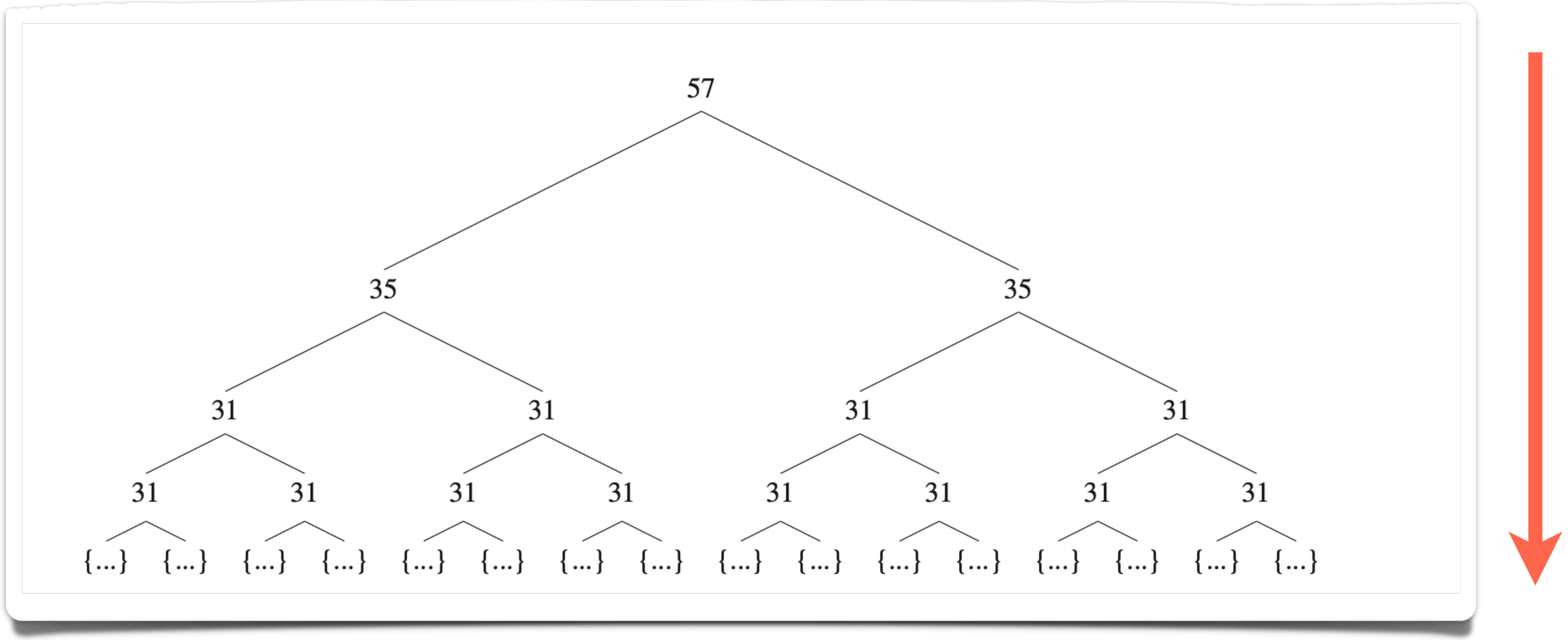


Ideen



- Zeige Unerreichbarkeit von Zahlen

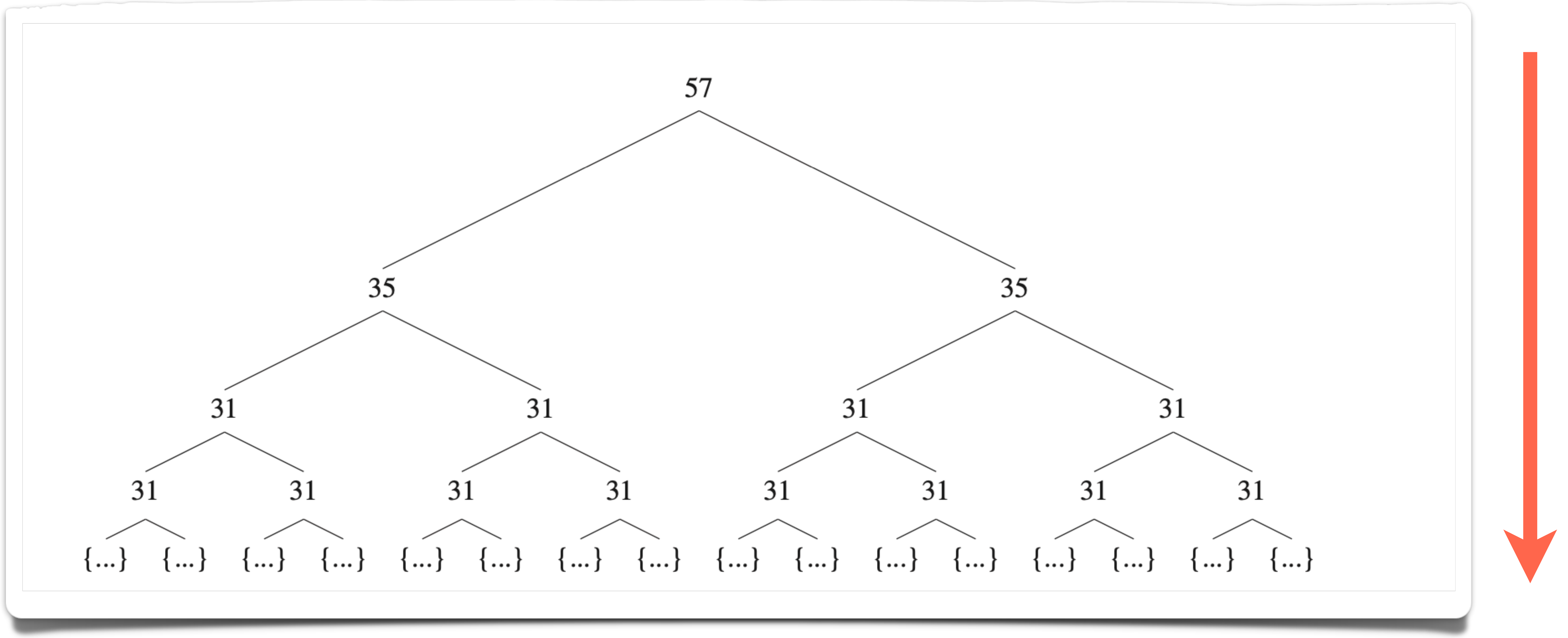
Ideen



- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: Schließe Zahlen aus

Ideen

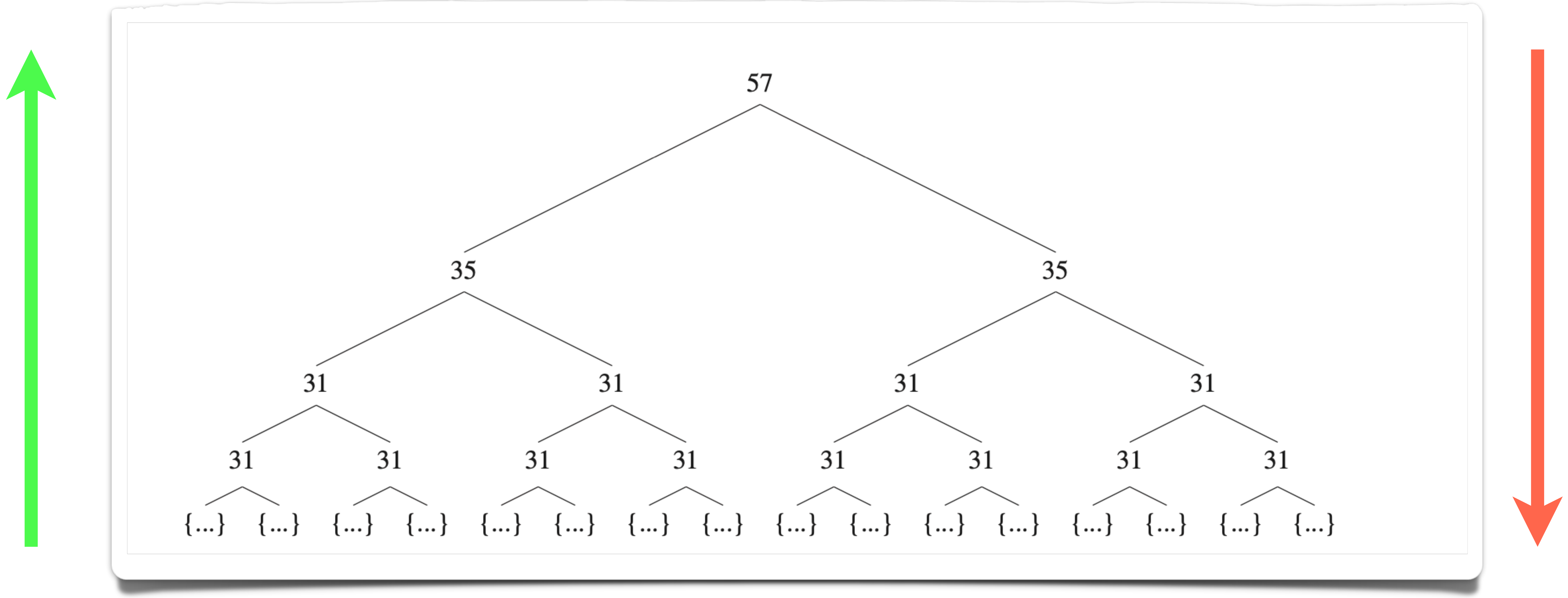
Dynamic Programming



- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: Schließe Zahlen aus

Ideen

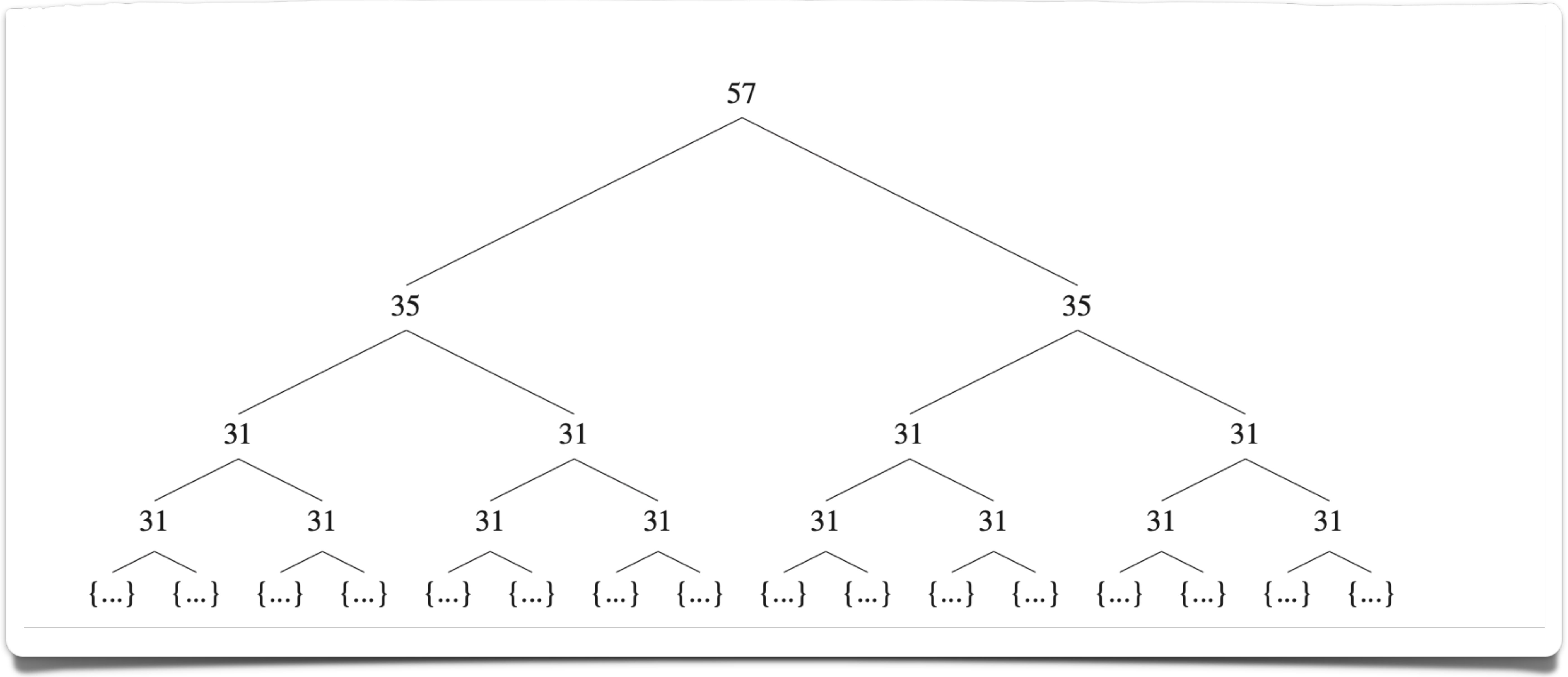
Dynamic Programming



- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: Schließe Zahlen aus

Ideen

Dynamic Programming

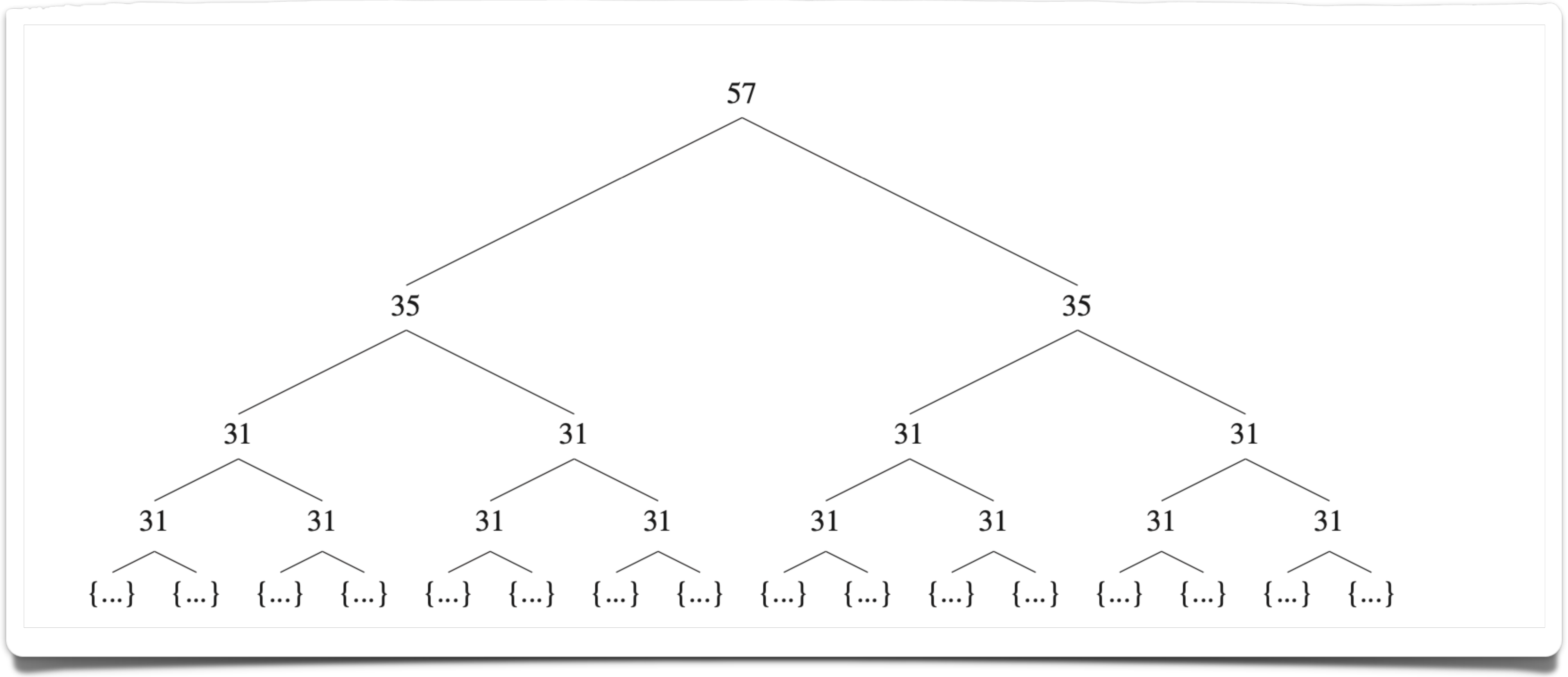


- Bestimme erreichbare Zahlen

- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: Schließe Zahlen aus

Ideen

Dynamic Programming



- Bestimme erreichbare Zahlen
- Vorwärts: Nimm Zahlen hinzu

- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: Schließe Zahlen aus

Beispiel

Beispiel

Beispiel 2.1

Beispiel 2.1

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

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!

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

Beispiel 2.1

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

Bestimme alle Zahlen, die als Teilsummen erreichbar sind!

- Bestimme erreichbare Zahlen
- Vorwärts: Nimm Zahlen hinzu

Beispiel

Beispiel 2.1

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

Bestimme alle Zahlen, die als Teilsummen erreichbar sind!

- Bestimme erreichbare Zahlen
- Vorwärts: Nimm Zahlen hinzu

Beispiel

Beispiel

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

Beispiel

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

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

Beispiel

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

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

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

Beispiel

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

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

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

Beispiel

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

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

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

Beispiel

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

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

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

Beispiel

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

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

$i \backslash x$	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																			
2																			
3																			

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	0	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	…	0
1																			
2																			
3																			

Beispiel

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

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

$i \backslash x$	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

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

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

$i \backslash x$	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																		
2																			
3																			

Beispiel

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

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

$i \backslash x$	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												
2																			
3																			

Beispiel

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

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

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												
2																			
3																			

+7

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	0	0	0	0	0 +7	0	0	…	0	0	0	0	0	0	0	0	…	0
1	1	0	0	0	0	0	0	1											
2																			
3																			

Beispiel

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

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

+7

$i \backslash x$	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																			

Beispiel

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

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

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																			

Beispiel

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

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

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																			

Beispiel

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

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

$i \backslash x$	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																			

Beispiel

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

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

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																			

Beispiel

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

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

$i \backslash x$	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																		
3																			

Beispiel

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

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

$i \backslash x$	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																		
3																			

Beispiel

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

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

$i \backslash x$	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							1											
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	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							1											
3																			

+13

Beispiel

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

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

$i \backslash x$	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							1		1									
3																			

+13

Beispiel

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

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

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
2	1							1		1									
3																			

+13

+13

Beispiel

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

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

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	1							1		1							1		
3																			

+13

+13

Beispiel

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

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

$i \backslash x$	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

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

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

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

Beispiel

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

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

$i \backslash x$	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																			

Beispiel

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

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

$i \backslash x$	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																			

Beispiel

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

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

$i \backslash x$	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																		

Beispiel

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

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

$i \backslash x$	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																		

Beispiel

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

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

$i \backslash x$	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							1											

Beispiel

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

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

$i \backslash x$	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							1											

Beispiel

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

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

$i \backslash x$	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							1		1									

Beispiel

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

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

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	1	0	0	0	0	0	0	1	…	1	0	0	0	0	0	0	1	…	0
3	1							1		1							1		

Beispiel

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

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

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	1	0	0	0	0	0	0	1	…	1	0	0	0	0	0	0	1	…	0
3	1							1		1							1		

Beispiel

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

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

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	1	0	0	0	0	0	0	1	…	1	0	0	0	0	0	0	1	…	0
3	1							1		1							1		

+17

Beispiel

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

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

$i \backslash x$	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							1		1				1			1		

+17

Beispiel

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

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

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	1	0	0	0	0	0	0	1	…	1	0	0	0	0	0	0	1	…	0
3	1							1		1				1			1		

+17

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	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							1		1				1			1	…	

+17

Beispiel

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

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

24

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							1		1				1			1	…	

+17

Beispiel

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

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

24

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							1		1				1			1	…	

+17

Beispiel

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

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

24 30

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							1		1				1			1	…	

+17

Beispiel

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

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

24 30

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							1		1				1			1	…	

+17

Beispiel

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

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

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							1		1				1			1	…	

+17

Beispiel

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

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

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	0	0	1	0	0	1	…	0

+17

Allgemeiner

Allgemeiner

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

Allgemeiner

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

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																			

Allgemeiner

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

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

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																			

Allgemeiner

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

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

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																			

Allgemeiner

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

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

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																			

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

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

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

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																			

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

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

Algorithmus

Algorithmus

Algorithmus 2.2. (Dynamic Programming für Subset Sum).

Algorithmus

Algorithmus 2.2 (Dynamic Programming für SUBSET SUM).

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

Algorithmus

Algorithmus 2.2 (Dynamic Programming für Subset Sum).

Eingabe: *Zahlen* $z_1, \ldots, z_n$ *mit* $\sum_{i=1}^{n} z_i = Z$
Ausgabe: *Boolesche Funktion* $\mathscr{S} : \{0, \ldots, Z\} \times \{0, \ldots, n\} \to \{0, 1\}$

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* $\mathscr{S} : \{0, ..., Z\} \times \{0, ..., n\} \rightarrow \{0, 1\}$

mit $\mathscr{S}(x, i) =$

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

$$\text{mit } \mathscr{S}(x, i) = \begin{cases} 1, \\ 0, \end{cases}$$

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

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

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* $\mathcal{S} : \{0, ..., Z\} \times \{0, ..., n\} \rightarrow \{0, 1\}$

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

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

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

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																			

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: $\mathscr{S}(0, 0) := 1$

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	[illegible]	0	1	…	1	0	0	0	0	0	0	1	…	0
3																			

Algorithmus 2.2 . (Dynamic Programming für SUBSET SUM).

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

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

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

1: $\mathcal{S}(0,0) := 1$

2: **for** $(x=1)$ to Z **do**

$i \backslash x$	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																			

Algorithmus

Algorithmus 2.2. (Dynamic Programming für Subset Sum).

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

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

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

1: $\mathcal{S}(0,0) := 1$

2: **for** $(x=1)$ to Z **do**

3: $\quad \mathcal{S}(x,0) := 0;$

$i \backslash x$	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																			

Algorithmus 2.2 . (Dynamic Programming für Subset Sum).

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

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

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

1: $\mathscr{S}(0,0) := 1$
2: **for** $(x=1)$ to Z **do**
3: $\quad\mathscr{S}(x,0) := 0;$

$i \backslash x$	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																			

Algorithmus

Algorithmus 2.2 . (Dynamic Programming für Subset Sum).

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

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

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

$\mathscr{S}(0,0) := 1$

for $(x=1)$ to Z **do**

  $\mathscr{S}(x,0) := 0;$

for $(i = 1)$ to n **do**

$i \backslash x$	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																			

Algorithmus

Algorithmus 2.2 . (Dynamic Programming für Subset Sum).

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

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

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

1: $\mathscr{S}(0,0) := 1$
2: **for** $(x{=}1)$ *to* Z **do**
3: $\quad\mathscr{S}(x,0) := 0;$
4: **for** $(i = 1)$ *to* n **do**

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																			

Algorithmus 2.2 . (Dynamic Programming für Subset Sum).

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

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

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

$i \backslash x$	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																			

$1{:}\ \mathcal{S}(0,0) := 1$
$2{:}\ \textbf{for } (x{=}1) \text{ to } Z\ \textbf{do}$
$3{:}\qquad \mathcal{S}(x,0) := 0;$
$4{:}\ \textbf{for } (i = 1) \text{ to } n\ \textbf{do}$
$5{:}\qquad \textbf{for } (x = 0) \text{ to } z_i - 1\ \textbf{do}$

Algorithmus

Algorithmus 2.2. (Dynamic Programming für Subset Sum).

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

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

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

$\mathscr{S}(0,0) := 1$

for $(x=1)$ to Z **do**

  $\mathscr{S}(x,0) := 0;$

for $(i=1)$ to n **do**

  for $(x=0)$ to $z_i - 1$ **do**

    $\mathscr{S}(x,i) := \mathscr{S}(x,i-1);$

$i \backslash x$	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																			

Algorithmus 2.2 . (Dynamic Programming für SUBSET SUM).

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

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

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

1: $\mathscr{S}(0,0) := 1$
2: **for** (x=1) to Z **do**
3: $\quad \mathscr{S}(x,0) := 0;$
4: **for** $(i = 1)$ to n **do**
5: $\quad$ **for** $(x = 0)$ to $z_i - 1$ **do**
6: $\quad\quad \mathscr{S}(x,i) := \mathscr{S}(x,i-1);$
7: $\quad$ **for** $(x = z_i)$ to Z **do**

$i \backslash x$	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																			

Algorithmus

Algorithmus 2.2. (Dynamic Programming für SUBSET SUM).

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

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

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

1: $\mathscr{S}(0,0) := 1$
2: **for** (x=1) to Z **do**
3: $\quad\mathscr{S}(x,0) := 0;$
4: **for** $(i = 1)$ to n **do**
5: $\quad$**for** $(x = 0)$ to $z_i - 1$ **do**
6: $\quad\quad\mathscr{S}(x,i) := \mathscr{S}(x,i-1);$
7: $\quad$**for** $(x = z_i)$ to Z **do**
8: $\quad\quad$**if** $((\mathscr{S}(x,i-1) = 1)$ OR $(\mathscr{S}((x-z_i),i-1) = 1))$ **then**

$i \backslash x$	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																			

Technische Universität Braunschweig

Algorithmus

Algorithmus 2.2. (Dynamic Programming für SUBSET SUM).

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

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

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

```
𝒮(0,0) := 1
for (x=1) to Z do
    𝒮(x,0) := 0;
for (i = 1) to n do
    for (x = 0) to z_i − 1 do
        𝒮(x,i) := 𝒮(x,i−1);
    for (x = z_i) to Z do
        if ((𝒮(x,i−1) = 1) OR (𝒮((x−z_i),i−1) = 1)) then
```

$i \backslash x$	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																			

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
```

$i \backslash x$	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																			

Algorithmus

Algorithmus 2.2 . (Dynamic Programming für SUBSET SUM).

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

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

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

$\mathscr{S}(0,0) := 1$
for $(x=1)$ to Z **do**
  $\mathscr{S}(x,0) := 0;$

for $(i = 1)$ to n **do**
  for $(x = 0)$ to $z_i - 1$ **do**
    $\mathscr{S}(x,i) := \mathscr{S}(x,i-1);$
  for $(x = z_i)$ to Z **do**
    if $(\boxed{\mathscr{S}(x,i-1) = 1}$ OR $(\boxed{\mathscr{S}((x-z_i),i-1) = 1})$ **then**
      $\mathscr{S}(x,i) := 1;$

$i \backslash x$	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																			

Algorithmus

Algorithmus 2.2 . (Dynamic Programming für Subset Sum).

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

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

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

$\mathscr{S}(0,0) := 1$
for $(x{=}1)$ *to* Z **do**
  $\mathscr{S}(x,0) := 0;$

for $(i = 1)$ *to* n **do**
  for $(x = 0)$ *to* $z_i - 1$ **do**
    $\mathscr{S}(x,i) := \mathscr{S}(x, i-1);$
  for $(x = z_i)$ *to* Z **do**
    if $(\boxed{\mathscr{S}(x, i-1) = 1}$ *OR* $(\boxed{\mathscr{S}((x - z_i), i-1) = 1})$ **then**
      $\mathscr{S}(x,i) := 1;$
    else
      $\mathscr{S}(x,i) := 0;$

$i \backslash x$	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																			

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* $\mathscr{S} : \{0, ..., Z\} \times \{0, ..., n\} \rightarrow \{0, 1\}$

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

$\mathscr{S}(0,0) := 1$
for (x=1) to Z **do**
  $\mathscr{S}(x,0) := 0;$
for $(i = 1)$ to n **do**
  for $(x = 0)$ to $z_i - 1$ **do**
    $\mathscr{S}(x,i) := \mathscr{S}(x,i-1);$
  for $(x = z_i)$ to Z **do**
    if $(\boxed{\mathscr{S}(x,i-1) = 1}$ *OR* $(\boxed{\mathscr{S}((x-z_i),i-1) = 1}))$ **then**
      $\mathscr{S}(x,i) := 1;$
    else
      $\mathscr{S}(x,i) := 0;$
return

$i \backslash x$	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																			

Korrektheit

Korrektheit

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

Korrektheit

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

Beweis.

Korrektheit

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

Beweis.

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																			

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.

$i \backslash x$	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																			

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 $n=0$.

$i \backslash x$	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																			

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 $n=0$.

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																			

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

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																			

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

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																			

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

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

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																			

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

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

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																			

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

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

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																			

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

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

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																			

(I) z_i wird nicht verwendet

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

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

(I) z_i wird nicht verwendet

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																			

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 $n=0$.

$i \backslash x$	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																			

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

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

(I) z_i wird nicht verwendet

(II) z_i wird verwendet

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.

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																			

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

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

(I) z_i wird nicht verwendet

(II) z_i wird verwendet

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 $z_1, \ldots, z_i$.

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

(II) z_i wird verwendet

$i \backslash x$	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																			

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

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

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							1											
3																			

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

(II) z_i wird verwendet

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

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

$i \backslash x$	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							1											
3																			

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

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

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: $\mathscr{S}(x, i-1)$ *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.

$i \backslash x$	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
2	1							1		1							1		
3																			

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

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

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

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

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																			

Laufzeit

Laufzeit

Algorithmus 2.2 . (Dynamic Programming für SUBSET SUM).

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

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

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

```
𝒮(0,0) := 1
for (x=1) to Z do
    𝒮(x,0) := 0;
for (i = 1) to n do
    for (x = 0) to z_i − 1 do
        𝒮(x,i) := 𝒮(x,i−1);
    for (x = z_i) to Z do
        if ((𝒮(x,i−1) = 1) OR (𝒮((x−z_i),i−1) = 1)) then
            𝒮(x,i) := 1;
        else
            𝒮(x,i) := 0;
return
```

Laufzeit

Algorithmus 2.2 . (Dynamic Programming für Subset Sum).

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

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

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

```
S(0,0) := 1
for (x=1) to Z do
    S(x,0) := 0;
for (i = 1) to n do
    for (x = 0) to z_i − 1 do
        S(x,i) := S(x,i − 1);
    for (x = z_i) to Z do
        if ((S(x,i − 1) = 1) OR (S((x − z_i),i − 1) = 1)) then
            S(x,i) := 1;
        else
            S(x,i) := 0;
return
```

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* $\mathscr{S} : \{0, ..., Z\} \times \{0, ..., n\} \rightarrow \{0, 1\}$

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

$\mathscr{S}(0,0) := 1$
for (x=1) to Z **do**
    $\mathscr{S}(x,0) := 0;$
for $(i = 1)$ to n **do**
    for $(x = 0)$ to $z_i - 1$ **do**
        $\mathscr{S}(x,i) := \mathscr{S}(x,i-1);$
    for $(x = z_i)$ to Z **do**
        if $((\mathscr{S}(x,i-1) = 1)$ *OR* $(\mathscr{S}((x-z_i),i-1) = 1))$ **then**
            $\mathscr{S}(x,i) := 1;$
        else
            $\mathscr{S}(x,i) := 0;$
return

Laufzeit

Algorithmus 2.2 . (Dynamic Programming für Subset Sum).

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

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

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

$\mathscr{S}(0,0) := 1$
for $(x=1)$ *to* Z **do**
    $\mathscr{S}(x,0) := 0;$
for $(i = 1)$ *to* n **do**
    for $(x = 0)$ *to* $z_i - 1$ **do**
        $\mathscr{S}(x,i) := \mathscr{S}(x,i-1);$
    for $(x = z_i)$ *to* Z **do**
        if $((\mathscr{S}(x,i-1) = 1)$ *OR* $(\mathscr{S}((x-z_i),i-1) = 1))$ **then**
            $\mathscr{S}(x,i) := 1;$
        else
            $\mathscr{S}(x,i) := 0;$
return

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

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

$\mathscr{S}(0,0) := 1$
for *(x=1) to Z* ***do***
    $\mathscr{S}(x,0) := 0;$
for $(i = 1)$ *to n* ***do***
    for $(x = 0)$ *to* $z_i - 1$ ***do***
        $\mathscr{S}(x,i) := \mathscr{S}(x,i-1);$
    for $(x = z_i)$ *to Z* ***do***
        if $((\mathscr{S}(x,i-1) = 1)$ *OR* $(\mathscr{S}((x-z_i),i-1) = 1))$ ***then***
            $\mathscr{S}(x,i) := 1;$
        else
            $\mathscr{S}(x,i) := 0;$
return

Laufzeit

Algorithmus 2.2 . (Dynamic Programming für Subset Sum).

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

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

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

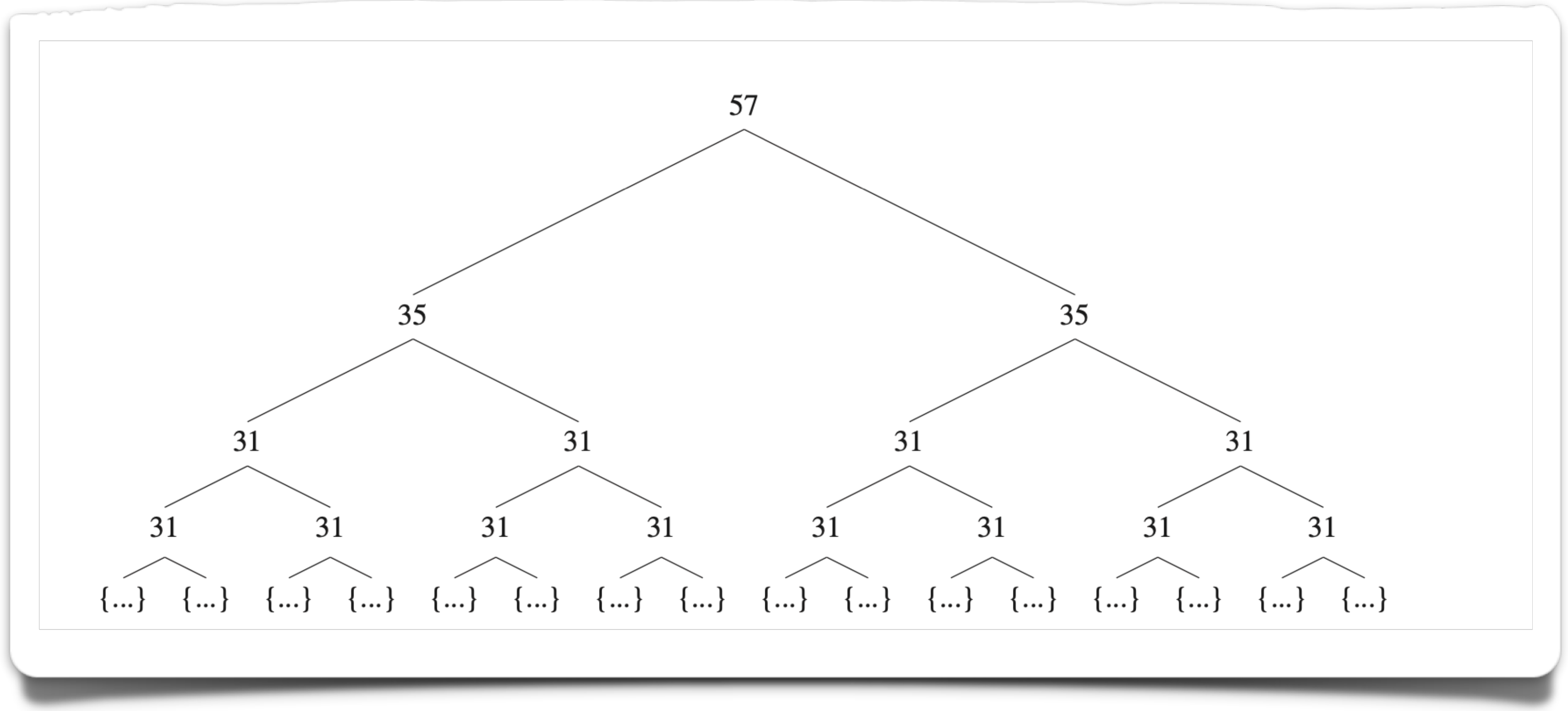
$\mathscr{S}(0,0) := 1$
for $(x=1)$ to Z **do**
    $\mathscr{S}(x,0) := 0;$

$O(nZ)$

for $(i = 1)$ *to* n **do**
    for $(x = 0)$ *to* $z_i - 1$ **do**
        $\mathscr{S}(x,i) := \mathscr{S}(x,i-1);$
    for $(x = z_i)$ *to* Z **do**
        if $((\mathscr{S}(x,i-1) = 1)$ *OR* $(\mathscr{S}((x-z_i),i-1) = 1))$ **then**
            $\mathscr{S}(x,i) := 1;$
        else
            $\mathscr{S}(x,i) := 0;$
return

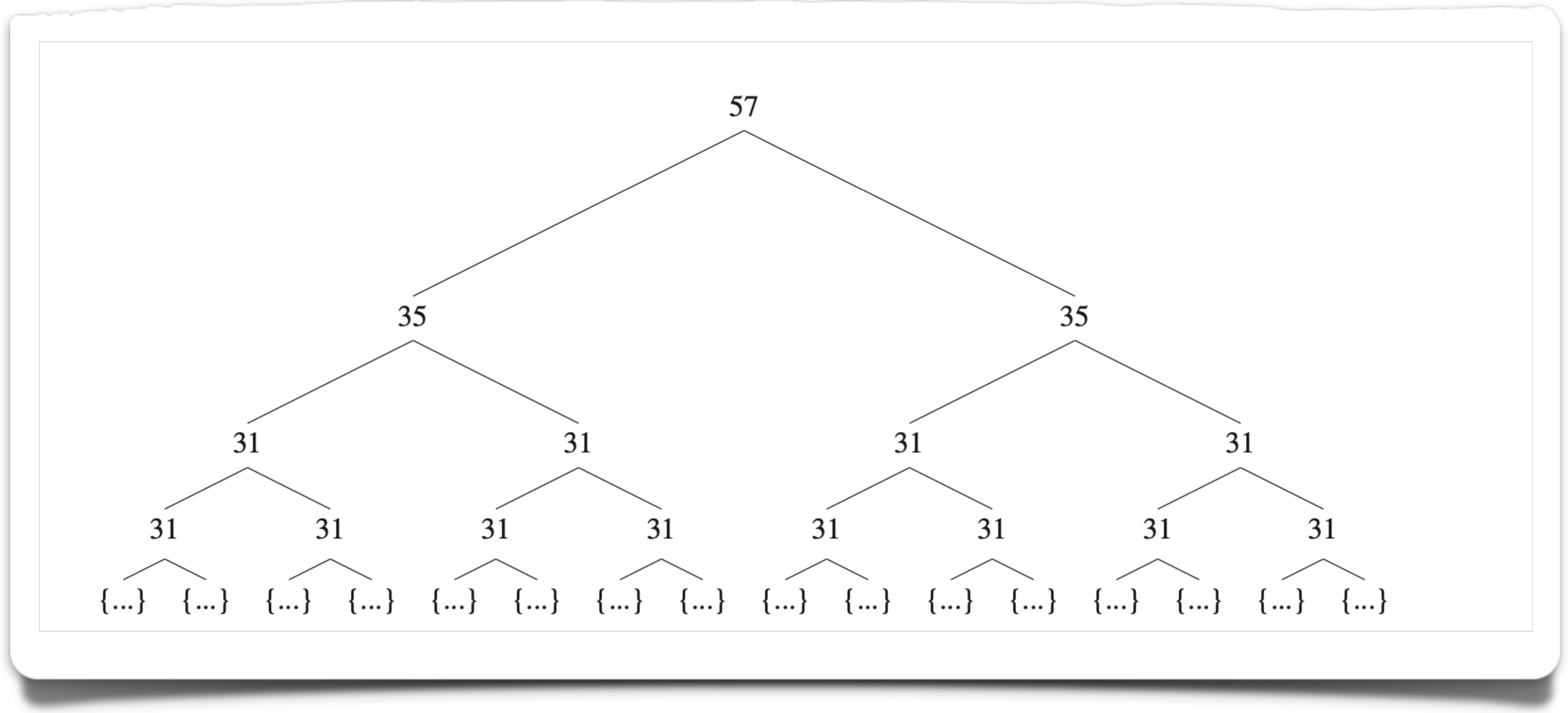
Effizienz?!

Effizienz?!



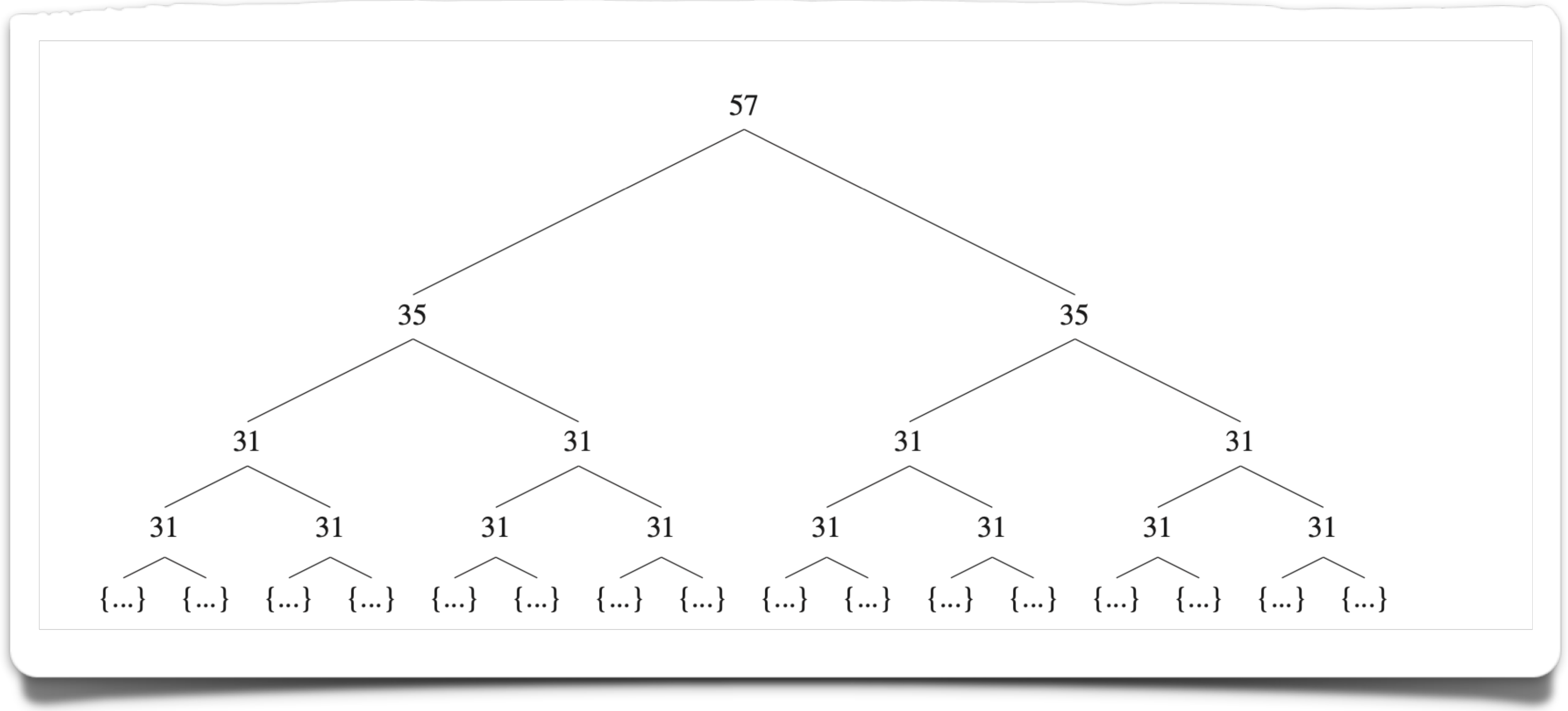
Effizienz?!

$$2^n$$



Effizienz?!

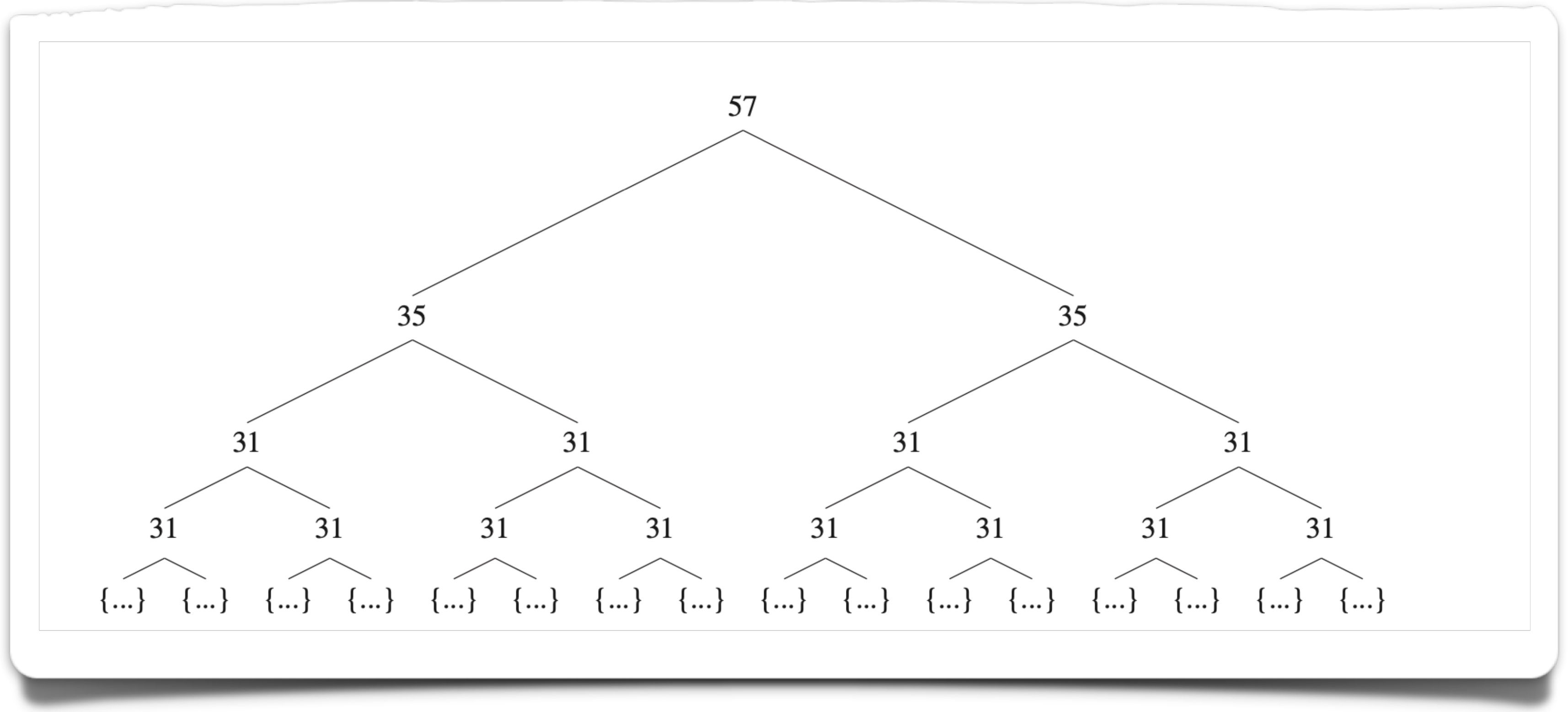
$$2^n$$



- Exponentiell viele Fälle!

Effizienz?!

$$2^n$$

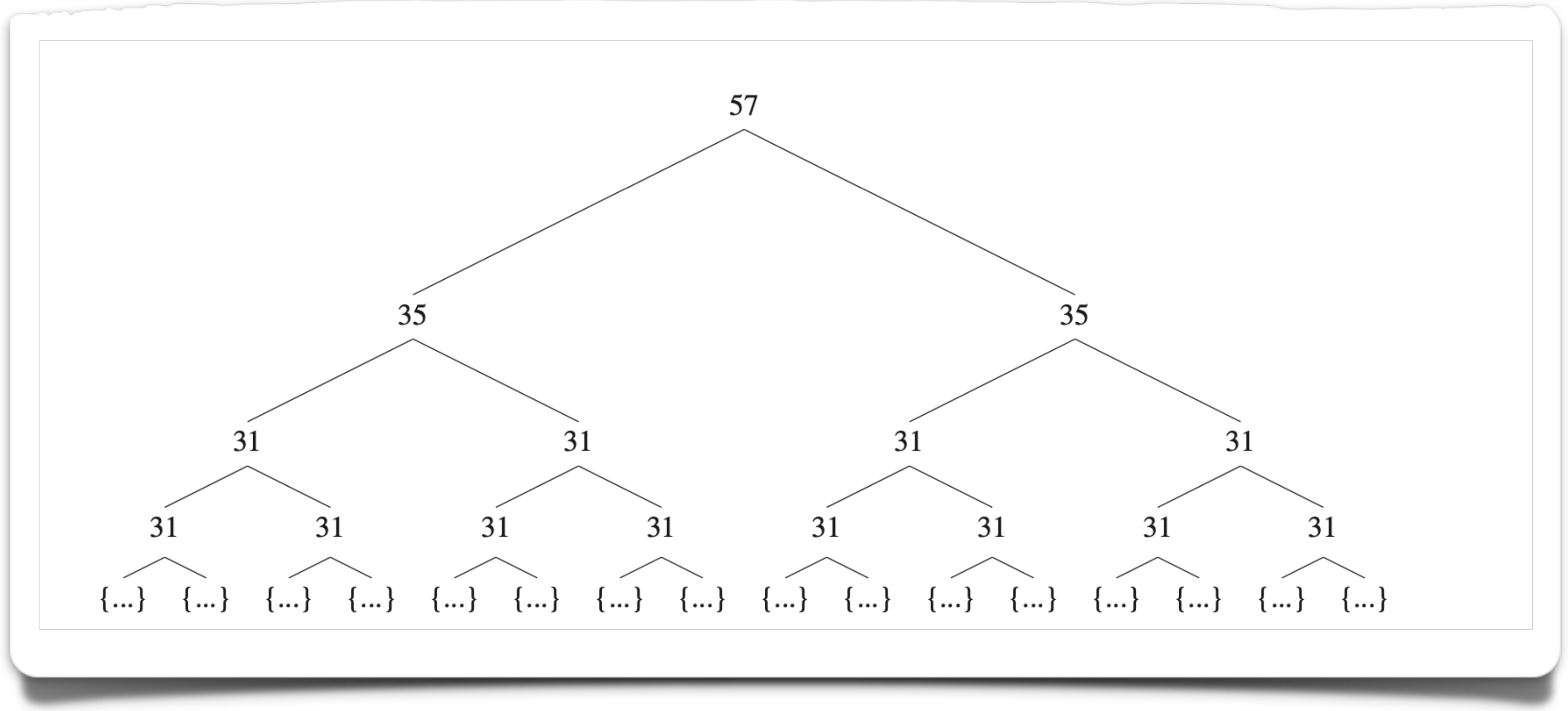


- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

$O(nZ)$

2^n



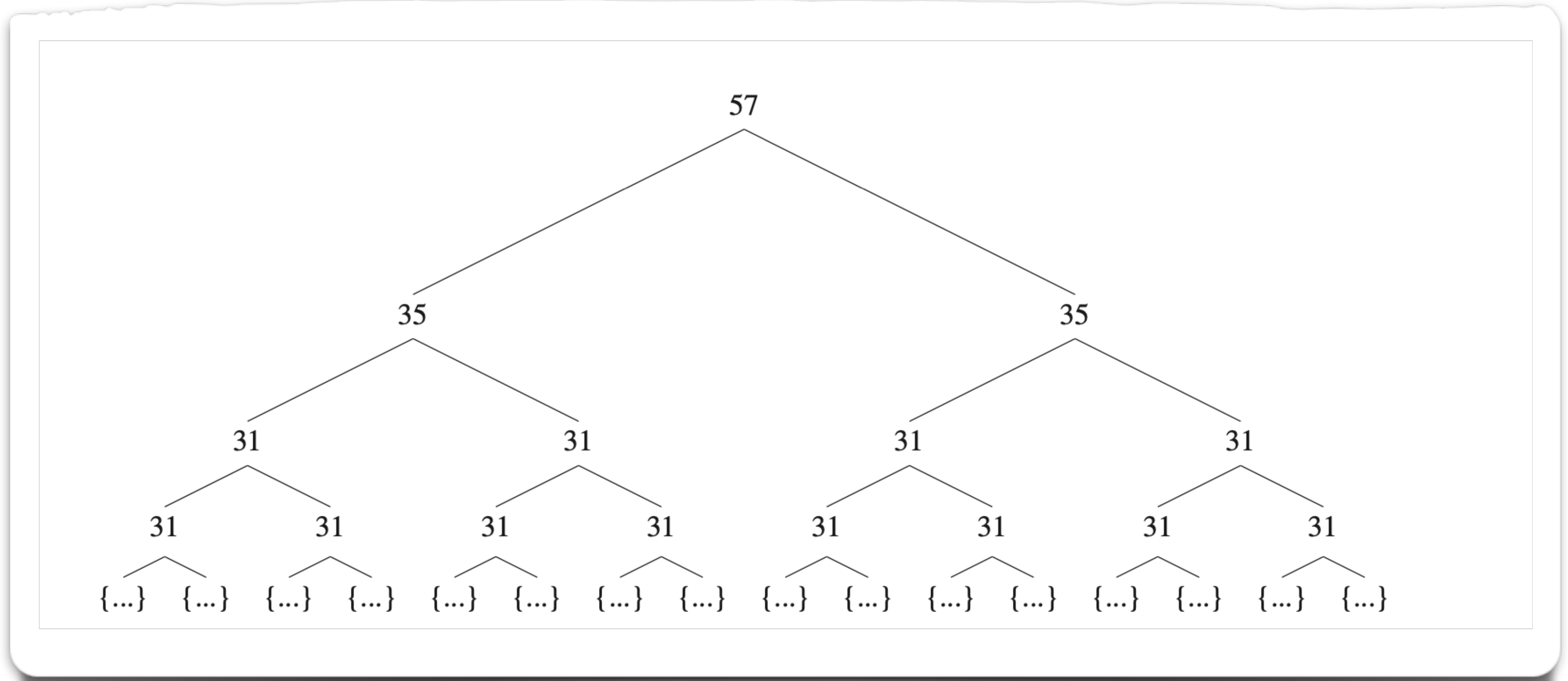
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

$O(nZ)$

$i \backslash x$	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



- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

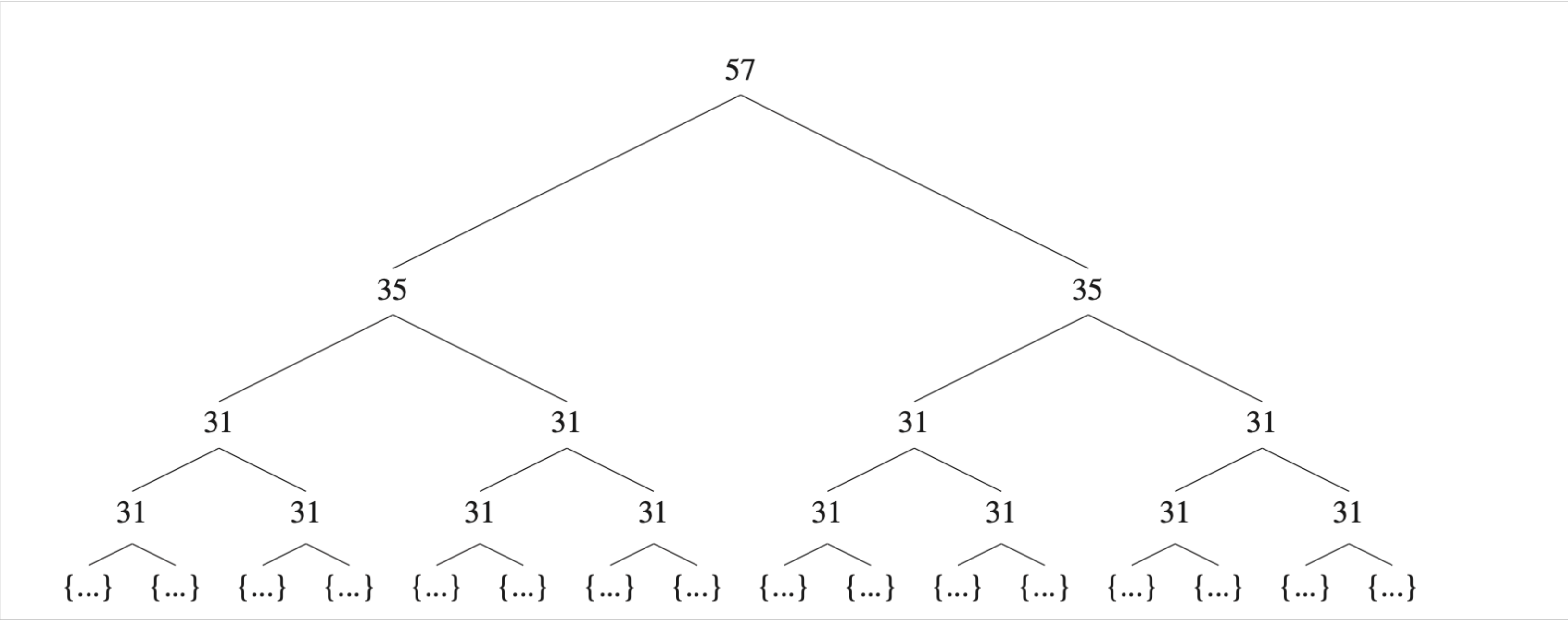
Effizienz?!

$O(nZ)$

$i \backslash x$	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																			

1

2^n



- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

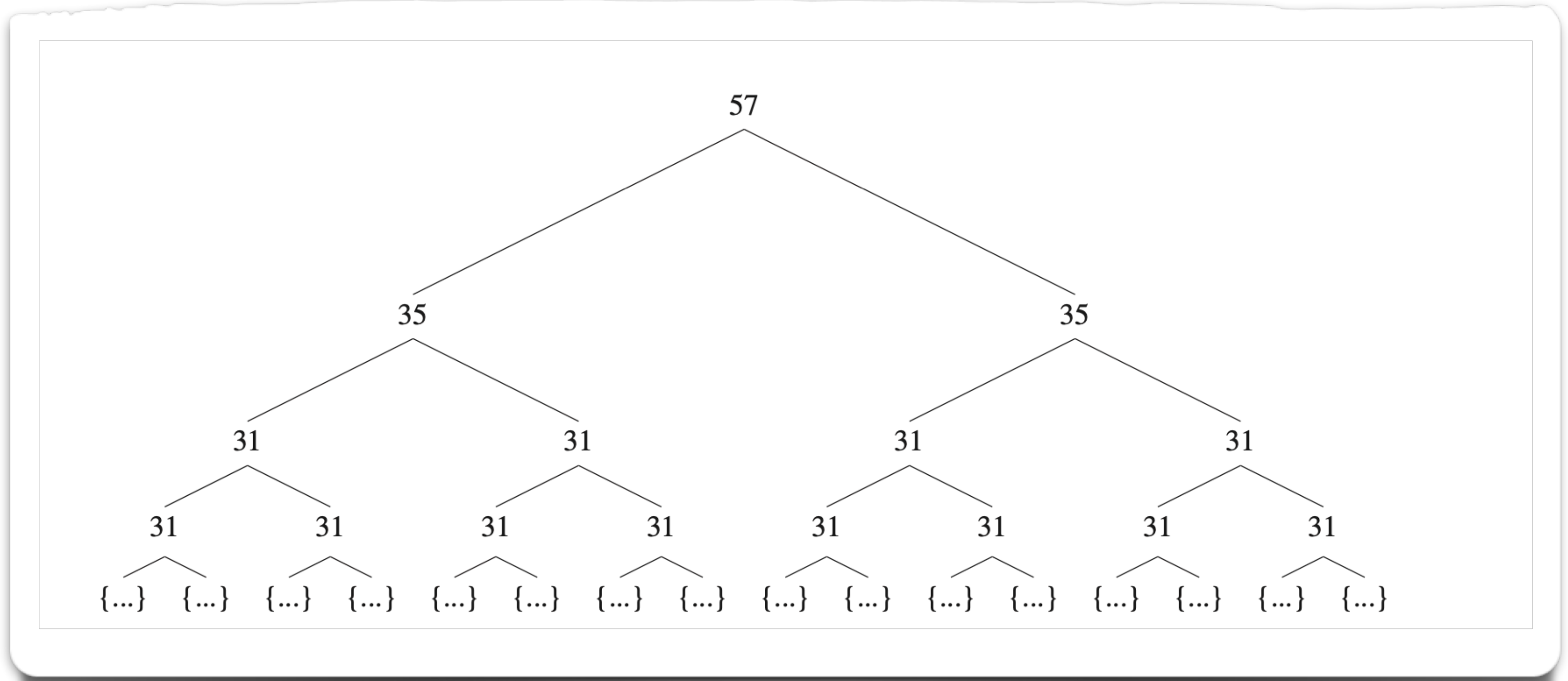
Effizienz?!

$O(nZ)$

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											
2																			
3																			

+7



1

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

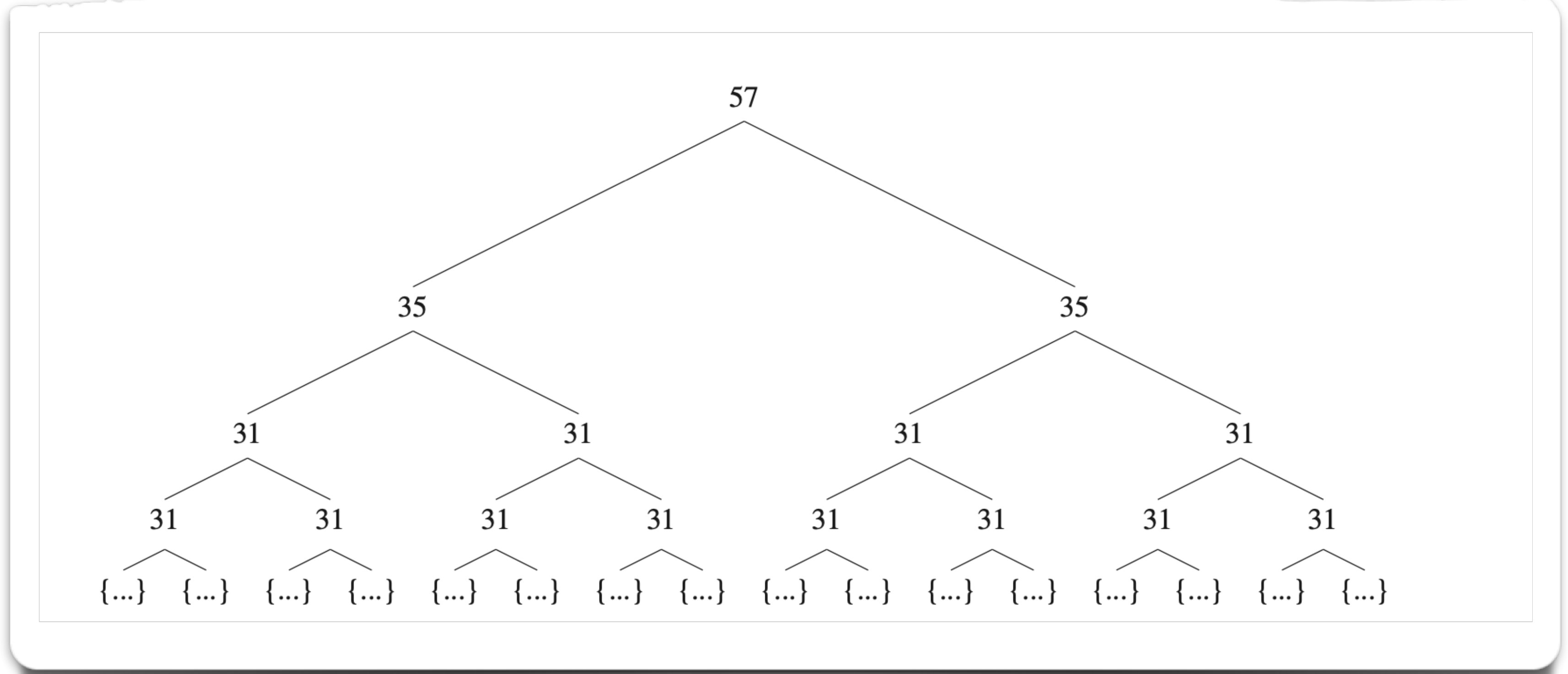
Effizienz?!

$O(nZ)$

2^n

+7

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											
2																			
3																			



2

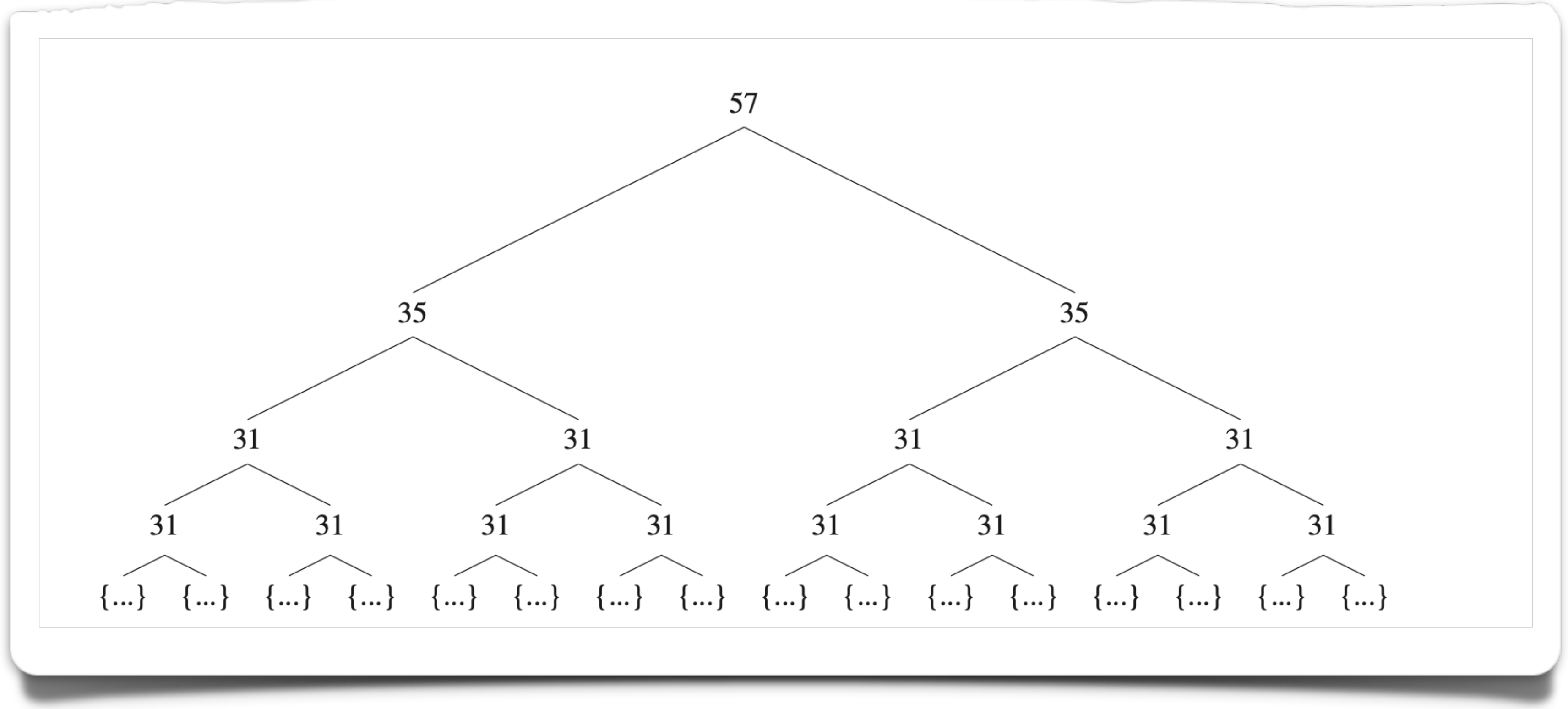
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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	+13	0	0	0	0	…	0
2	1	0	0	0	+13		0	1	…	1	0	0	0	0	0	0	1	…	0
3																			



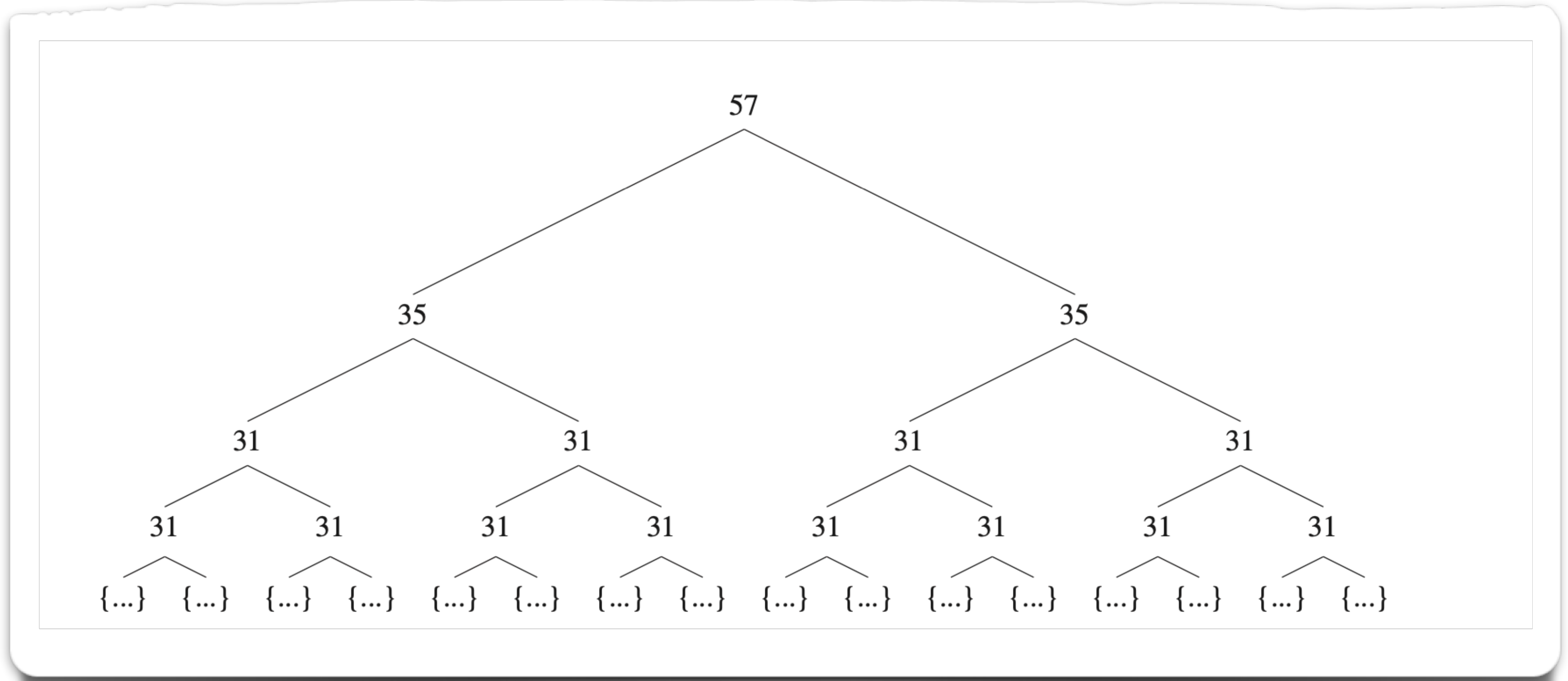
2

- 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	+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

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

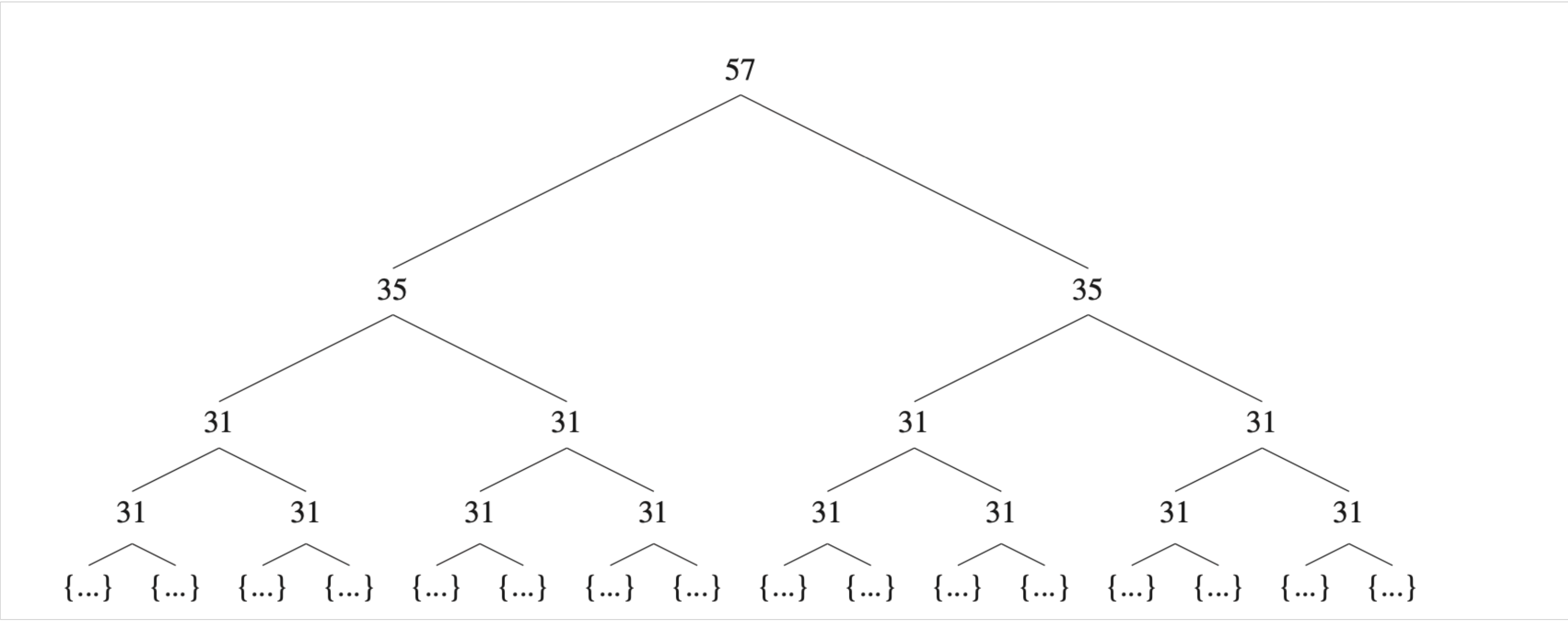
$O(nZ)$

2^n

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	0	0	1	0	0	1	…	0

+17



4

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

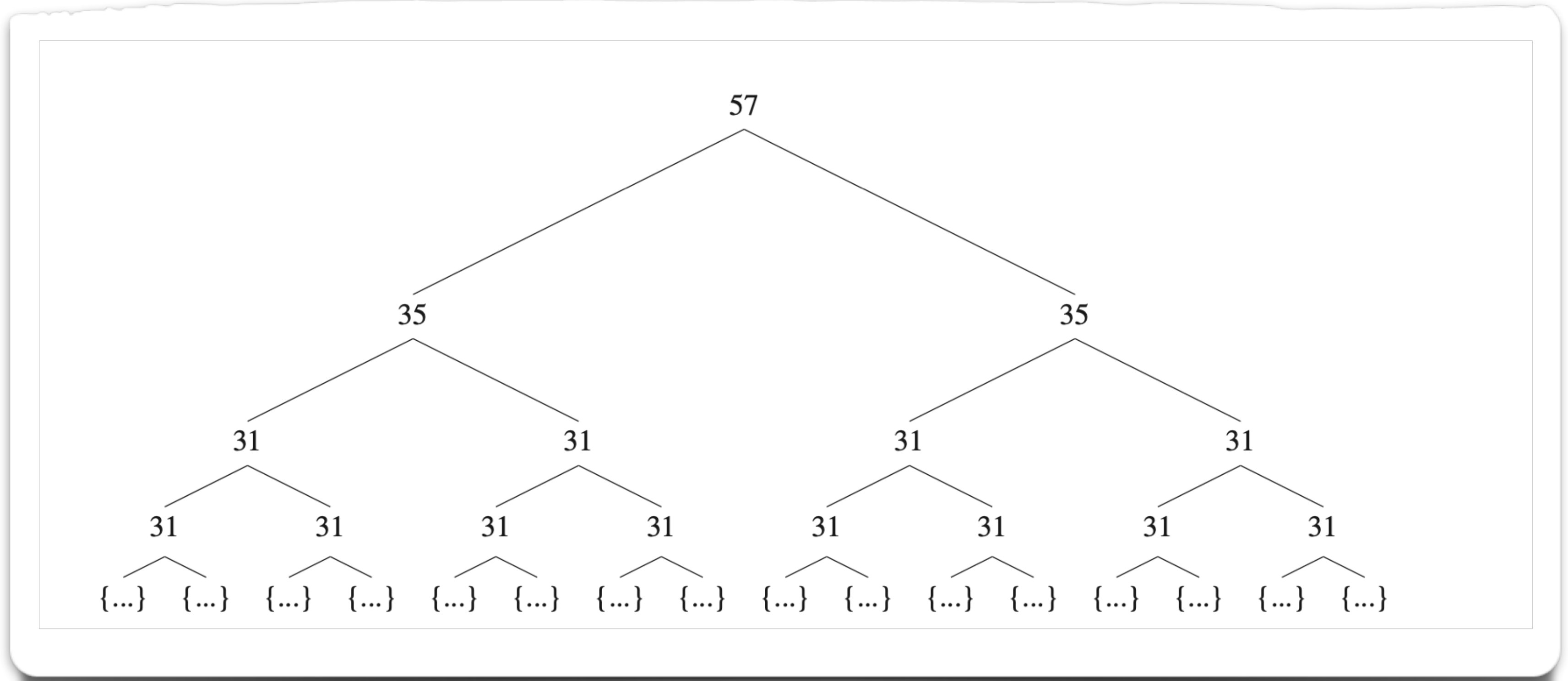
$O(nZ)$

2^n

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	0	0	1	0	0	1	…	0

+17



8

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

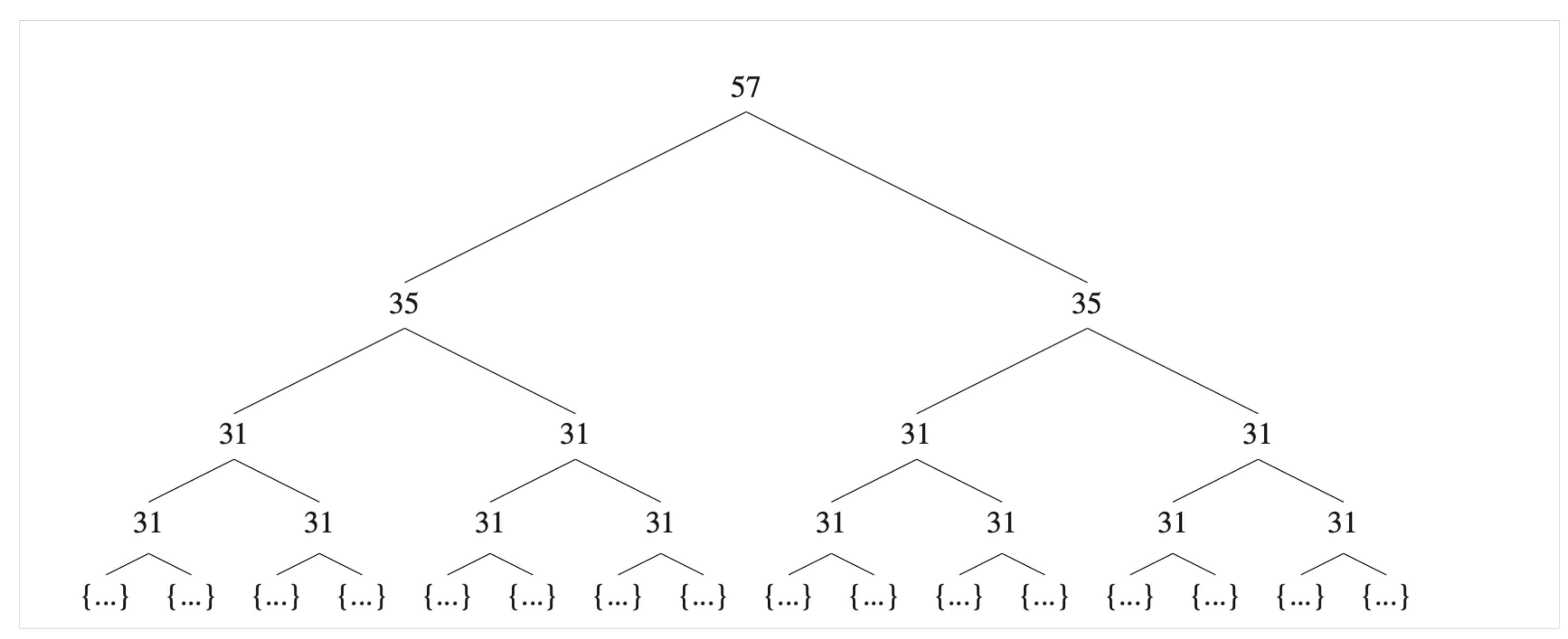
$O(nZ)$

2^n

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	0	0	1	0	0	1	…	0

+17



16

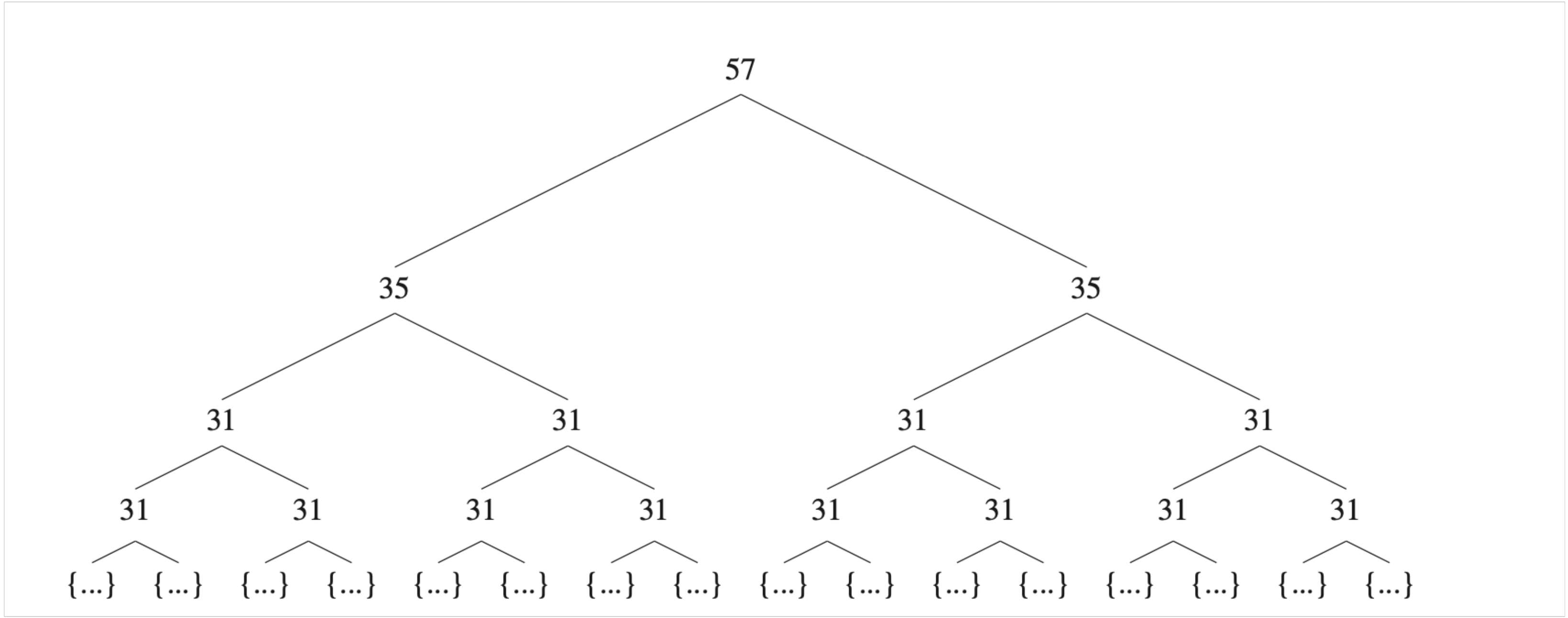
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

$O(nZ)$

2^n

$i \backslash x$	0	1	2	3	3	5	6	7	…	13	14	15	16	17	18	19	20	… 24 30 37	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 +17	0	1	0	0	1	…	0



32

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

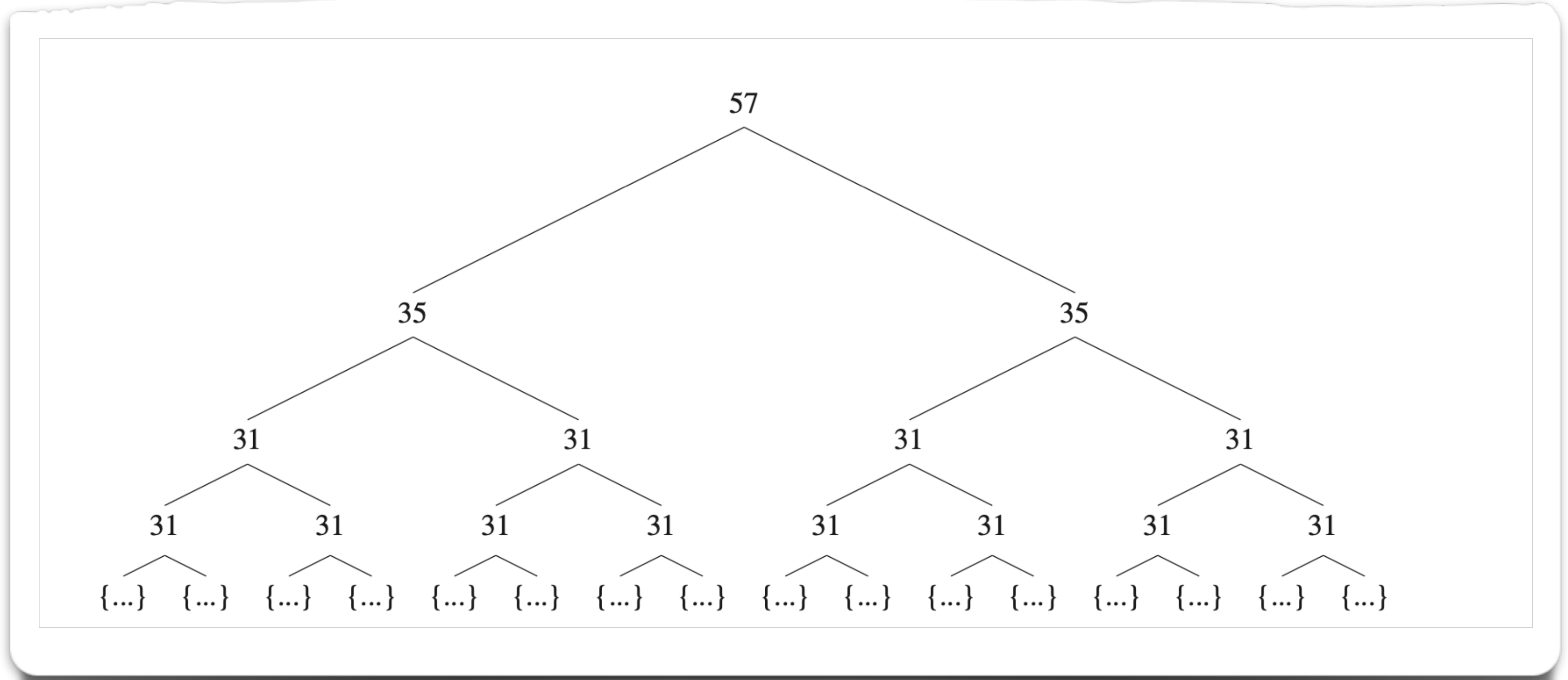
$O(nZ)$

2^n

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	0	0	1	0	0	1	…	0

+17



64

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

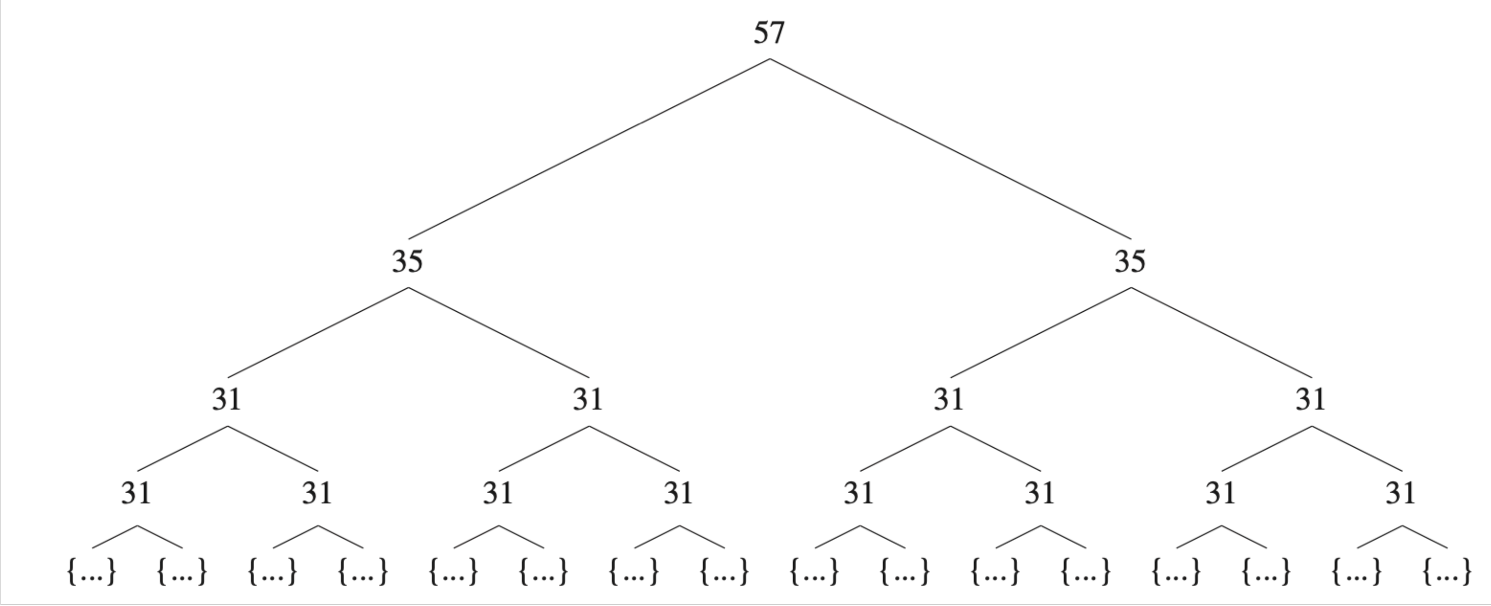
$O(nZ)$

2^n

24 30 37

$i \backslash x$	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

+17



128

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

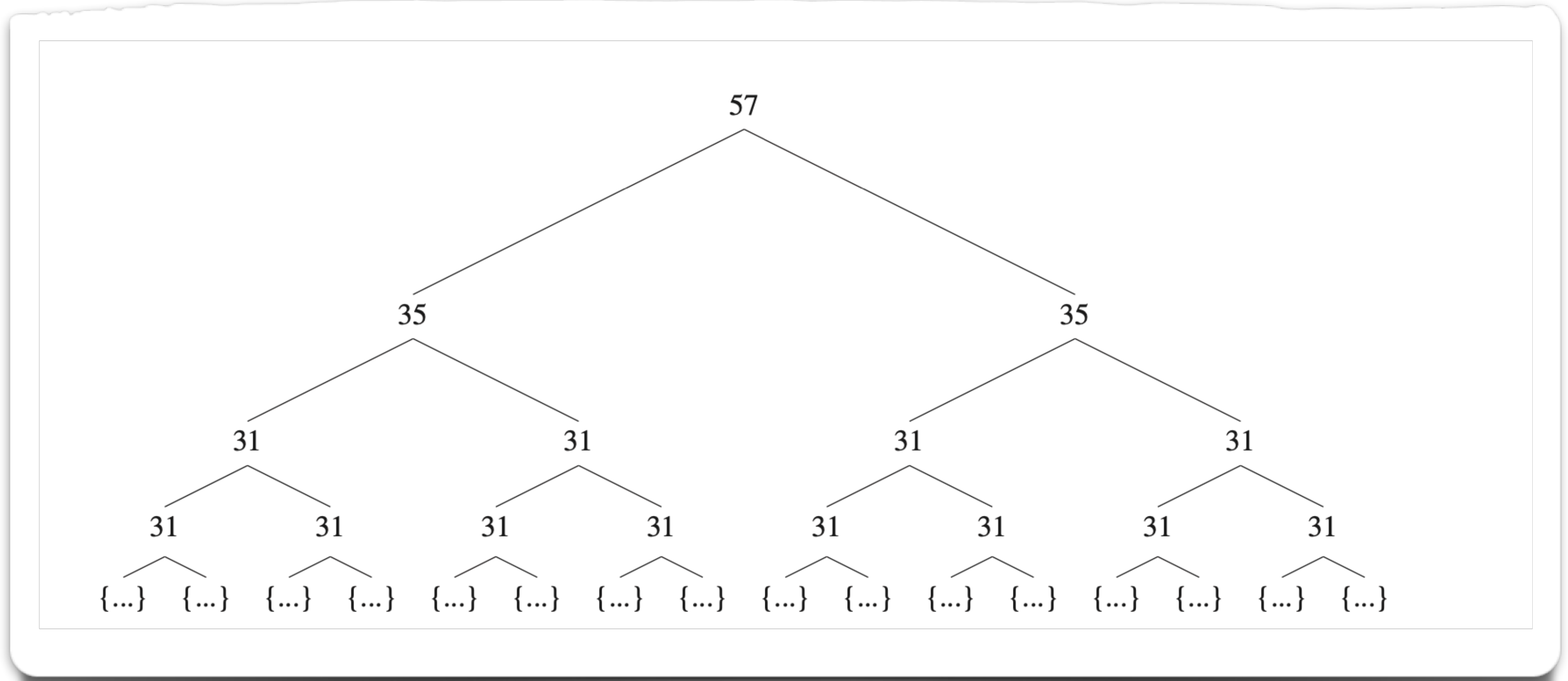
$O(nZ)$

2^n

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	0	0	1	0	0	1	…	0

+17



$256?!$

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

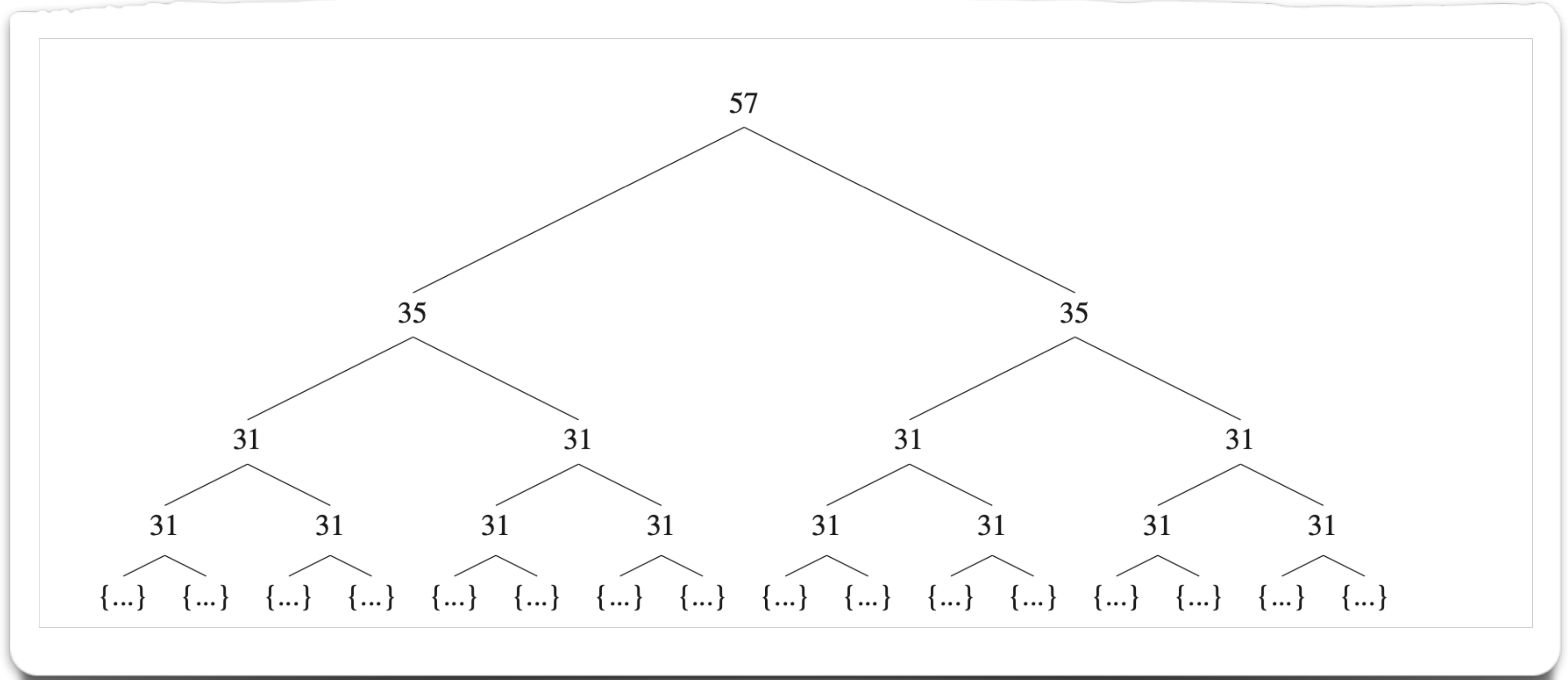
Effizienz?!

$O(nZ)$ 2^n

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	0	0	1	0	0	1	…	0

+17



$256?!$

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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

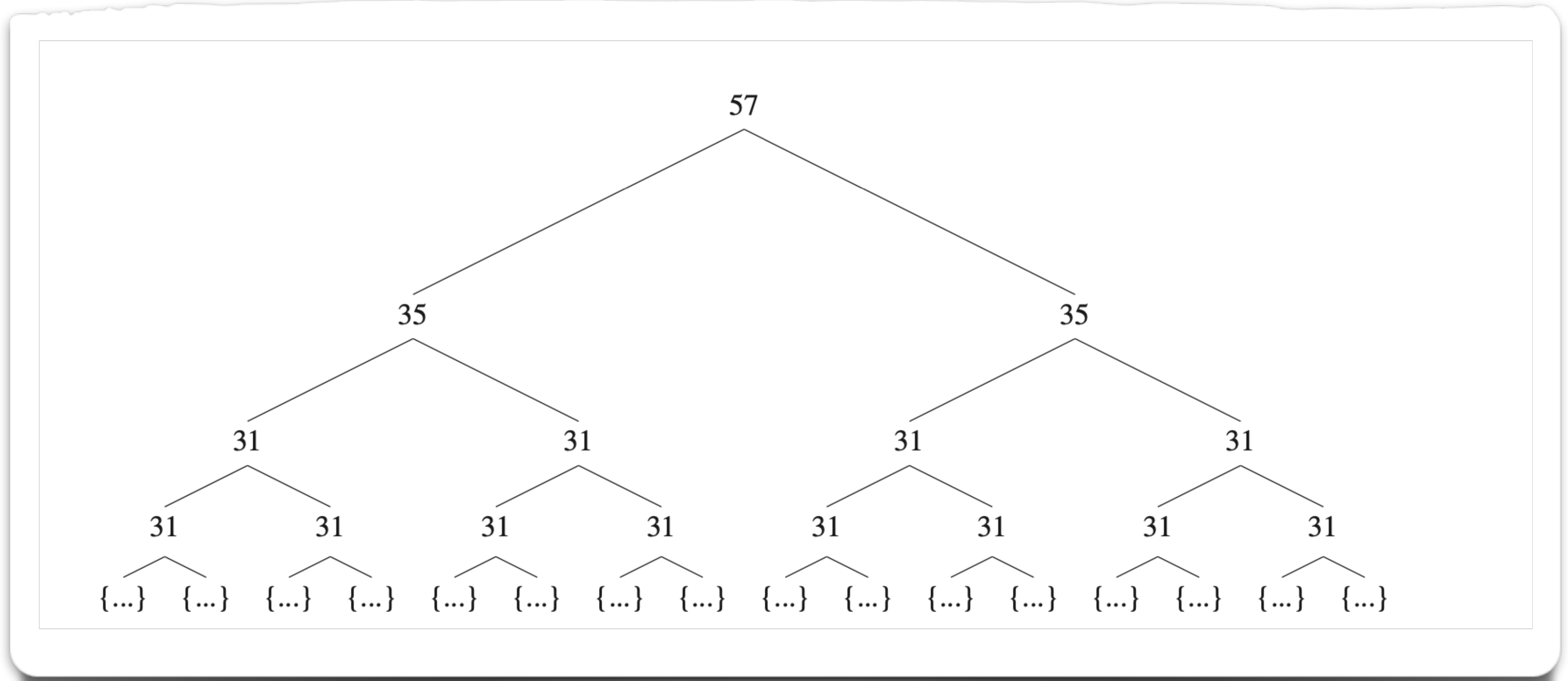
Effizienz?!

$O(nZ)$

2^n

$i \backslash x$	0	1	2	3	3	5	6	7	…	13	14	15	16	17	18	19	20	… 24 30 37	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

+17



$256?!$

- 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

Fragen

- Wie lässt sich das Prinzip verallgemeinern?

Fragen

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

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