



$$P(x, i) = \begin{cases} P(x, i-1), \\ \max\{P(x, i-1), \\ P(x, i-2) + 13\} \end{cases}$$

x \ i	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	0	+13
3						

2 Dynamic Programming

*Algorithmen und Datenstrukturen 2
Sommer 2021*

Prof. Dr. Sándor Fekete

Beispiel

Beispiel

Beispiel 1.10.

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

Gesamtsumme: 240

Beispiel

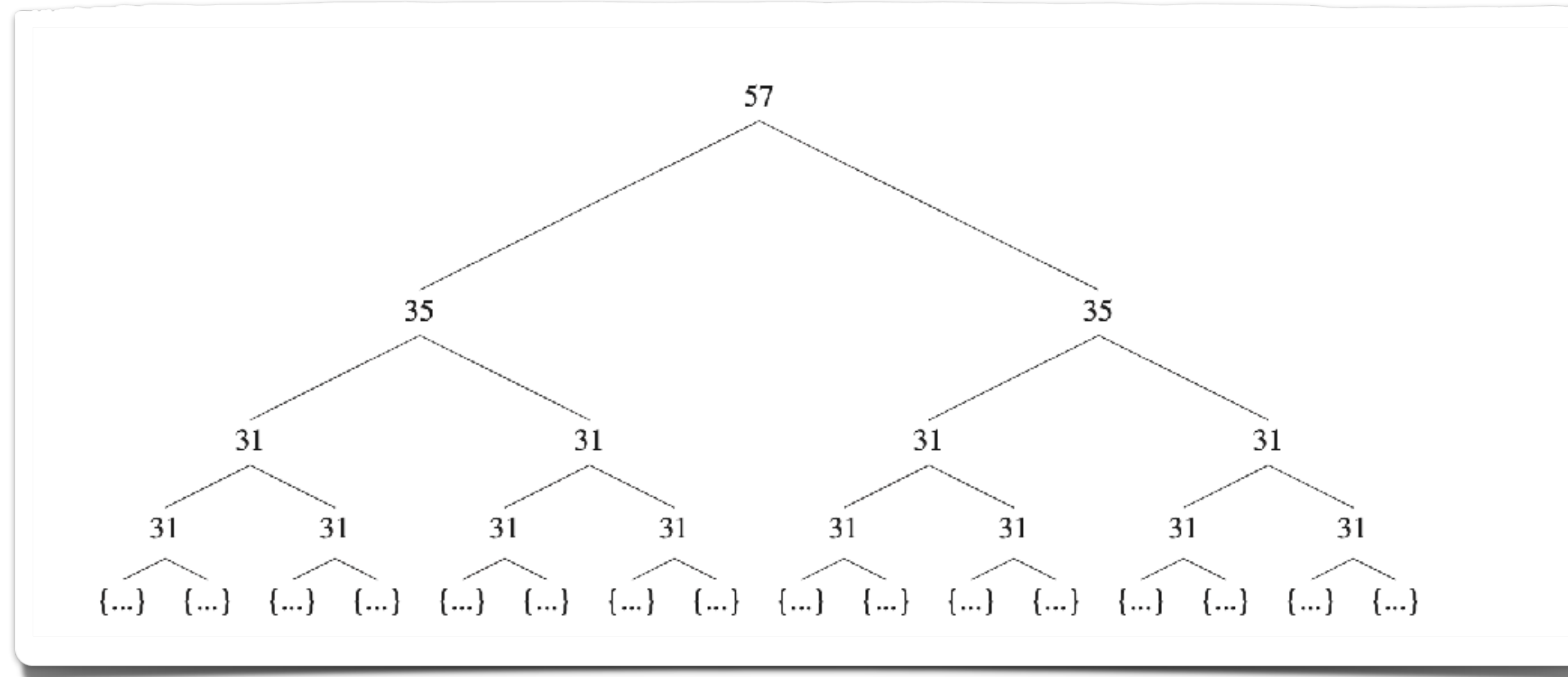
Beispiel 1.10.

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

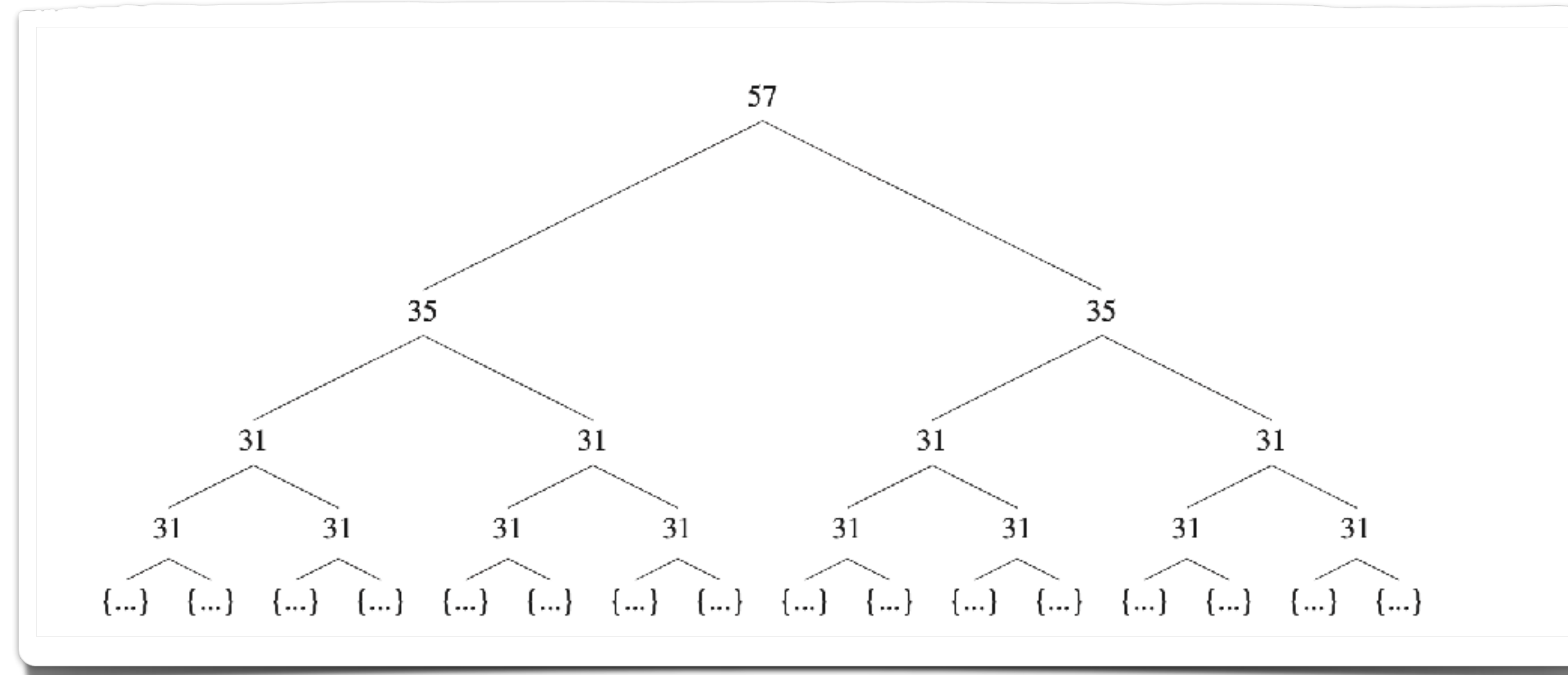
Gesamtsumme: 240

Finde Aufteilung in zwei gleiche Teilmengen!

Enumerationsprinzip

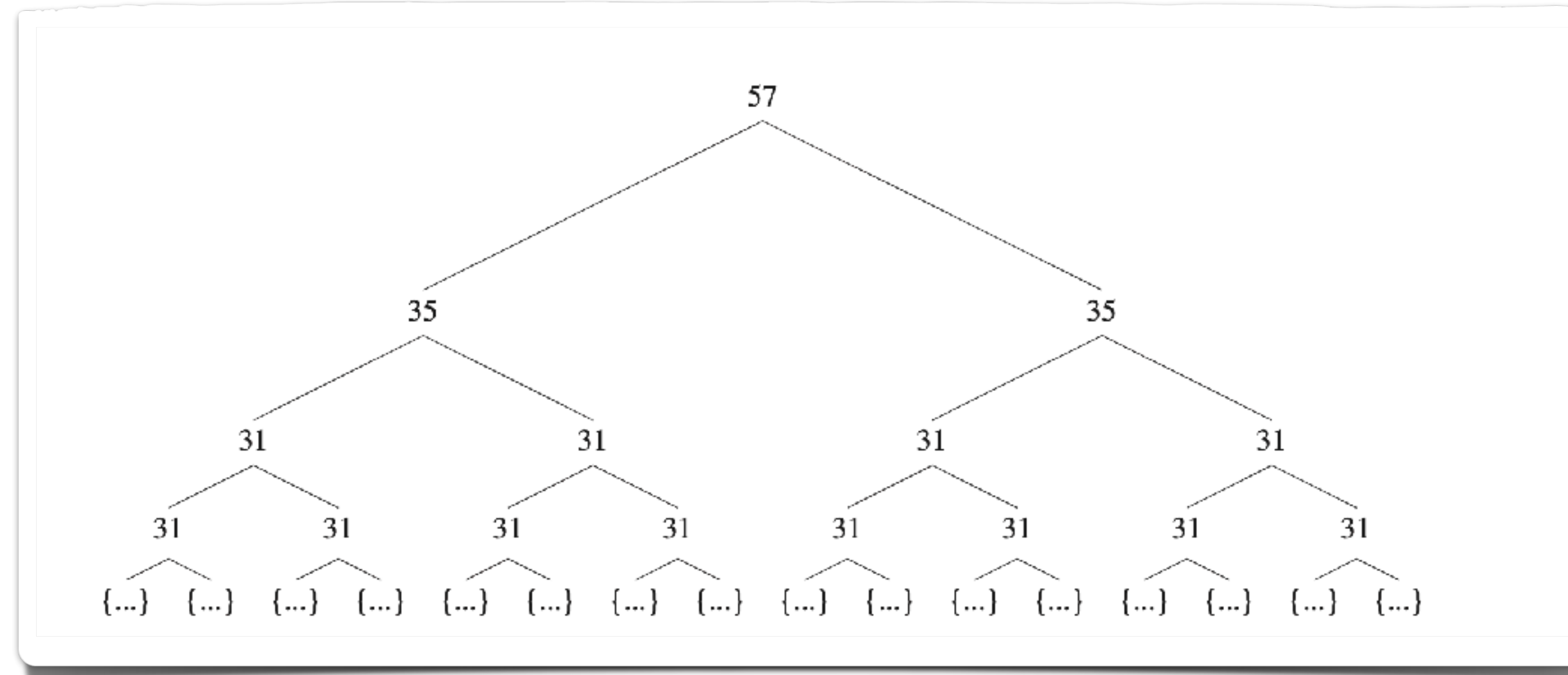


Enumerationsprinzip



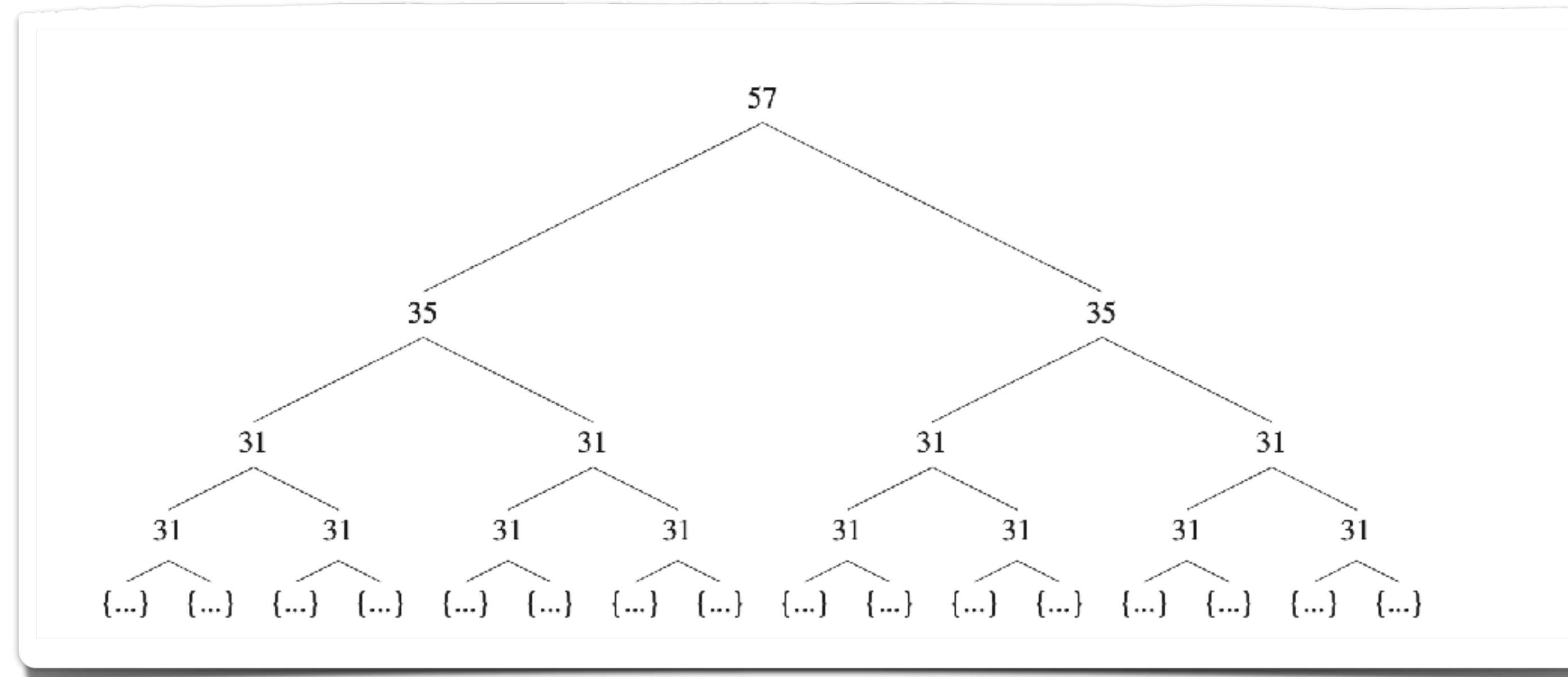
• Exponentiell viele Fälle!

Enumerationsprinzip



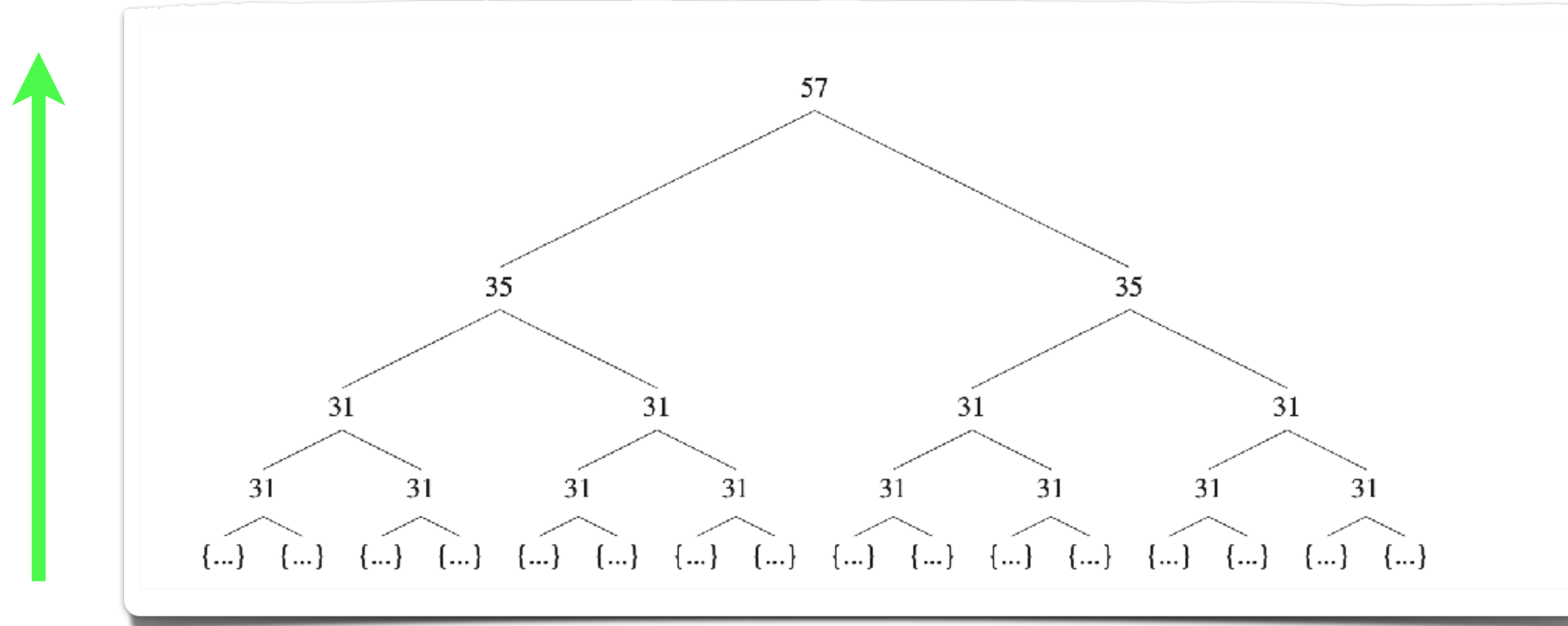
- Exponentiell viele Fälle!
- Wie geht das systematisch?

Enumerationsprinzip



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

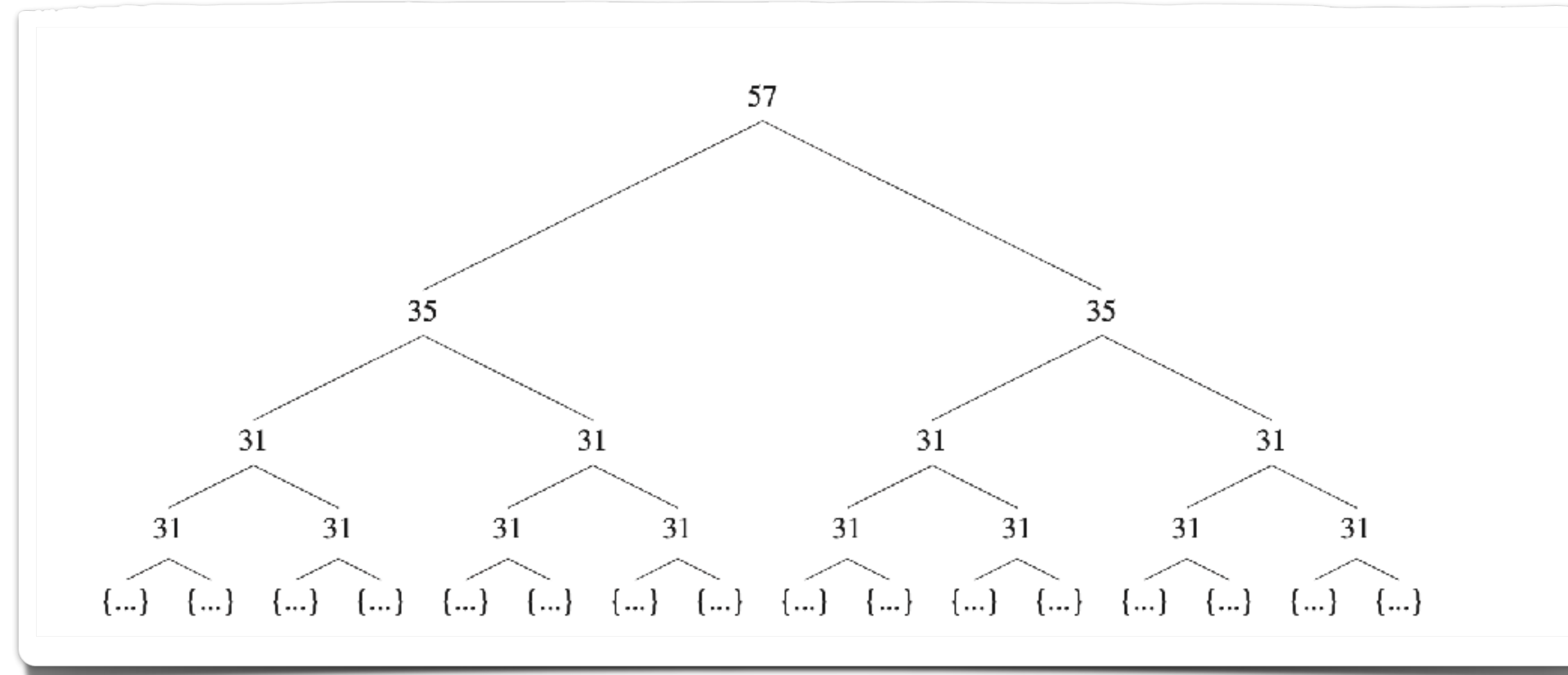
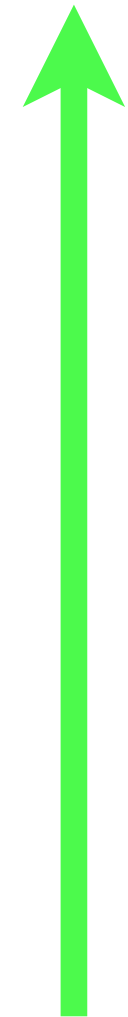
Enumerationsprinzip



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

Enumerationsprinzip

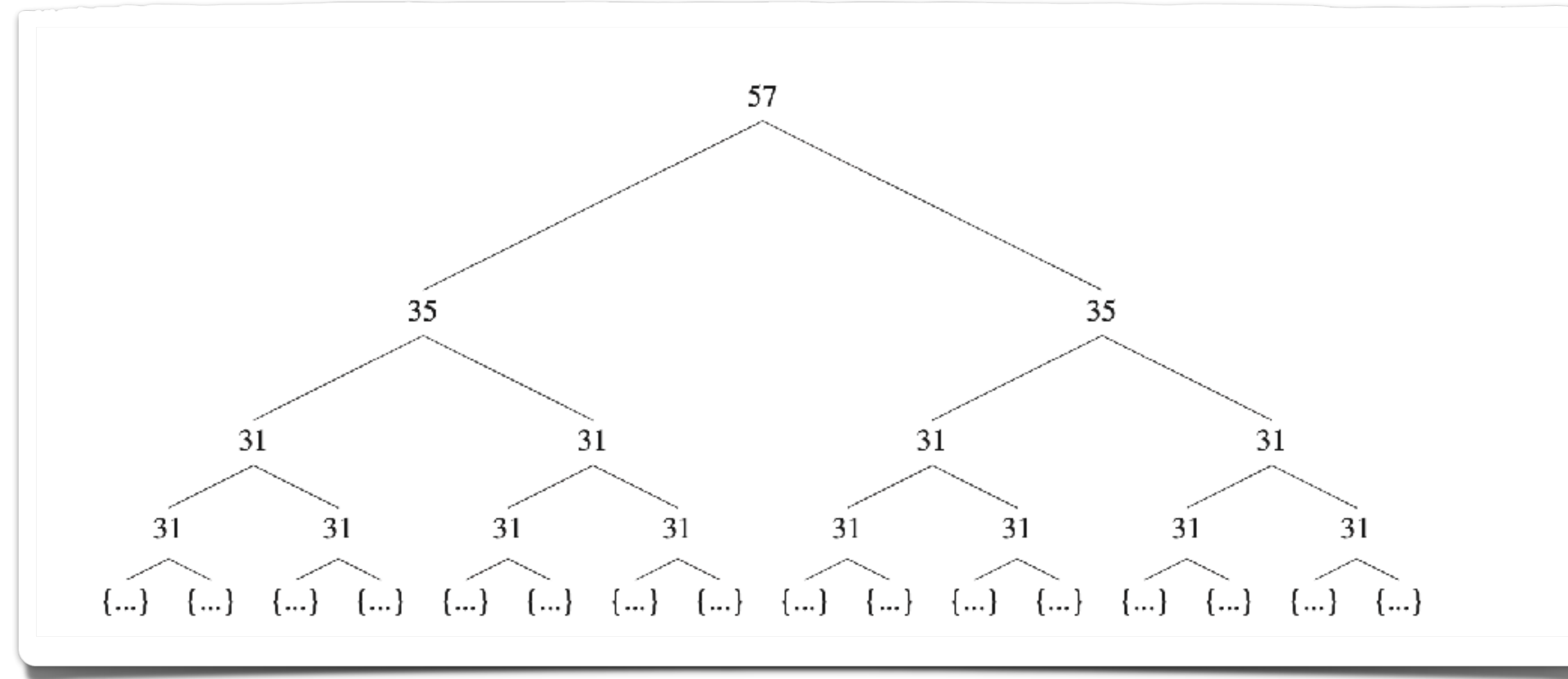
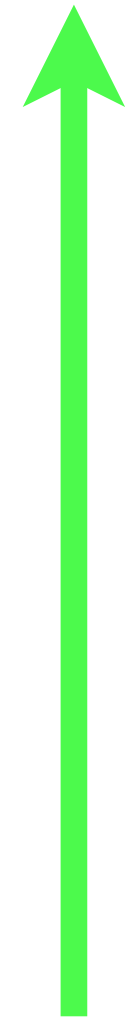
Dynamic
Programming



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

Enumerationsprinzip

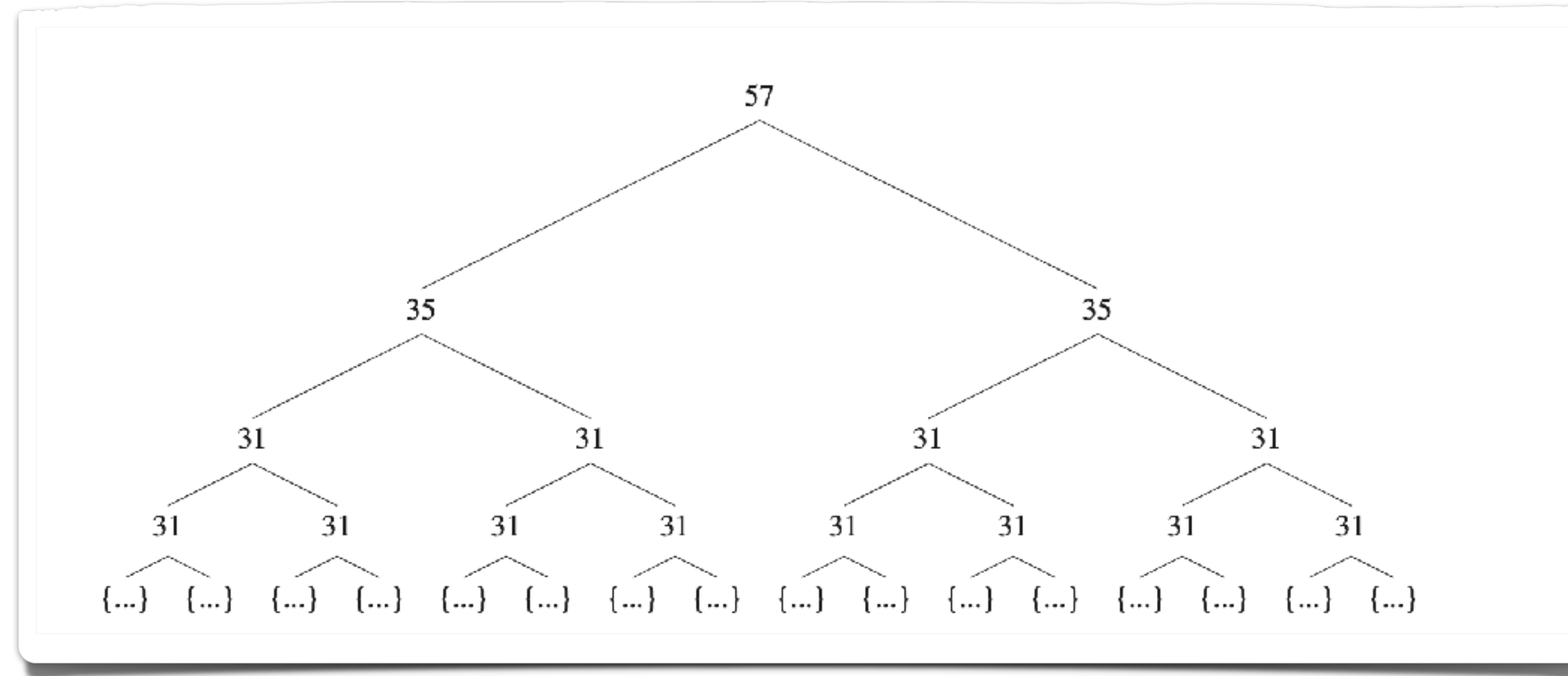
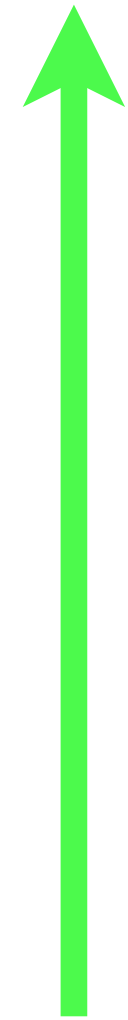
Dynamic
Programming



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

Enumerationsprinzip

Dynamic
Programming



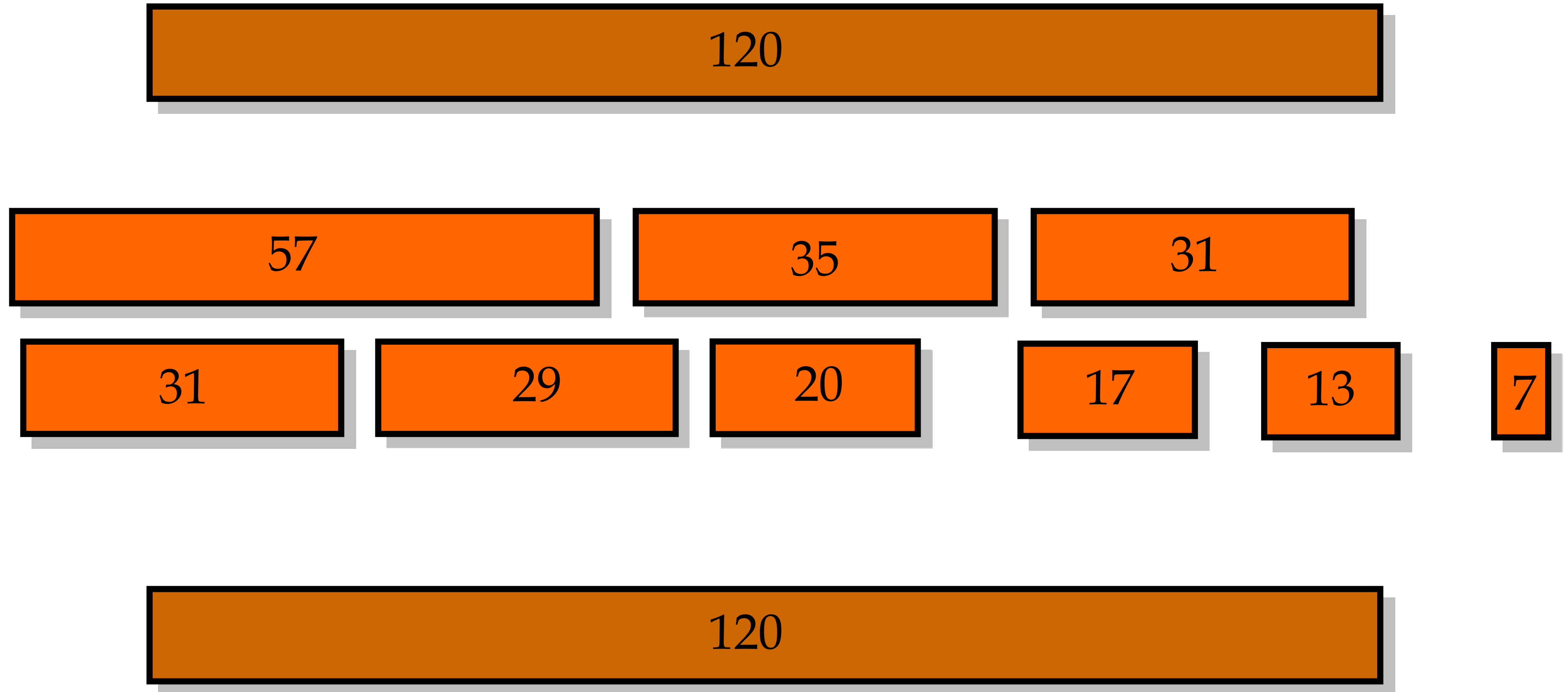
Branch and
Bound



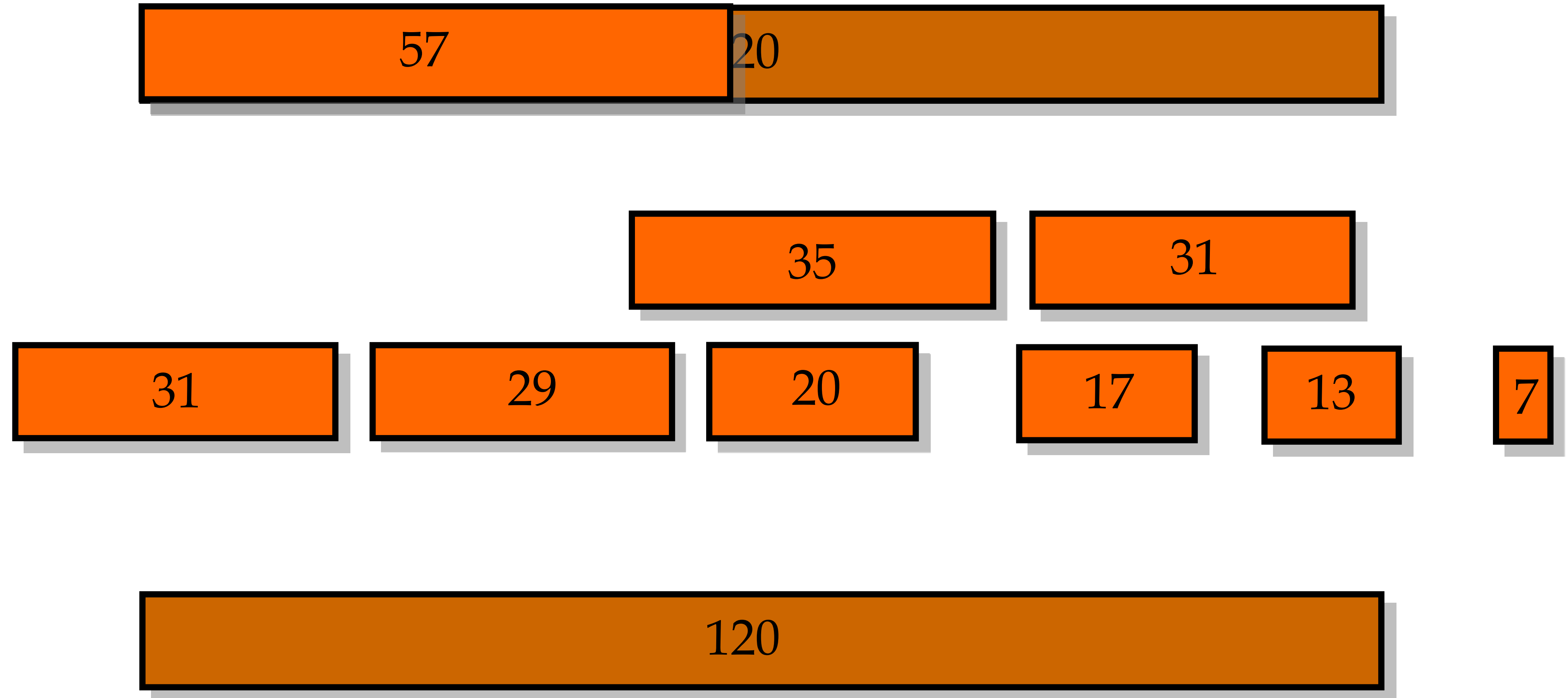
- Exponentiell viele Fälle!
- Wie geht das systematisch?
- Wo kann man Arbeit sparen?

Keine Lösung?!

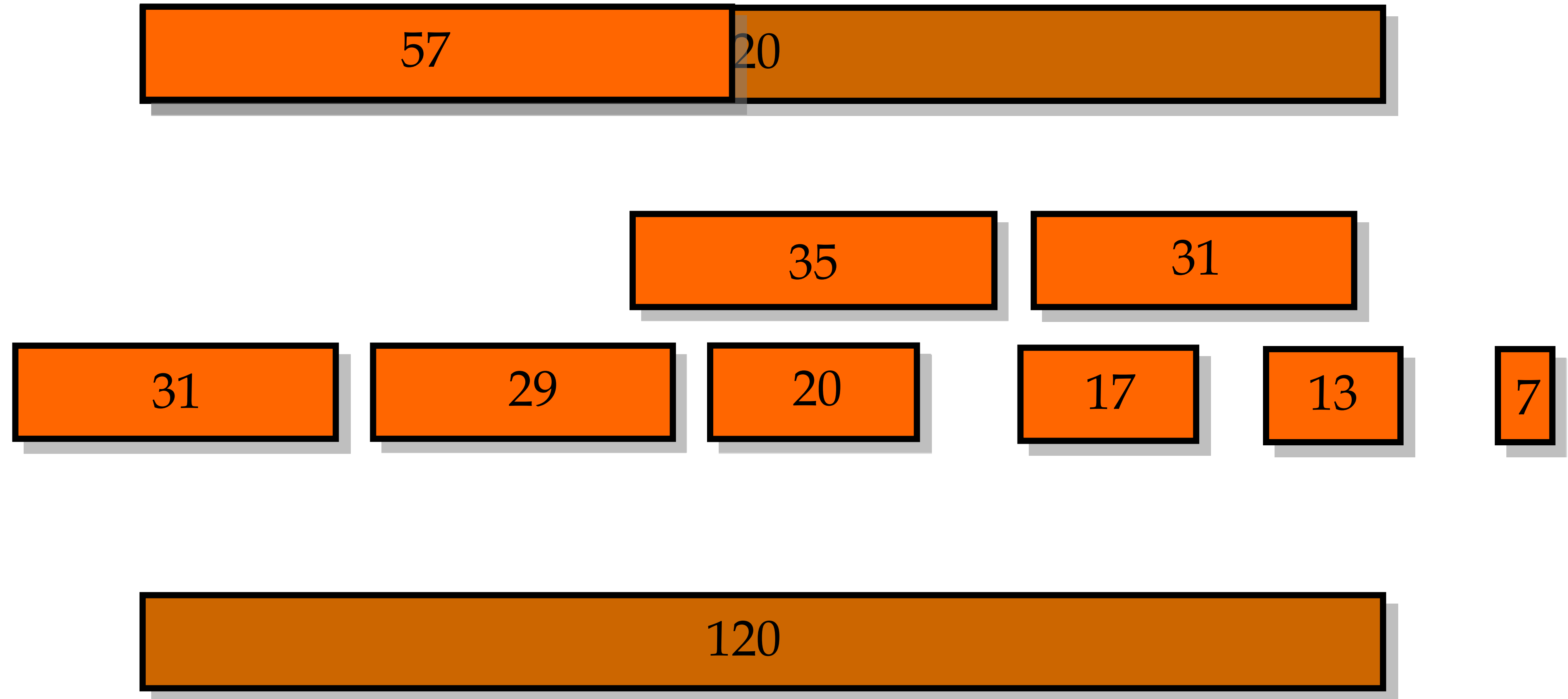
Keine Lösung?!



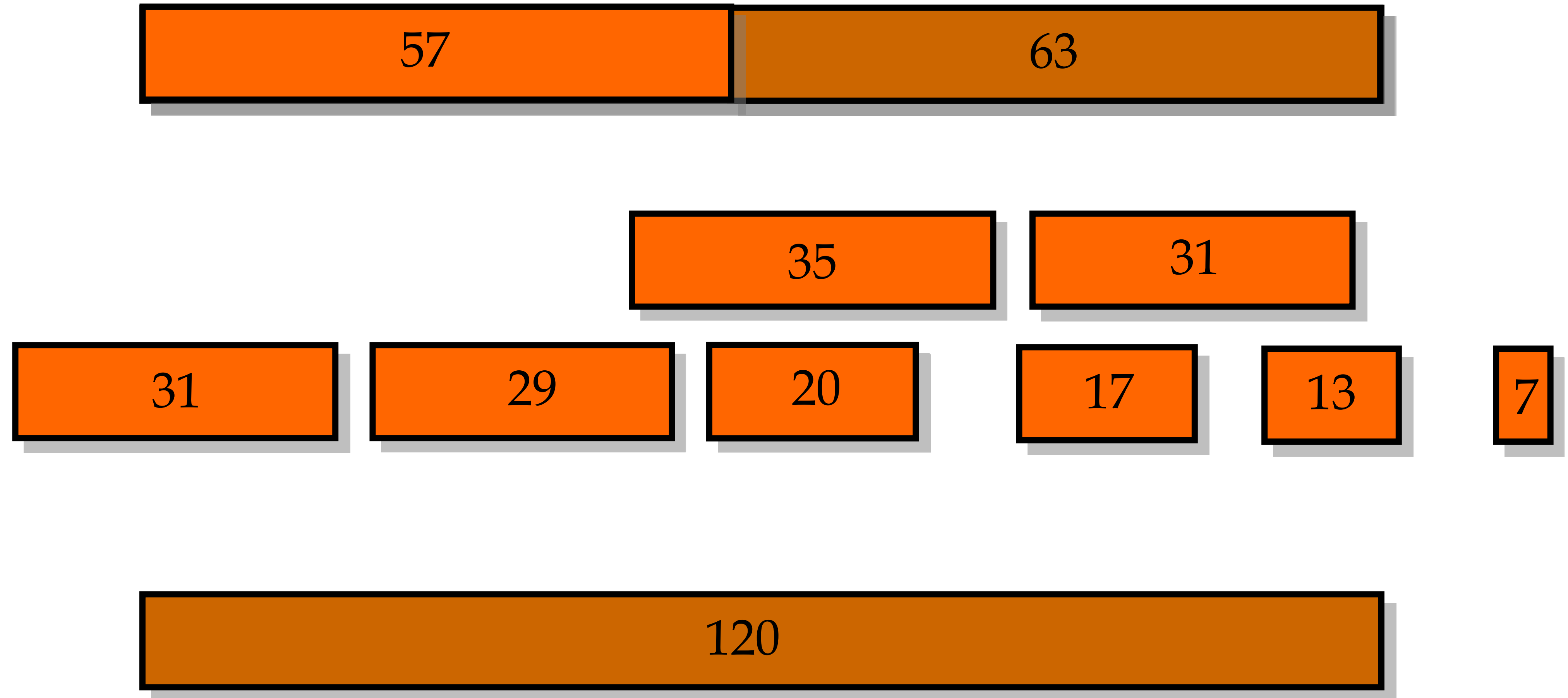
Keine Lösung?!



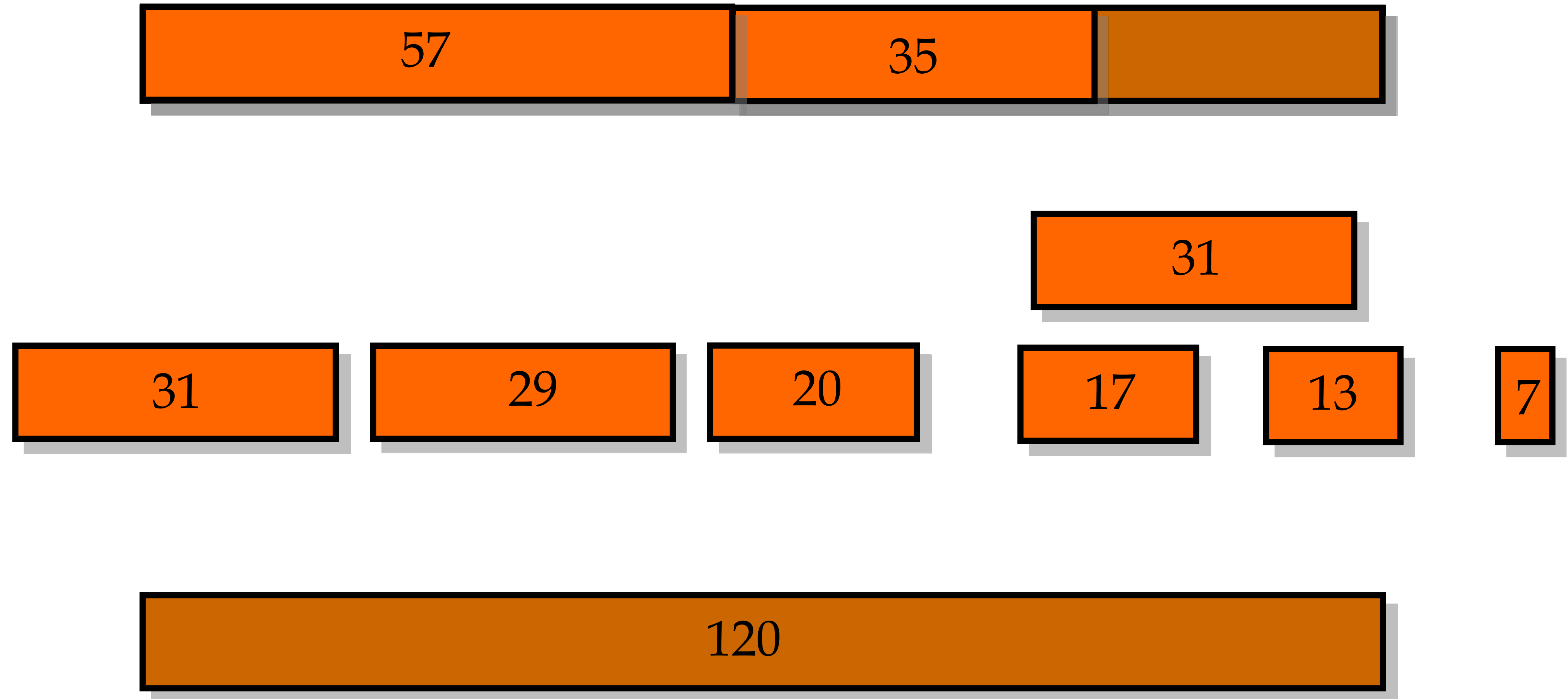
Keine Lösung?!



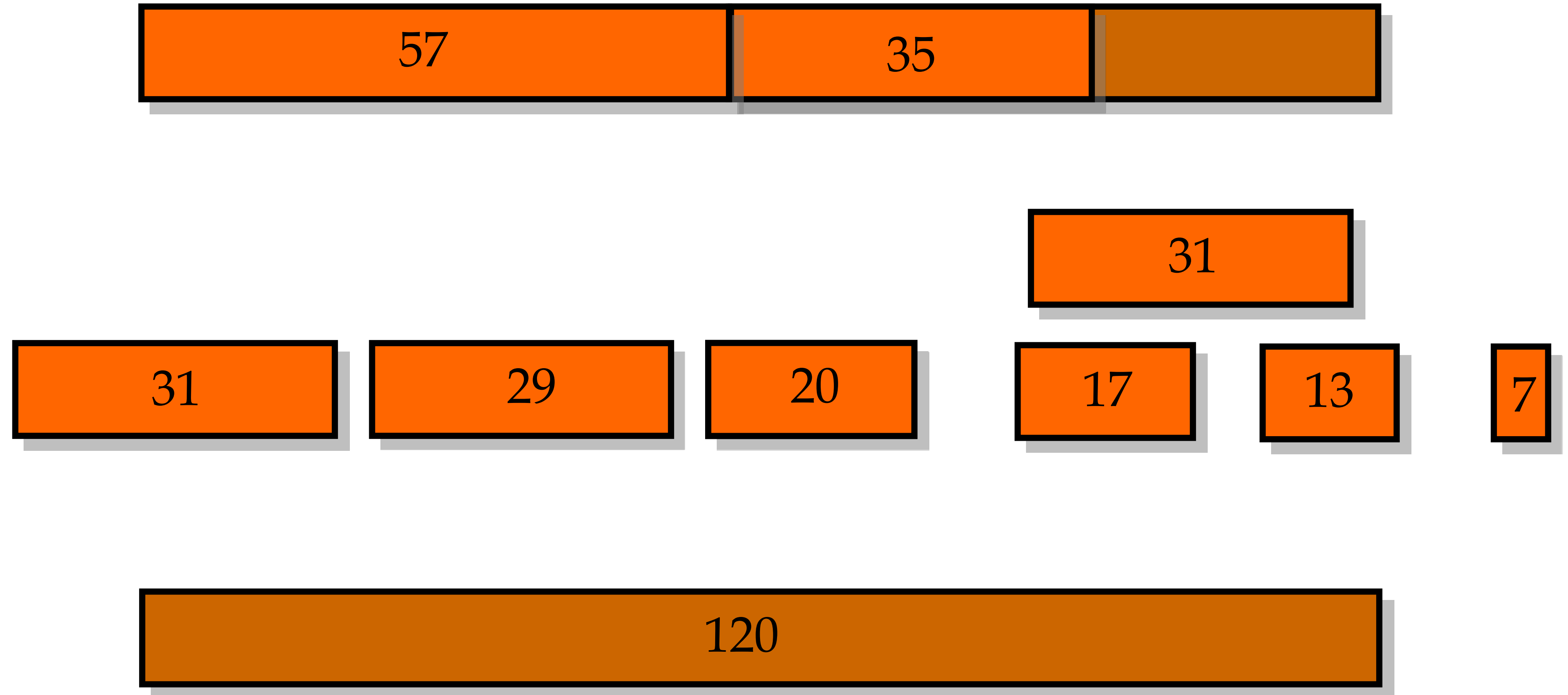
Keine Lösung?!



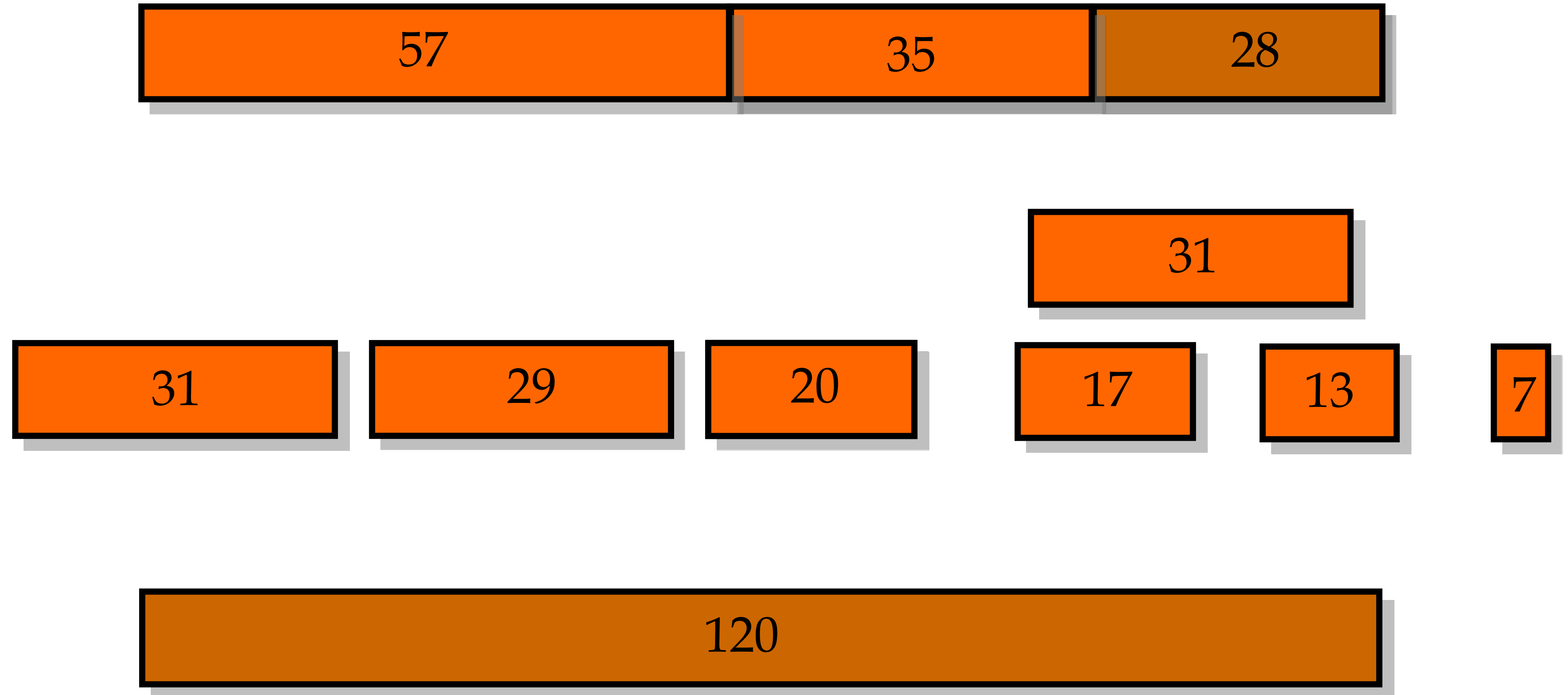
Keine Lösung?!



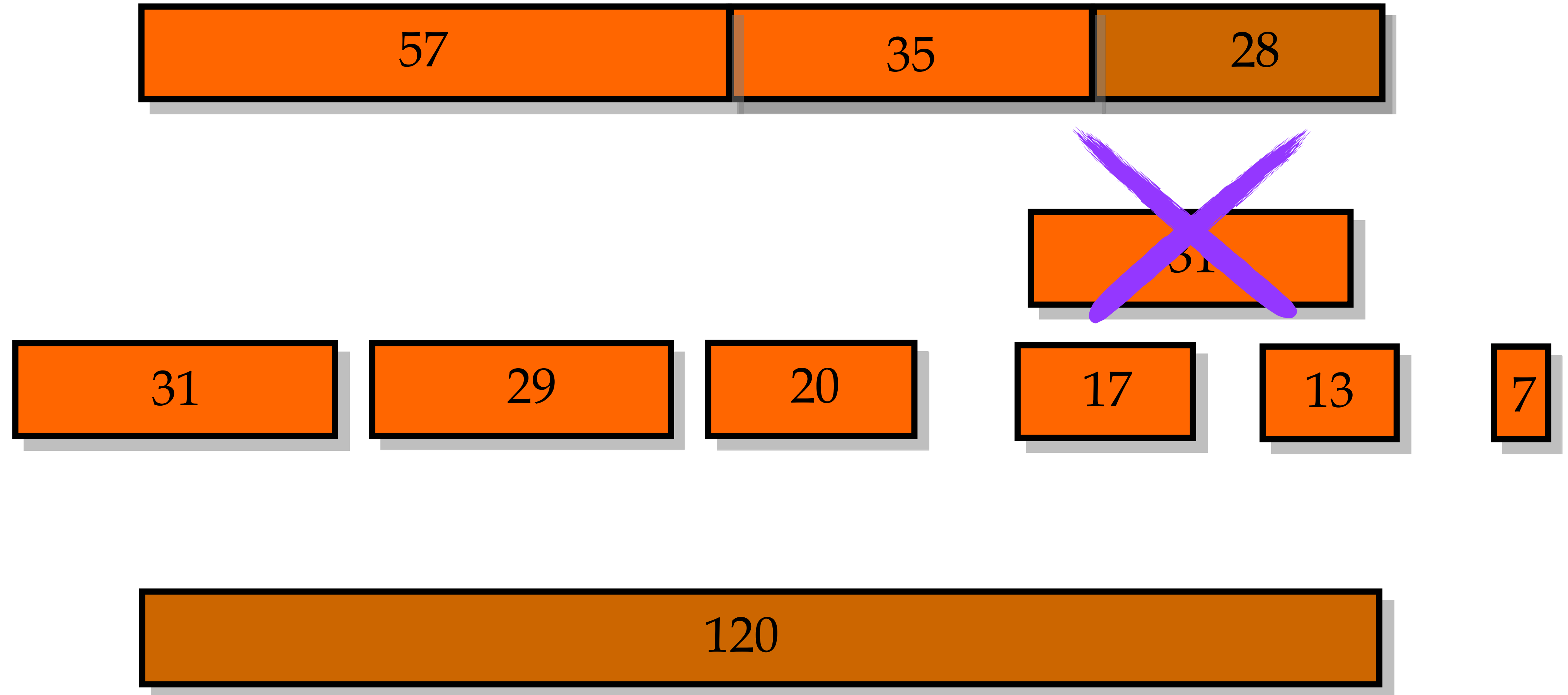
Keine Lösung?!



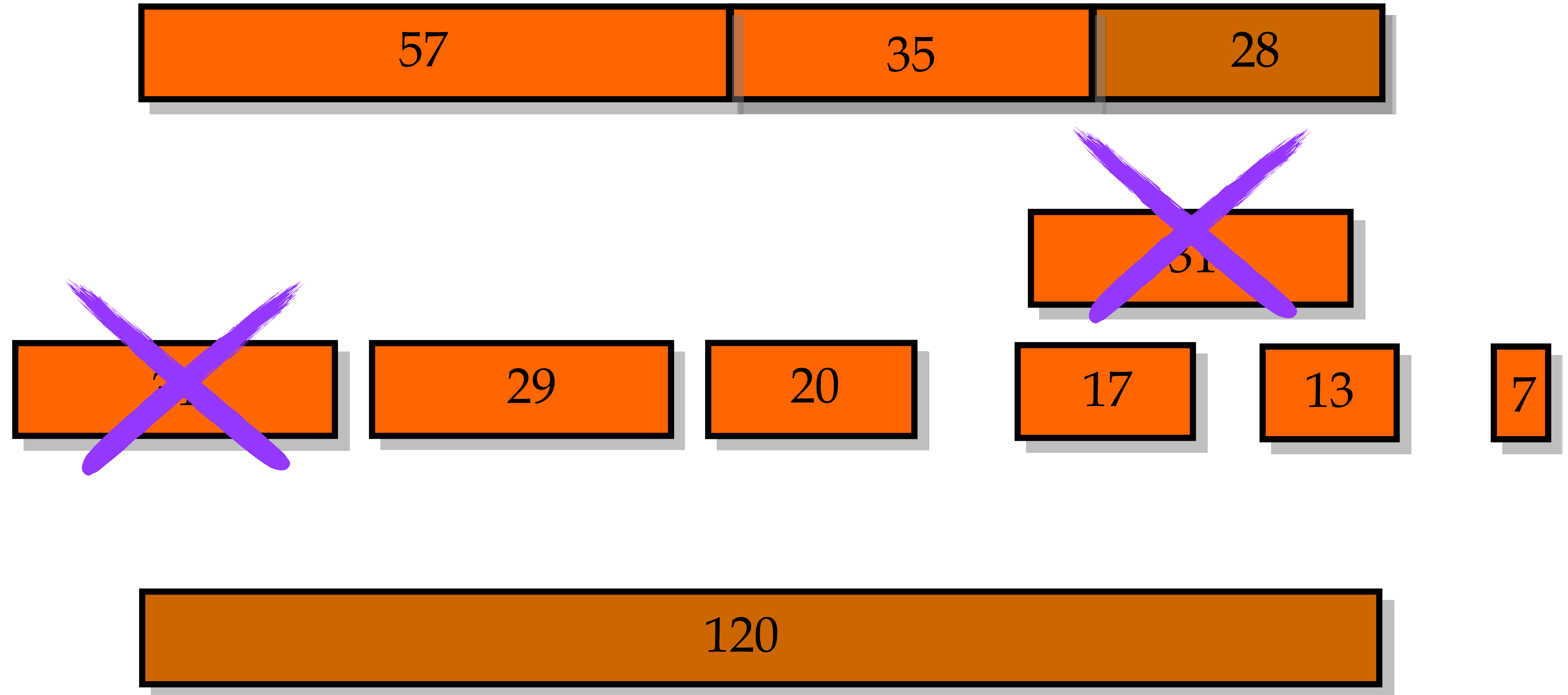
Keine Lösung?!



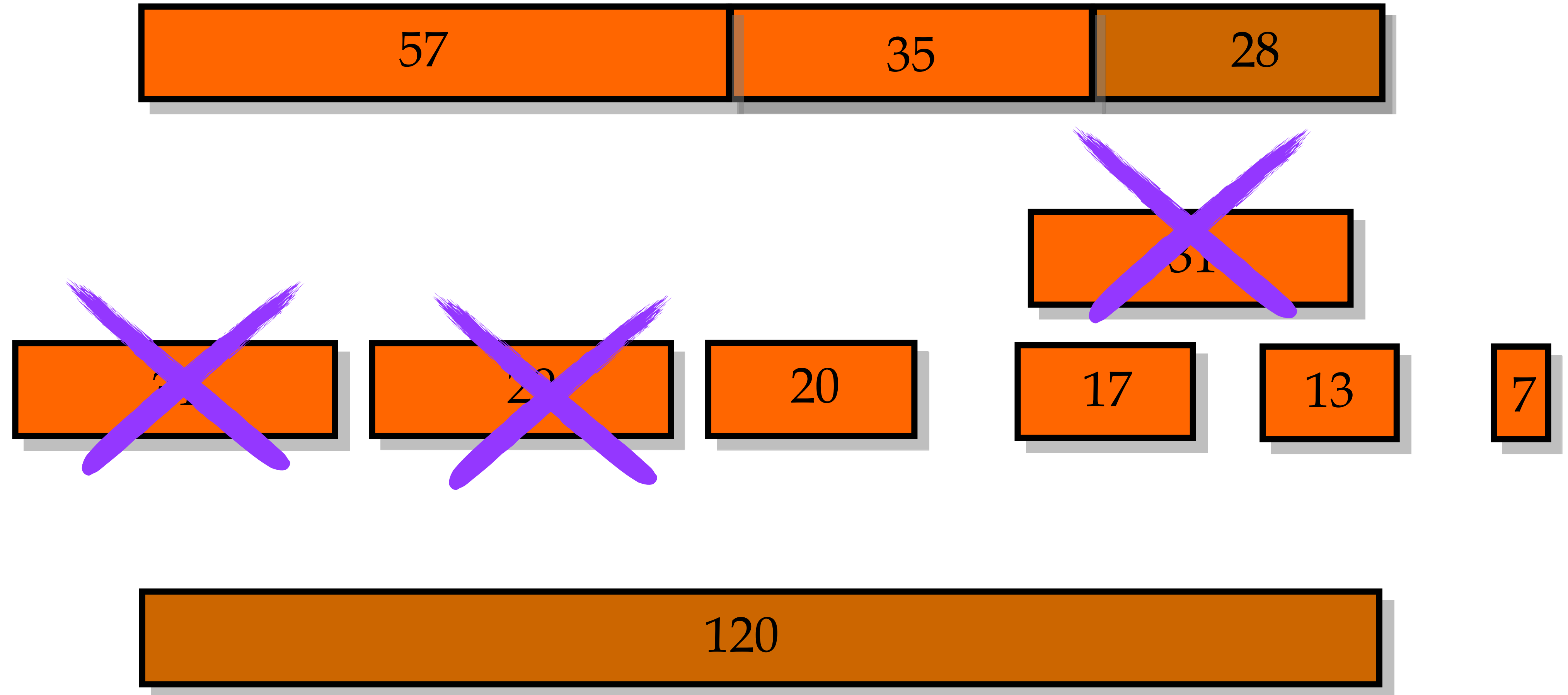
Keine Lösung?!



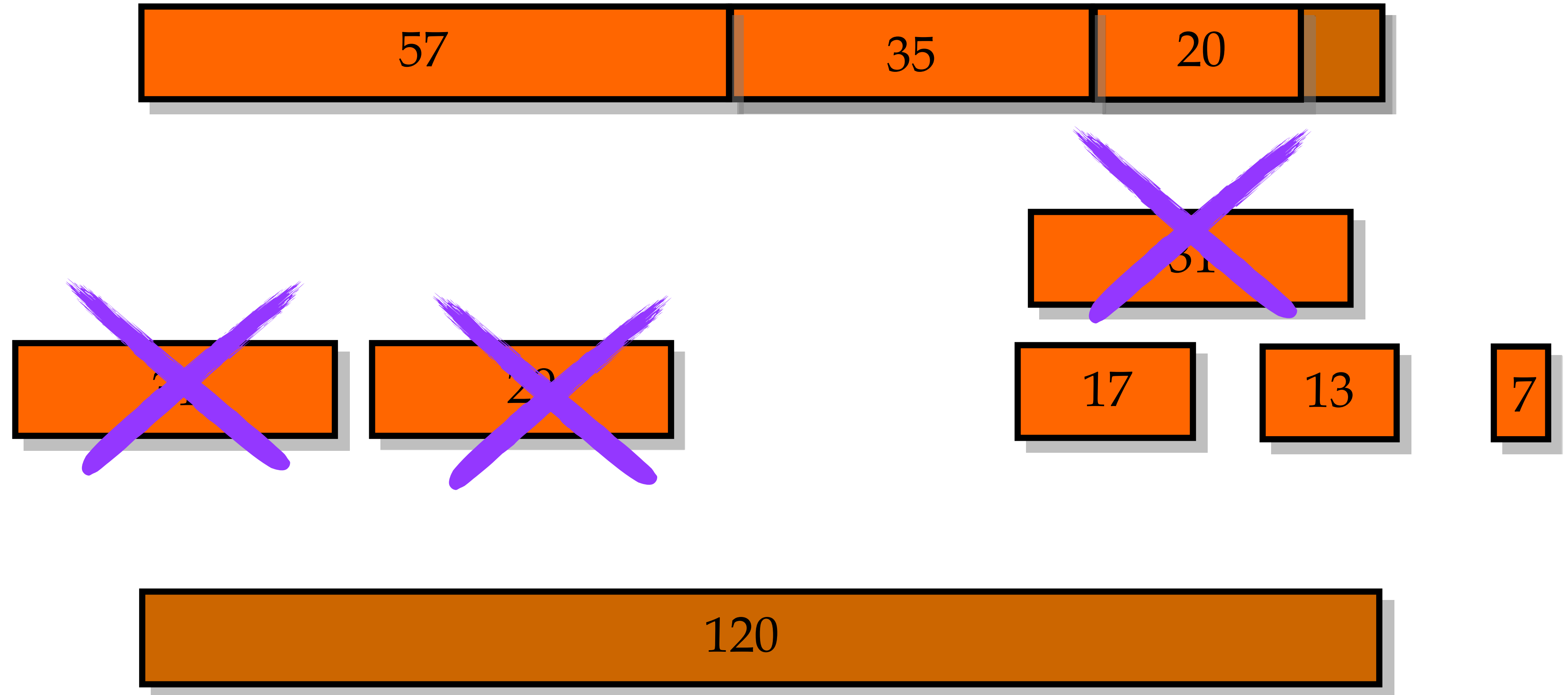
Keine Lösung?!



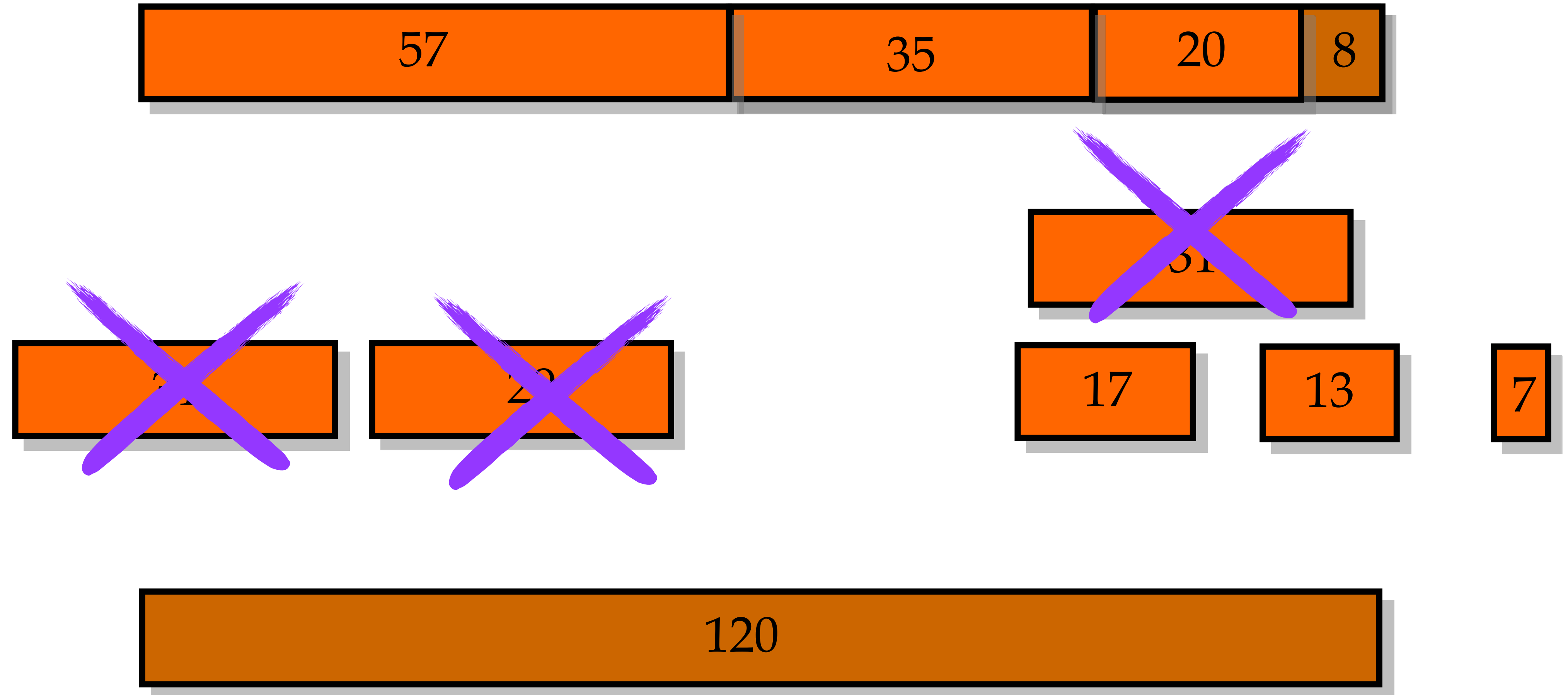
Keine Lösung?!



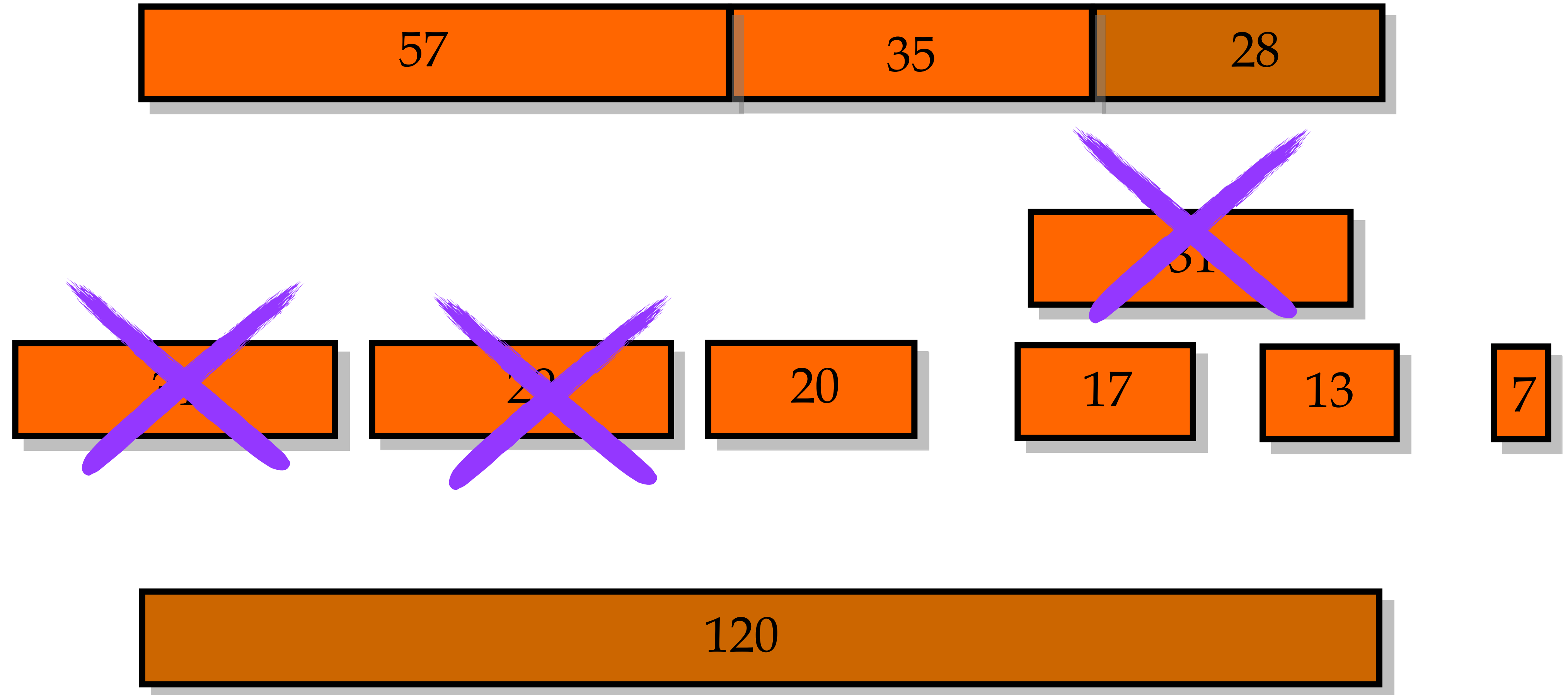
Keine Lösung?!



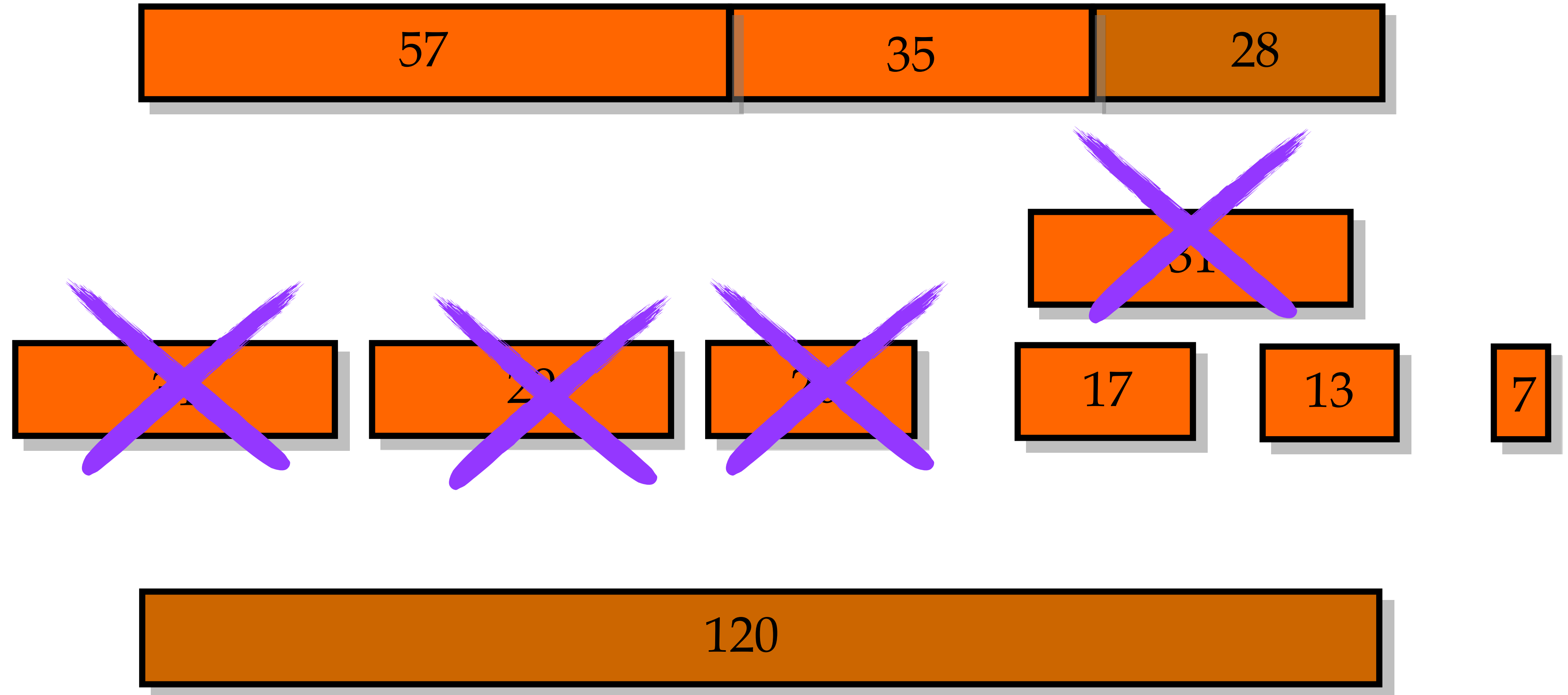
Keine Lösung?!



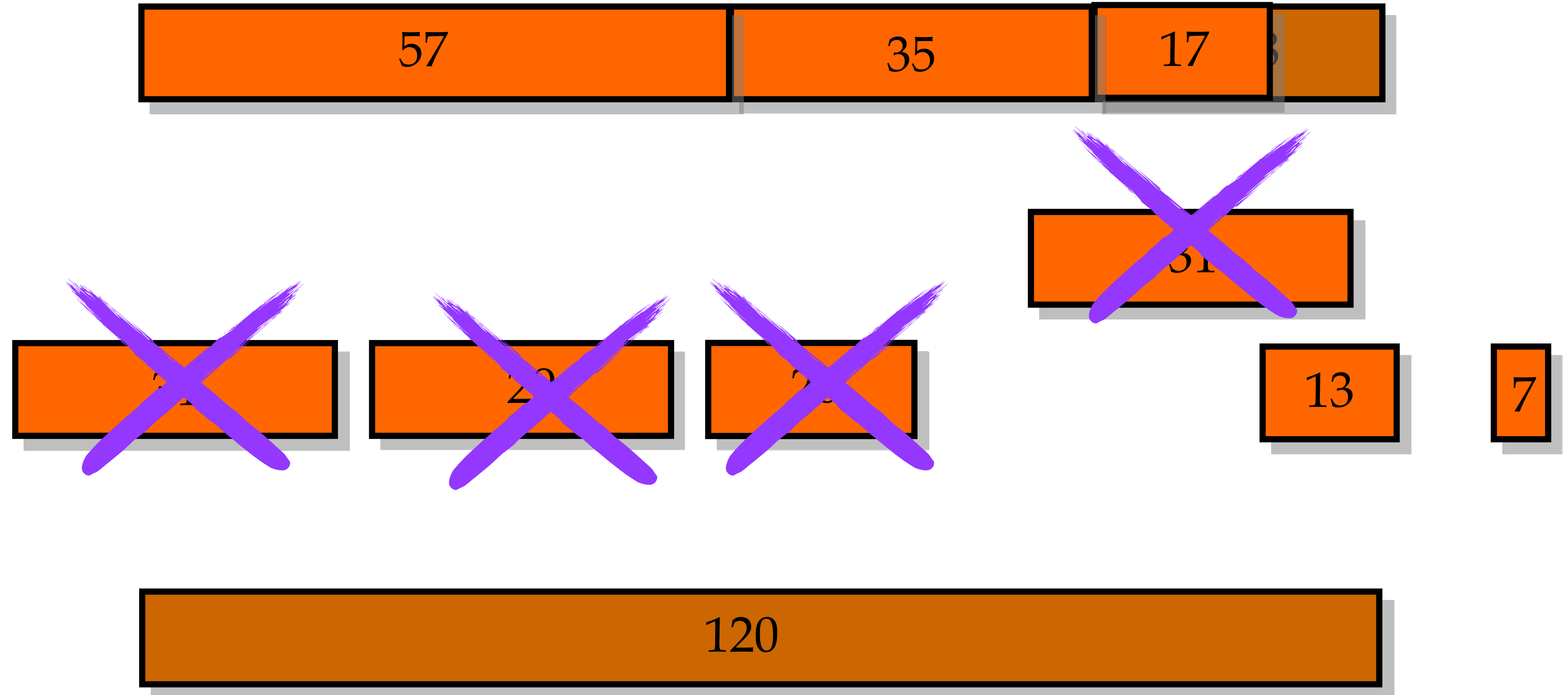
Keine Lösung?!



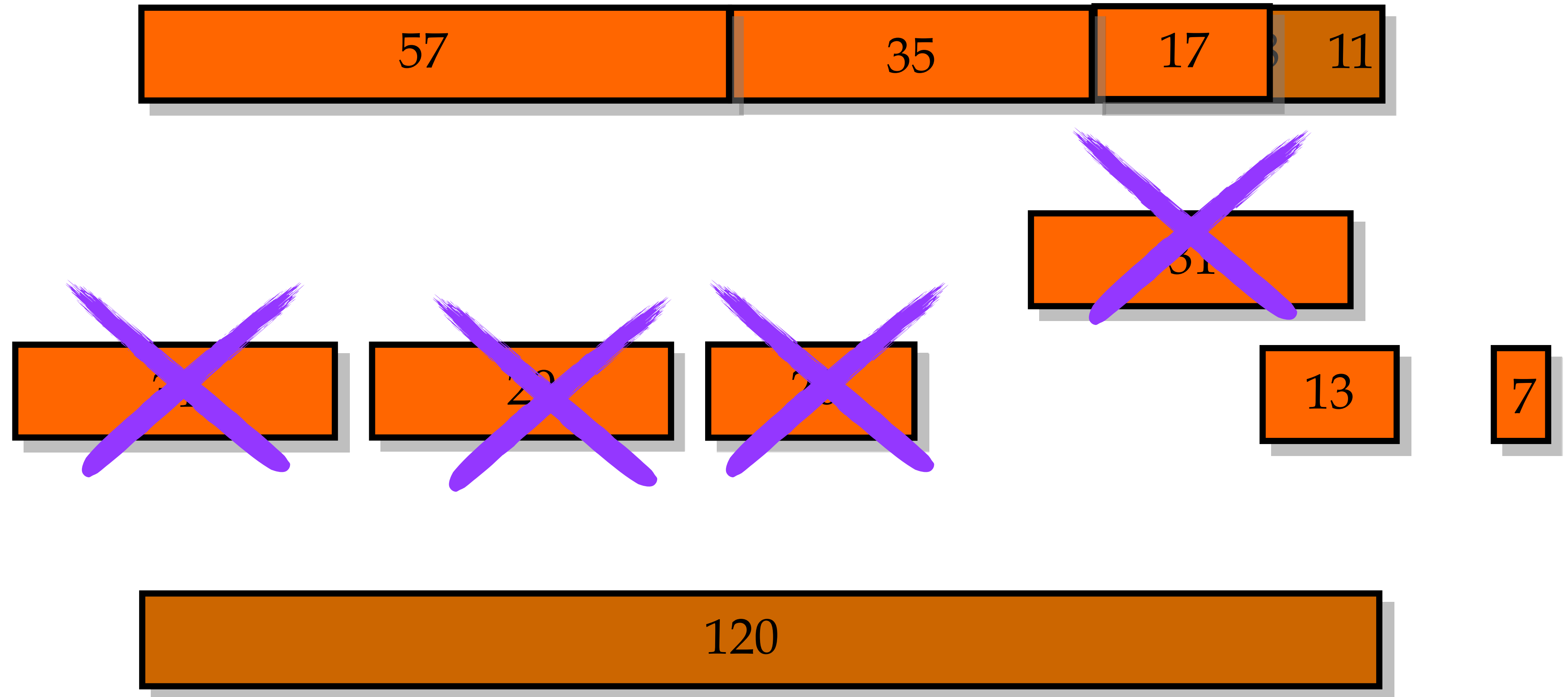
Keine Lösung?!



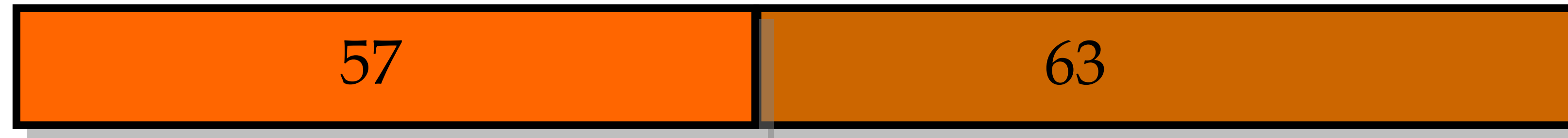
Keine Lösung?!



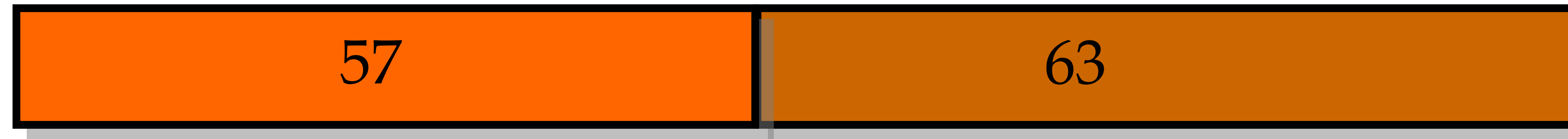
Keine Lösung?!



Keine Lösung?!



Keine Lösung?!

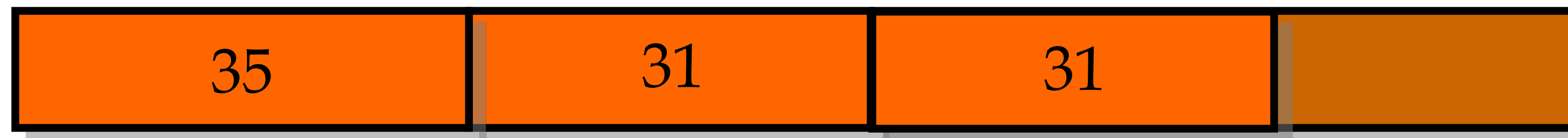


!

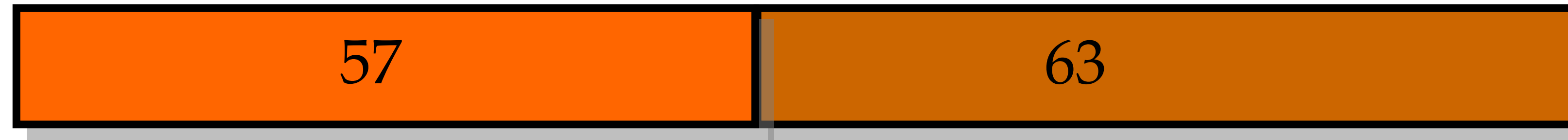


!

!



Keine Lösung?!



!



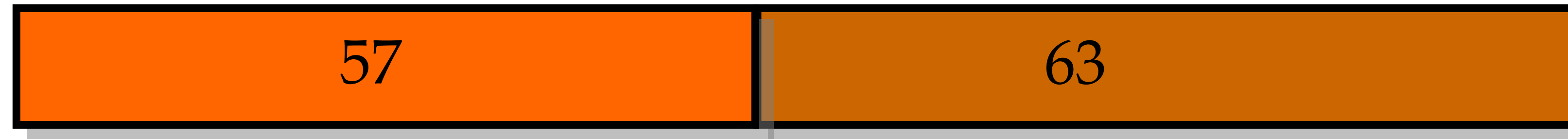
!

!

!



Keine Lösung?!



!



!

!

!



Keine Lösung?!



!

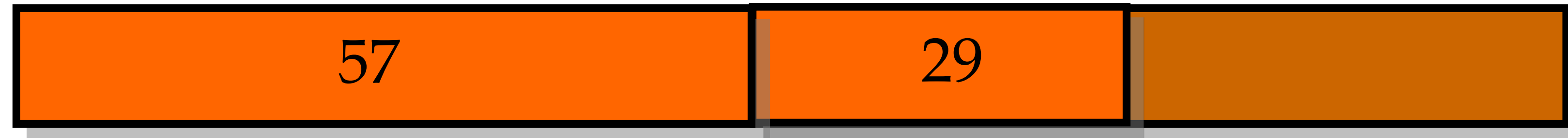


!

!

!

Keine Lösung?!



!

!



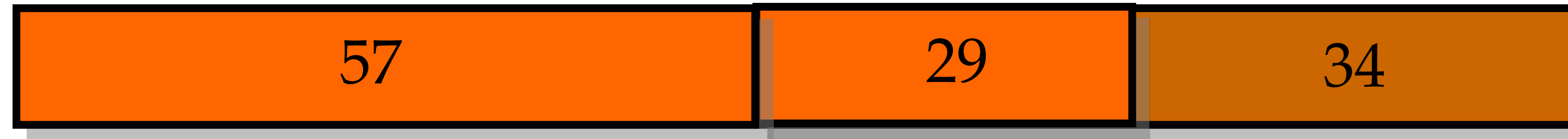
!

!

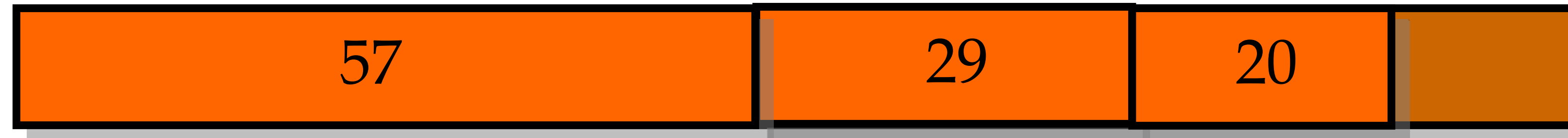
!



Keine Lösung?!



Keine Lösung?!



!

!



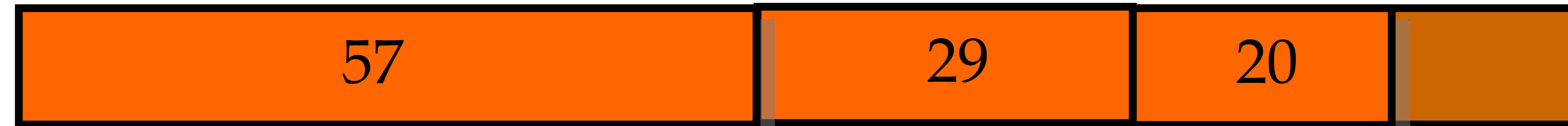
!

!

!



Keine Lösung?!



!

!

!



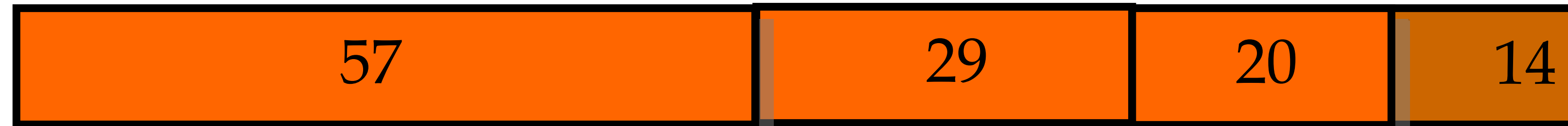
!

!

!



Keine Lösung?!



!

!

!



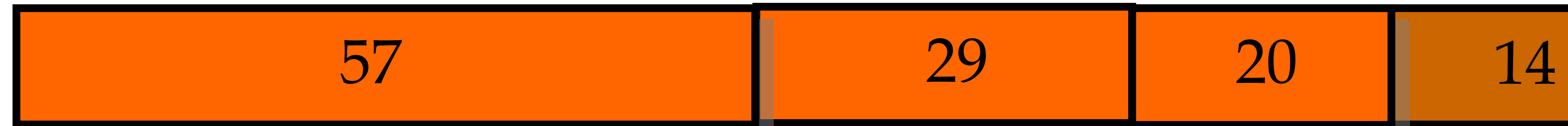
!

!

!



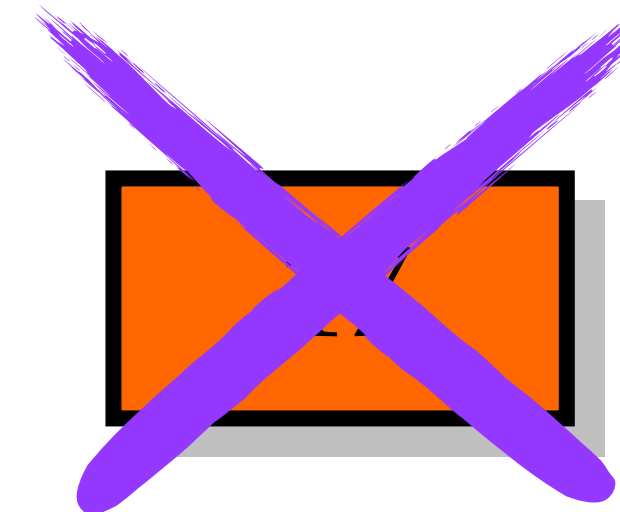
Keine Lösung?!



!

!

!



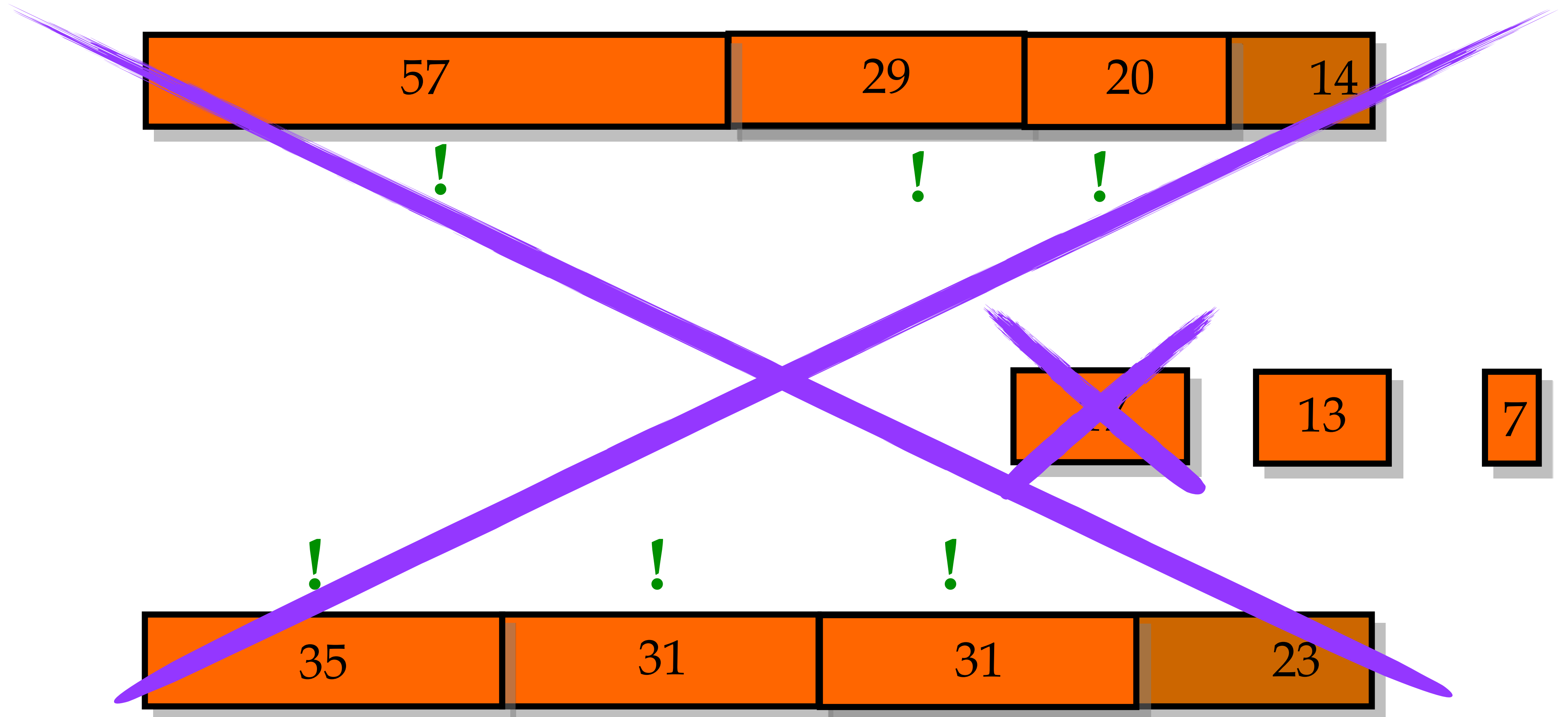
!

!

!

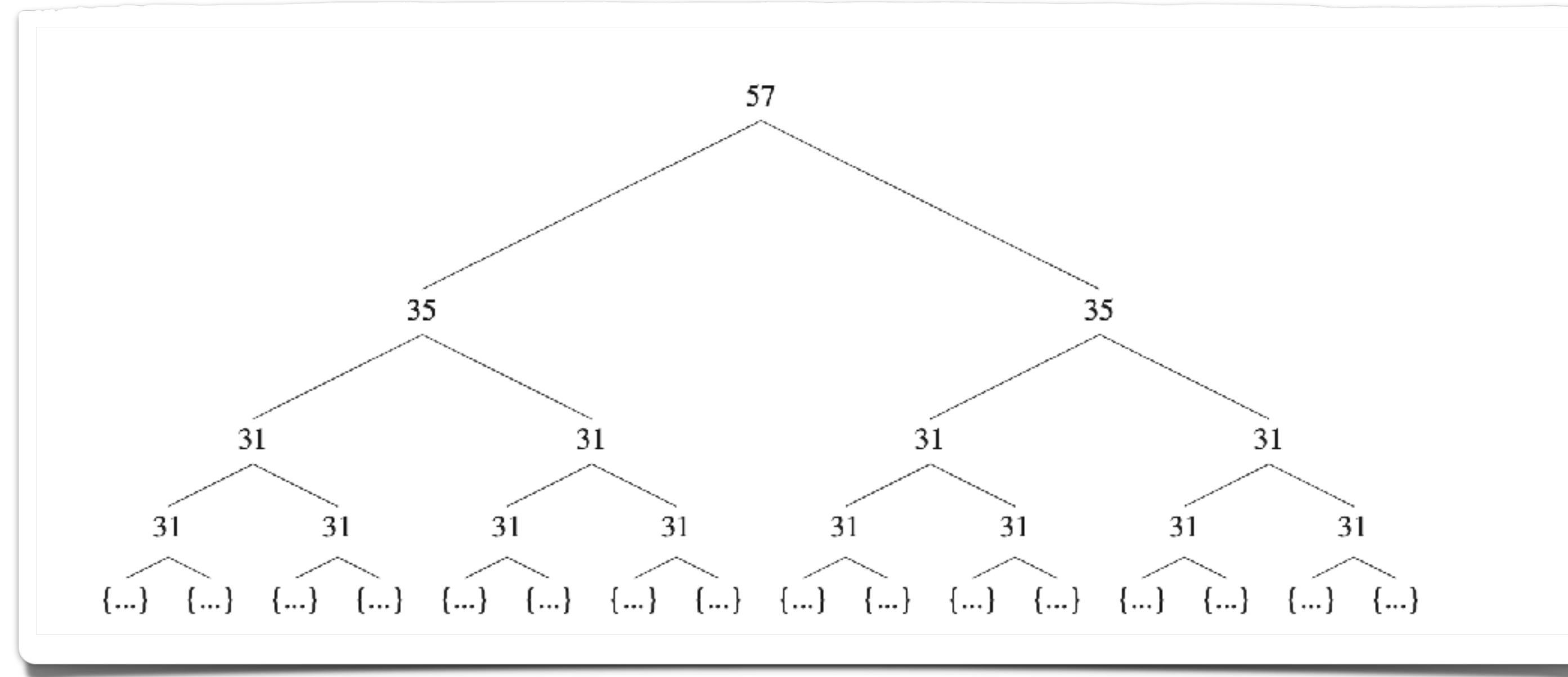


Keine Lösung?!

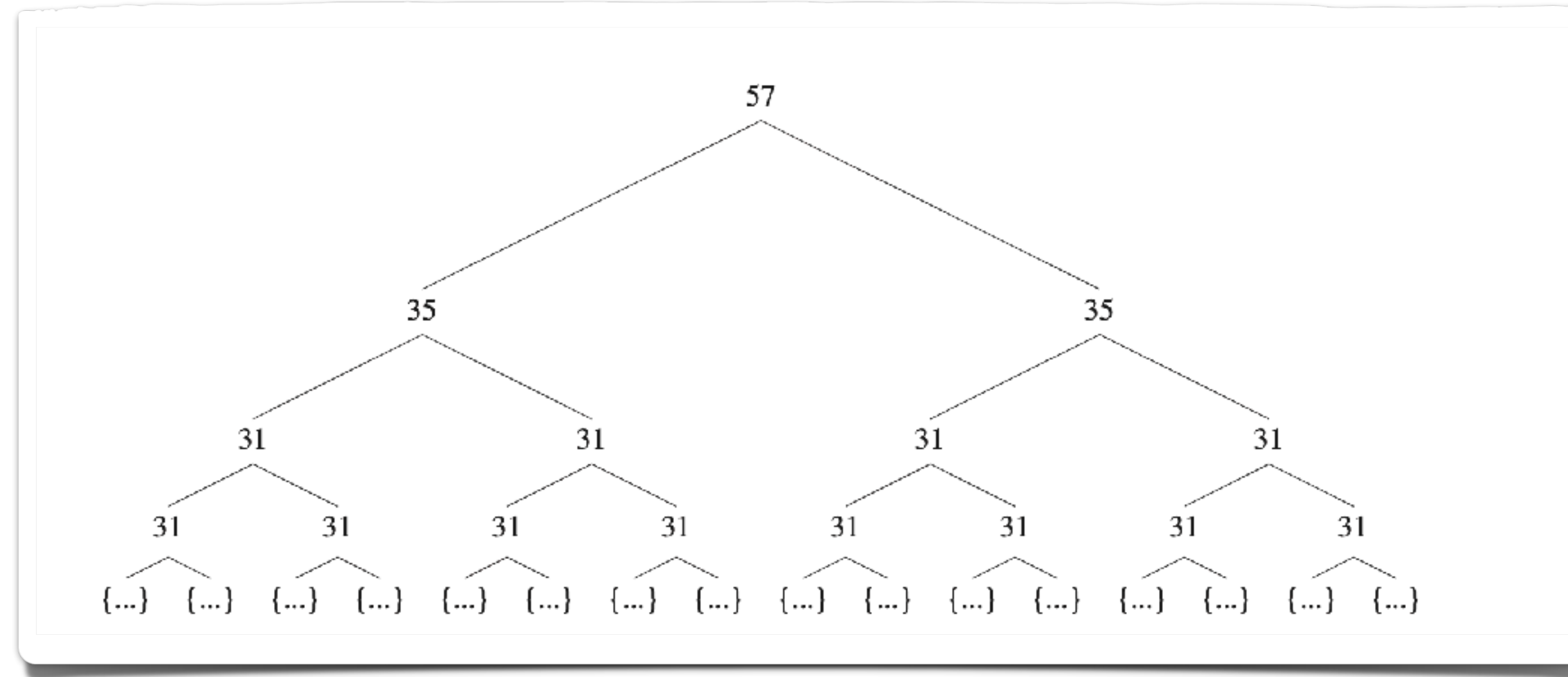


Ideen

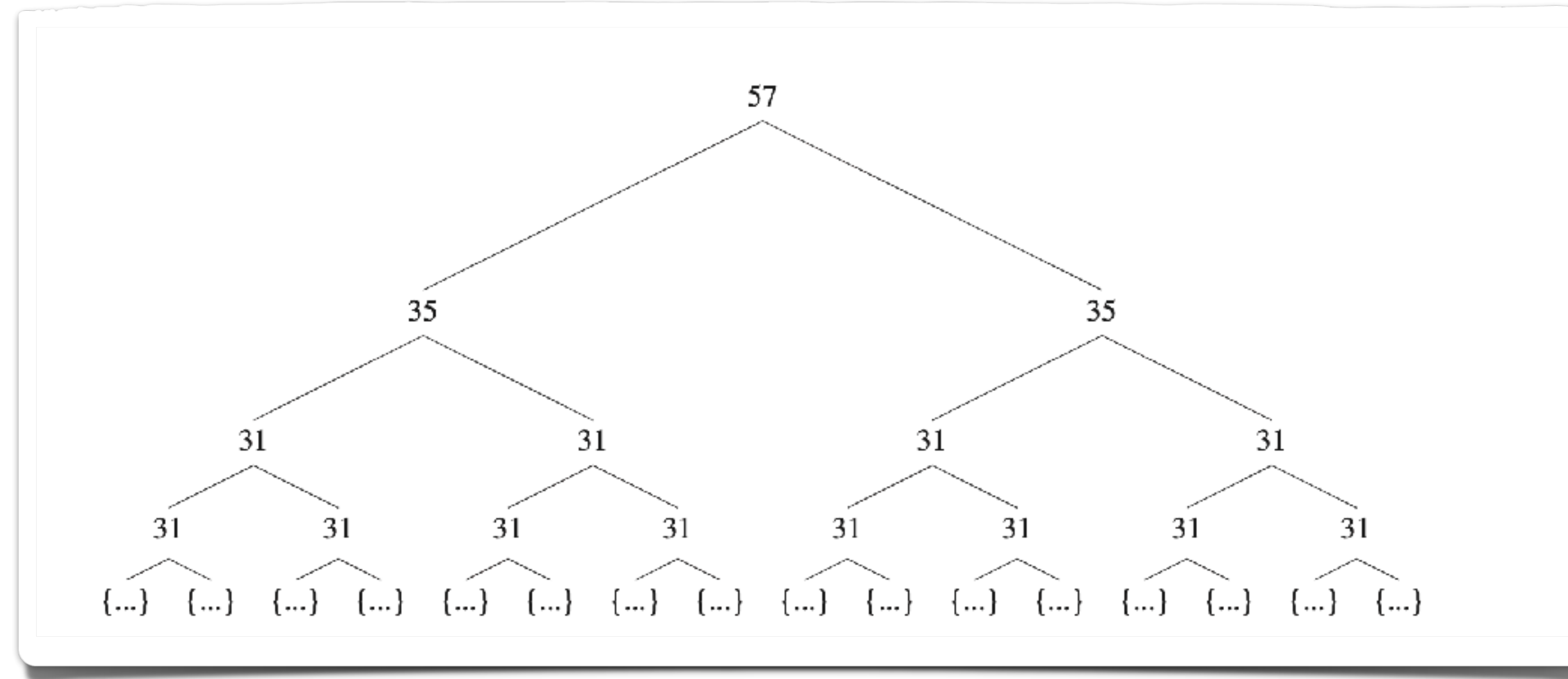
Ideen



Ideen

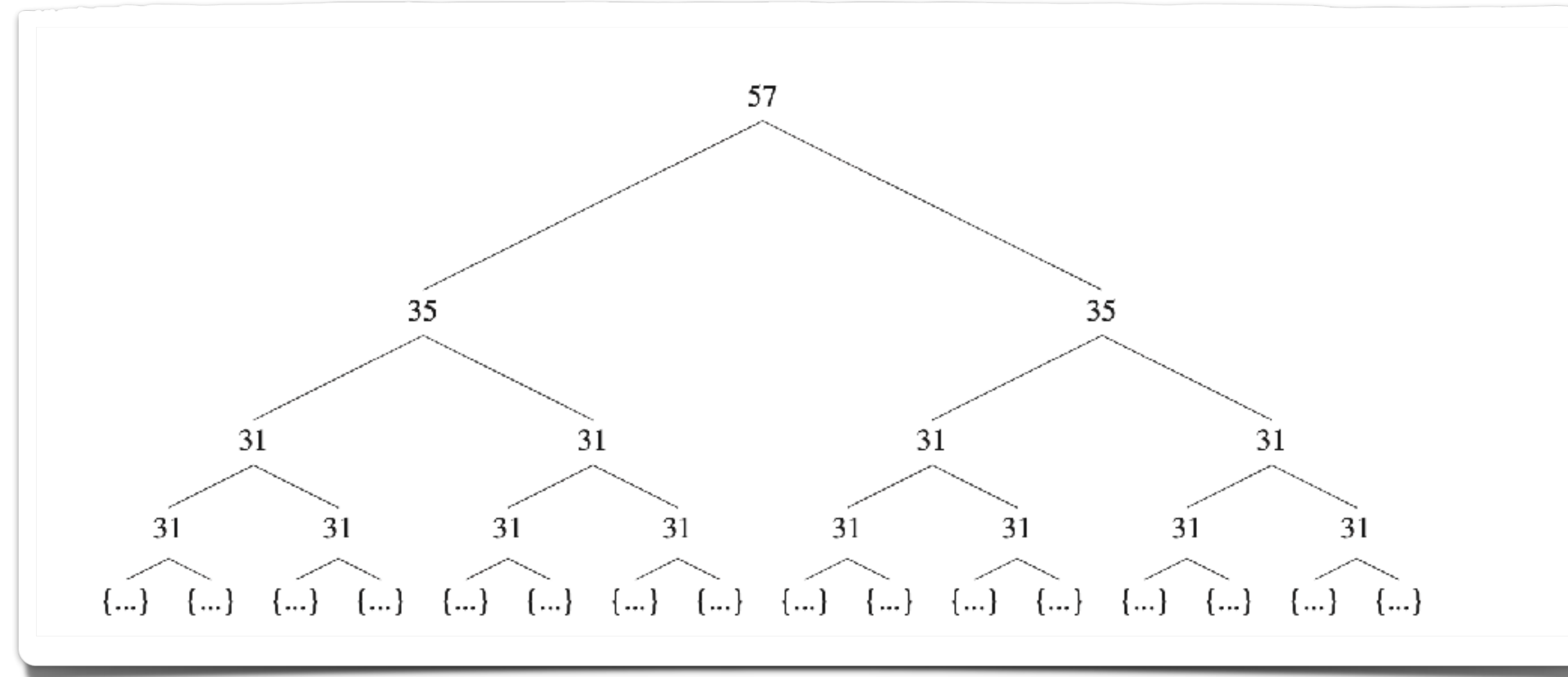


Ideen



- Zeige Unerreichbarkeit von Zahlen

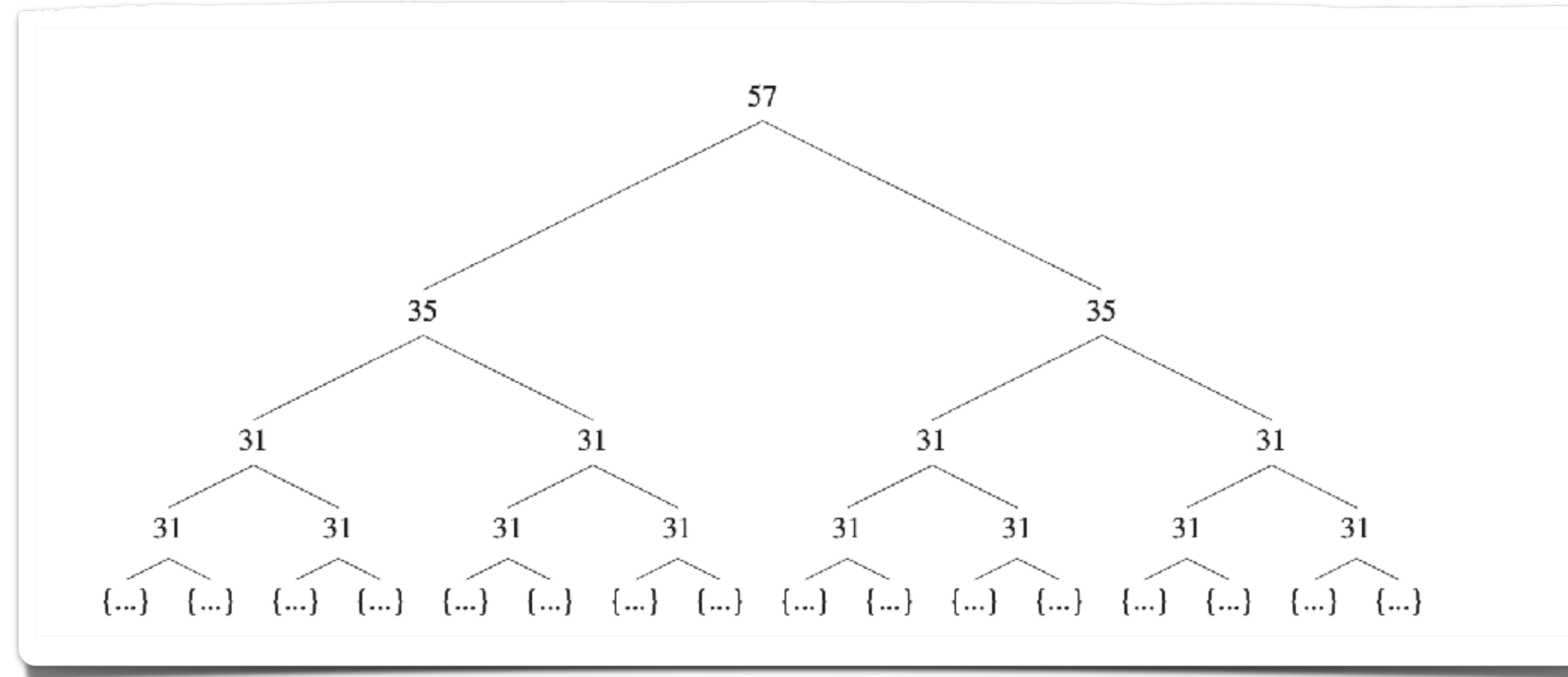
Ideen



- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: SchlieÙe Zahlen aus

Ideen

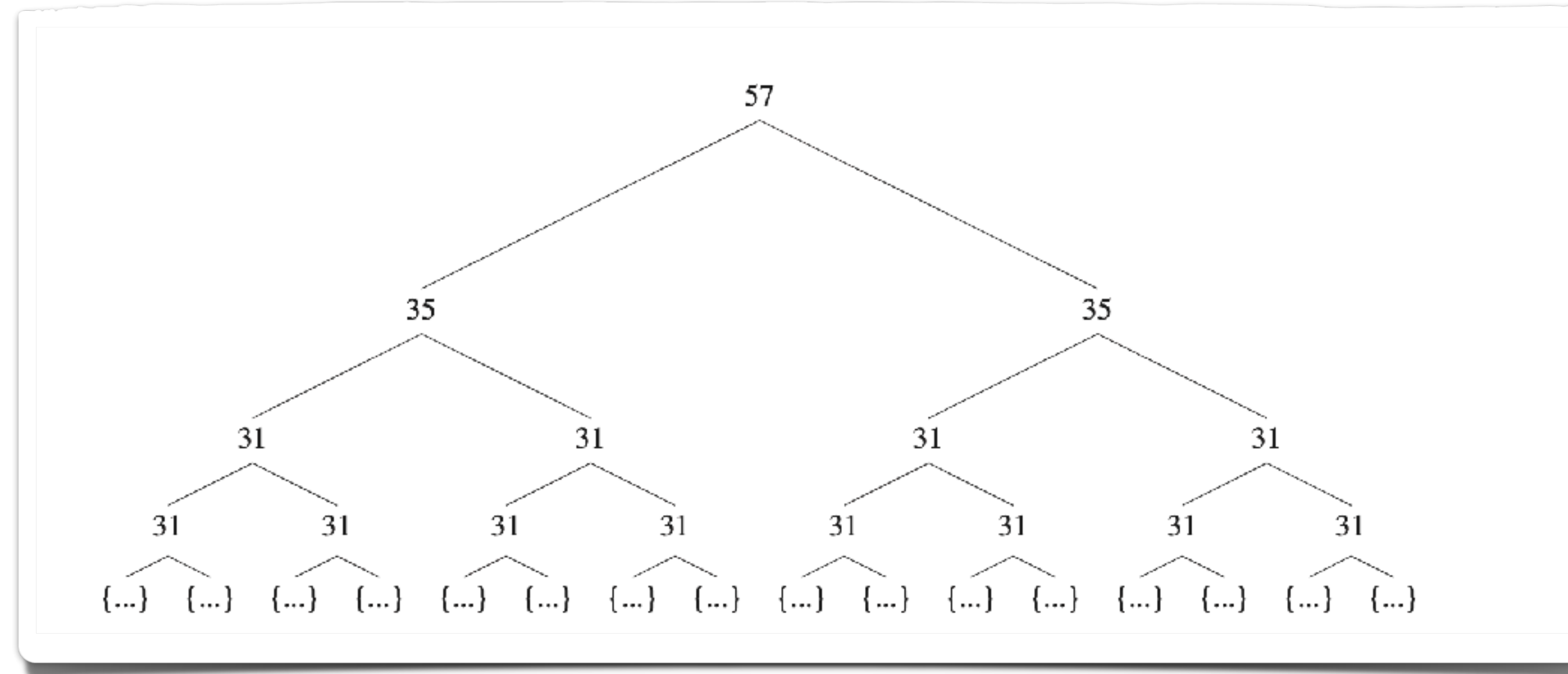
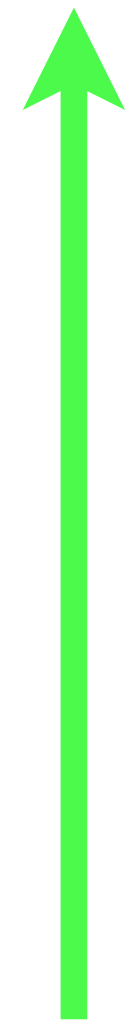
Dynamic
Programming



- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: Schließe Zahlen aus

Ideen

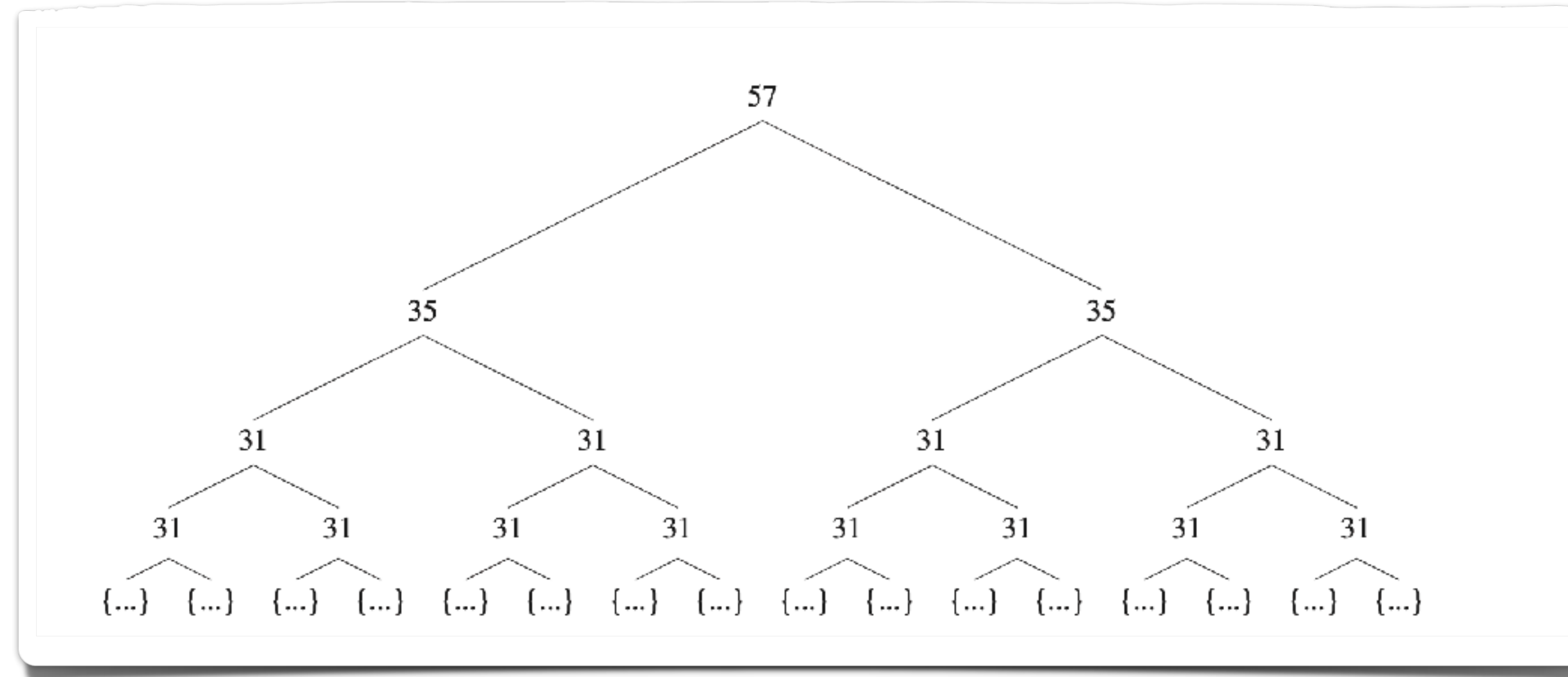
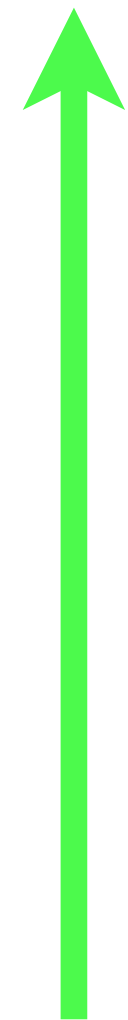
Dynamic
Programming



- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: Schließe Zahlen aus

Ideen

Dynamic
Programming

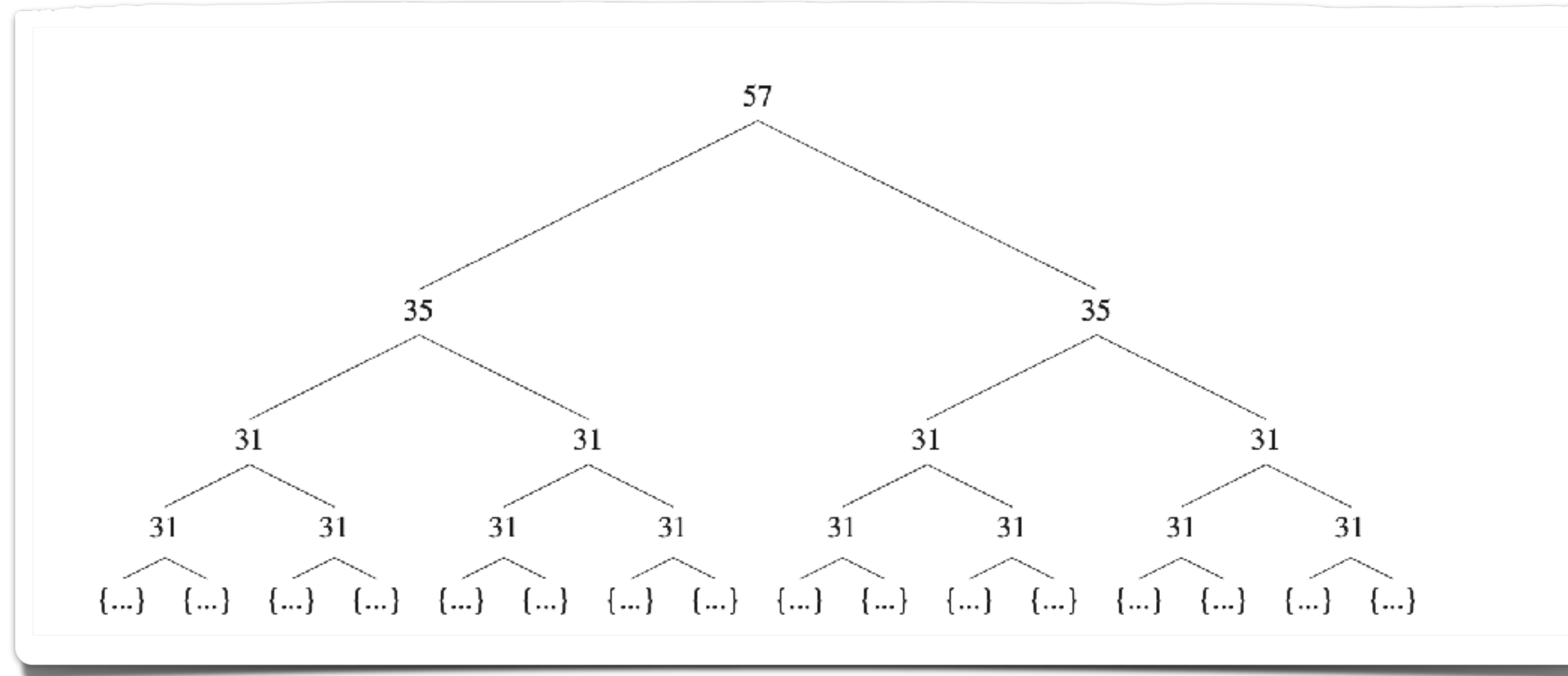
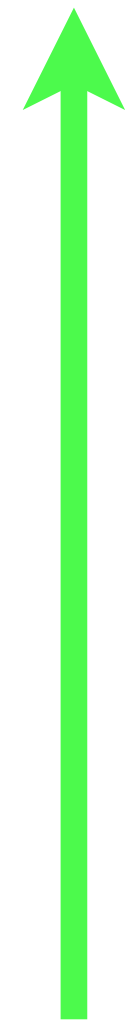


- Bestimme erreichbare Zahlen

- Zeige Unerreichbarkeit von Zahlen
- Rückwärts: SchlieÙe 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

Beispiel 2.1

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

Beispiel

Beispiel 2.1

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

Bestimme alle Zahlen, die als Teilsummen erreichbar sind!

Beispiel

Beispiel 2.1

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

Bestimme alle Zahlen, die als Teilsummen erreichbar sind!

- Bestimme erreichbare Zahlen

Beispiel

Beispiel 2.1

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

Bestimme alle Zahlen, die als Teilsummen erreichbar sind!

- Bestimme erreichbare Zahlen
- Vorwärts: Nimm Zahlen hinzu

Beispiel

Beispiel 2.1

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


Bestimme alle Zahlen, die als Teilsummen erreichbar sind!

- Bestimme erreichbare Zahlen
- Vorwärts: Nimm Zahlen hinzu

Beispiel

Beispiel

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

Beispiel

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

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

Beispiel

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

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

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

Beispiel

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

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

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

Beispiel

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

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

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

Beispiel

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

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

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

Beispiel

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

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

$x \backslash i$	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

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

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

$x \backslash i$	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

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

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

$x \backslash i$	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

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

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

$x \backslash i$	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

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

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

$x \backslash i$	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

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

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

$i \backslash x$	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												
2																			
3																			

Beispiel

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

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

$x \backslash i$	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	0	...	0	0	0	0	0	0	0	0	...	0
2																			
3																			

A blue arrow points to the first column (x=0). A red arrow points down from the cell (0,0) to (1,0). A green arrow points right from the cell (0,0) to (0,7), with a green '+7' written above it.

Beispiel

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

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

$x \backslash i$	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

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

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

$x \backslash i$	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

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

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

$i \backslash x$	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	0	0	0	0	0	...	0
2																			
3																			

Beispiel

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

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

$x \backslash i$	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

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

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

$i \backslash x$	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	0	0	0	0	0	...	0	
2																				
3																				

Beispiel

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

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

$i \backslash x$	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	0	0	0	0	0	...	0	
2																				
3																				

Beispiel

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

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

$i \backslash x$	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	0	0	0	0	0	...	0
2	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	...	0
3	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	...	0

Beispiel

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

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

$i \backslash x$	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	0	0	0	0	0	...	0
2	1																		
3																			

Beispiel

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

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

$i \backslash x$	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	0	0	0	0	0	...	0
2	1	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	...	0
3																			

Beispiel

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

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

$x \backslash i$	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

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

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

$i \backslash x$	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	0	0	0	0	0	...	0	
2	1	0	0	0	0	0	0	1	...	1	0	0	0	0	0	0	0	...	0	
3																				

A blue arrow points to the first column (x=0). A green arrow points from the cell (i=1, x=0) to (i=2, x=13), with a red arrow pointing down to the cell (i=1, x=7) and the text "+13" below it.

Beispiel

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

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

$i \backslash x$	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							1		1									
3																			

Beispiel

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

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

$i \backslash x$	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							1		1										1
3																				

Beispiel

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

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

$i \backslash x$	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	0	1	...	0
3																			

Beispiel

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

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

$i \backslash x$	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	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

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

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

$x \backslash i$	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

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

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

$i \backslash x$	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	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

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

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

$i \backslash x$	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	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

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

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

$i \backslash x$	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	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

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

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

$i \backslash x$	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	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	0	0	0	1	...	0

Beispiel

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

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

$i \backslash x$	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	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

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

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

$i \backslash x$	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	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

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

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

$i \backslash x$	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	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

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

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

$i \backslash x$	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	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	0	0	0	1	...	0

Beispiel

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

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

$i \backslash x$	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	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	0	0	0	1	...	0

Beispiel

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

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

$i \backslash x$	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	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

Beispiel

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

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

$i \backslash x$	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	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

Beispiel

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

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

$i \backslash x$	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	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

Beispiel

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

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

$i \backslash x$	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	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

Beispiel

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

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

$i \backslash x$	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	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

Beispiel

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

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

$i \backslash x$	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	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

Beispiel

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

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

$i \backslash x$	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	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

Beispiel

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

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

$i \backslash x$	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	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

Beispiel

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

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

$i \backslash x$	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	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

Allgemeiner

Allgemeiner

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

Allgemeiner

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

$x \backslash i$	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	0	1	...	0
3																			

Allgemeiner

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

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

$i \backslash x$	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	0	1	...	0
3																			

Allgemeiner

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

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

$x \backslash i$	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	0	1	...	0
3																			

Allgemeiner

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

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

$x \backslash i$	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	0	1	...	0
3																			

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

Allgemeiner

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

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

$x \backslash i$	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	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

Algorithmus 2.2. (Dynamic Programming für SUBSET SUM).

Algorithmus

Algorithmus 2.2. (Dynamic Programming für SUBSET SUM).

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

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

:

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

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

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

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

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

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

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

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

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

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

$x \backslash i$	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	0	...	0	0	0	+13	0	0	0	0	...	0
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	0
3																			

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

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

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

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

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

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

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

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

$x \backslash i$	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	0	...	0	0	0	+13	0	0	0	0	...	0	
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	1	...	0	
3																				

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

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

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

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

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

$x \backslash i$	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	0	...	0	0	0	+13	0	0	0	0	...	0
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	1
3																			

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

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

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

$i \backslash x$	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	0	...	0	0	0	+13	0	0	0	0	...	0
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	1
3																			

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

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

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

$x \backslash i$	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	0	...	0	0	0	+13	0	0	0	0	...	0	
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	0	
3																				

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

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

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

- 4: **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	0	...	0	0	0	+13	0	0	0	0	...	0	
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	0	
3																				

Algorithmus

Algorithmus 2.2. (Dynamic Programming für SUBSET SUM).

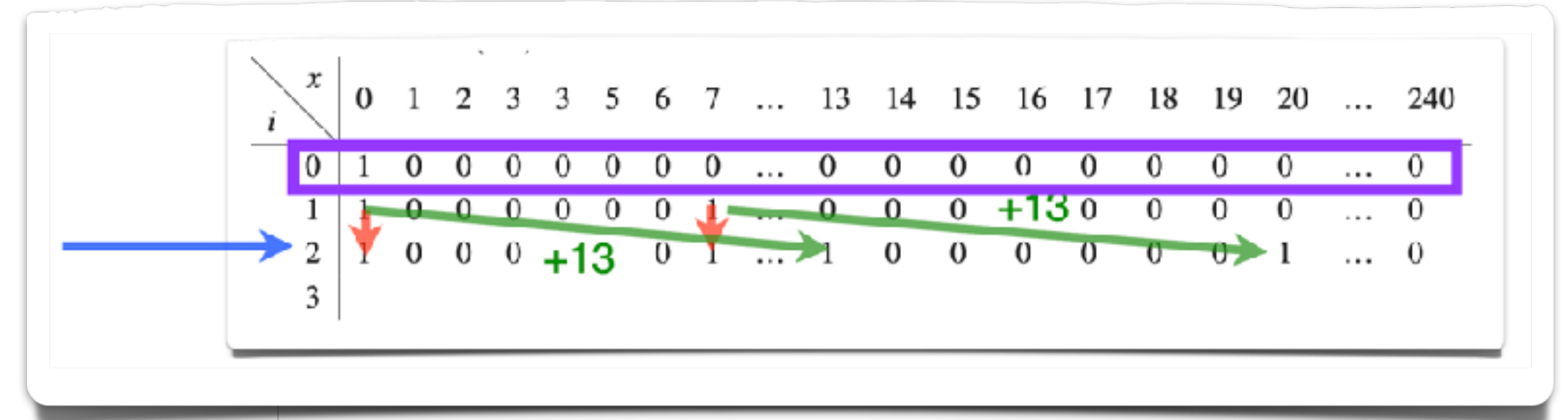
Eingabe: Zahlen z_1, \dots, z_n mit $\sum_{i=1}^n z_i = Z$

Ausgabe: Boolesche Funktion $\mathcal{S} : \{0, \dots, Z\} \times \{0, \dots, n\} \rightarrow \{0, 1\}$

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

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

4: **for** ($i = 1$) **to** n **do**
5: **for** ($x = 0$) **to** $z_i - 1$ **do**



Algorithmus

Algorithmus 2.2. (Dynamic Programming für SUBSET SUM).

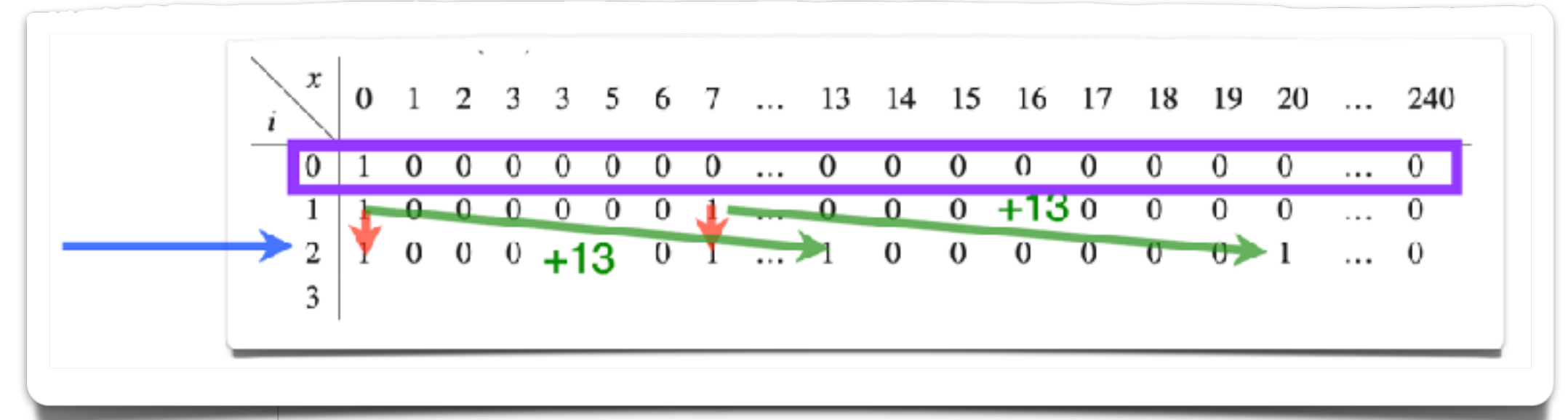
Eingabe: Zahlen z_1, \dots, z_n mit $\sum_{i=1}^n z_i = Z$

Ausgabe: Boolesche Funktion $\mathcal{S} : \{0, \dots, Z\} \times \{0, \dots, n\} \rightarrow \{0, 1\}$

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

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

4: **for** ($i = 1$) **to** n **do**
5: **for** ($x = 0$) **to** $z_i - 1$ **do**
6: $\mathcal{S}(x, i) := \mathcal{S}(x, i - 1);$



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

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

```

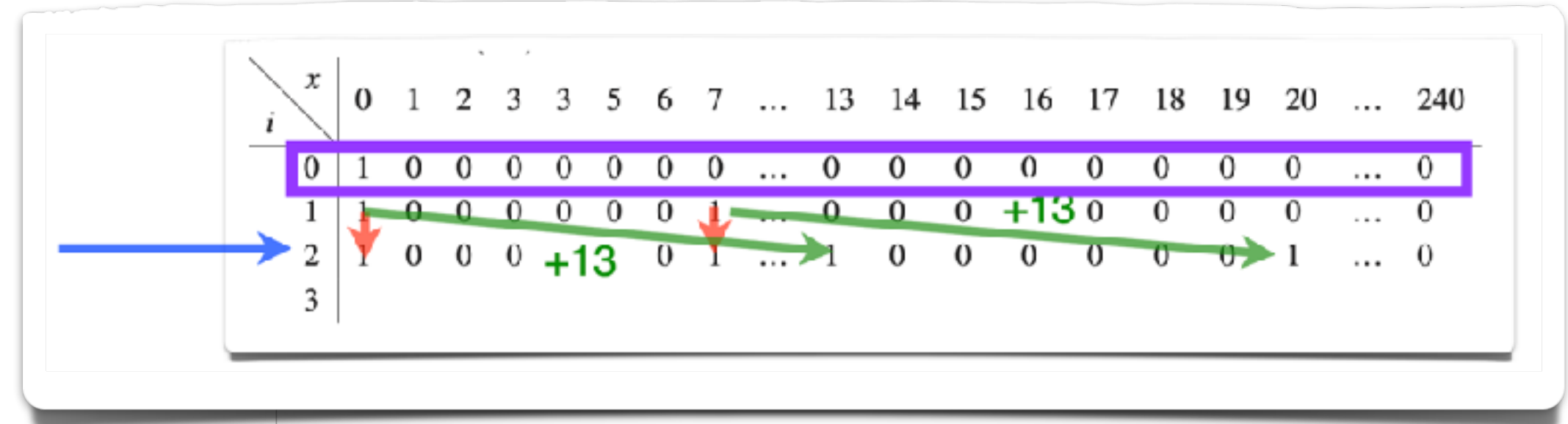
1:  $\mathcal{S}(0, 0) := 1$ 
2: for  $(x=1)$  to  $Z$  do
3:    $\mathcal{S}(x, 0) := 0;$ 

```

```

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

```



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

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

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

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

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

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

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

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

7: **for** ($x = z_i$) **to** Z **do**

8: **if** ($(\mathcal{S}(x, i - 1) = 1)$ **OR** $(\mathcal{S}((x - z_i), i - 1) = 1)$) **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	0	...	0	0	0	+13	0	0	0	0	...	0	
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	0	
3																				

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

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

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

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

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

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

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

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

7: **for** ($x = z_i$) **to** Z **do**

8: **if** ($\mathcal{S}(x, i - 1) = 1$ **OR** ($\mathcal{S}((x - z_i), i - 1) = 1$)) **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	0	...	0	0	0	+13	0	0	0	0	...	0
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	0
3	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	...	0

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

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

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

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

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

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

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

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

7: **for** ($x = z_i$) **to** Z **do**

8: **if** ($\mathcal{S}(x, i - 1) = 1$ **OR** ($\mathcal{S}((x - z_i), i - 1) = 1$)) **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	0	...	0	0	0	+13	0	0	0	0	...	0
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	0
3	1	0	0	0	0	0	0	1	...	1	0	0	0	0	0	0	0	...	0

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

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

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

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

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

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

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

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

7: **for** ($x = z_i$) **to** Z **do**

8: **if** ($\mathcal{S}(x, i - 1) = 1$ OR $\mathcal{S}((x - z_i), i - 1) = 1$) **then**

9: $\mathcal{S}(x, i) := 1;$

$i \backslash x$	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	0	...	0	0	0	+13	0	0	0	0	...	0
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	0
3	1	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	...	0

Algorithmus

Algorithmus 2.2 . (Dynamic Programming für SUBSET SUM).

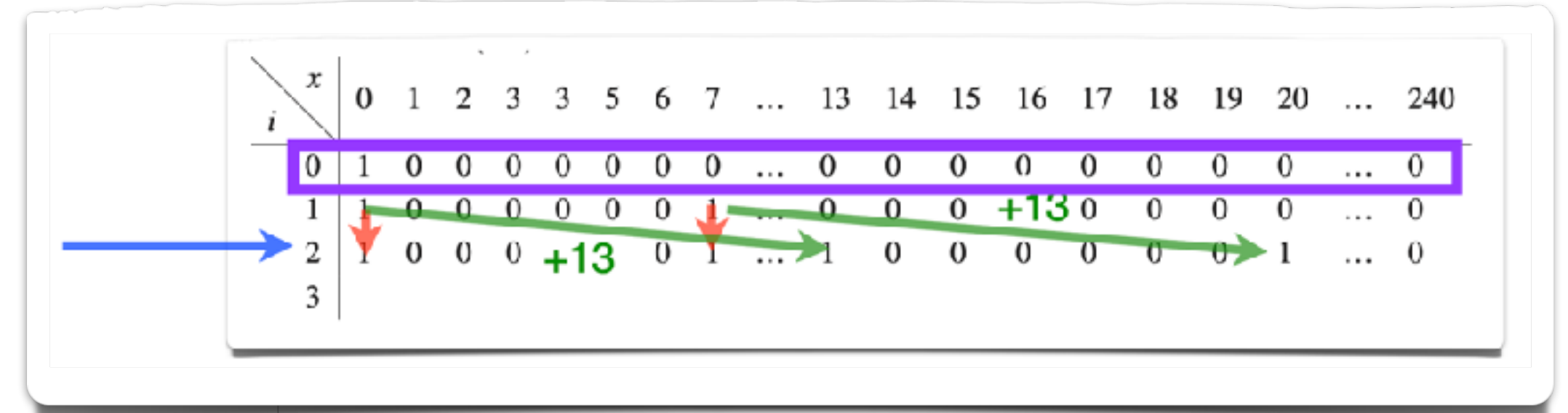
Eingabe: Zahlen z_1, \dots, z_n mit $\sum_{i=1}^n z_i = Z$

Ausgabe: Boolesche Funktion $\mathcal{S} : \{0, \dots, Z\} \times \{0, \dots, n\} \rightarrow \{0, 1\}$

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

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

4: **for** ($i = 1$) **to** n **do**
 5: **for** ($x = 0$) **to** $z_i - 1$ **do**
 6: $\mathcal{S}(x, i) := \mathcal{S}(x, i - 1);$
 7: **for** ($x = z_i$) **to** Z **do**
 8: **if** ($\mathcal{S}(x, i - 1) = 1$ **OR** ($\mathcal{S}((x - z_i), i - 1) = 1$)) **then**
 9: $\mathcal{S}(x, i) := 1;$
 10: **else**
 11: $\mathcal{S}(x, i) := 0;$



Algorithmus

Algorithmus 2.2 . (Dynamic Programming für SUBSET SUM).

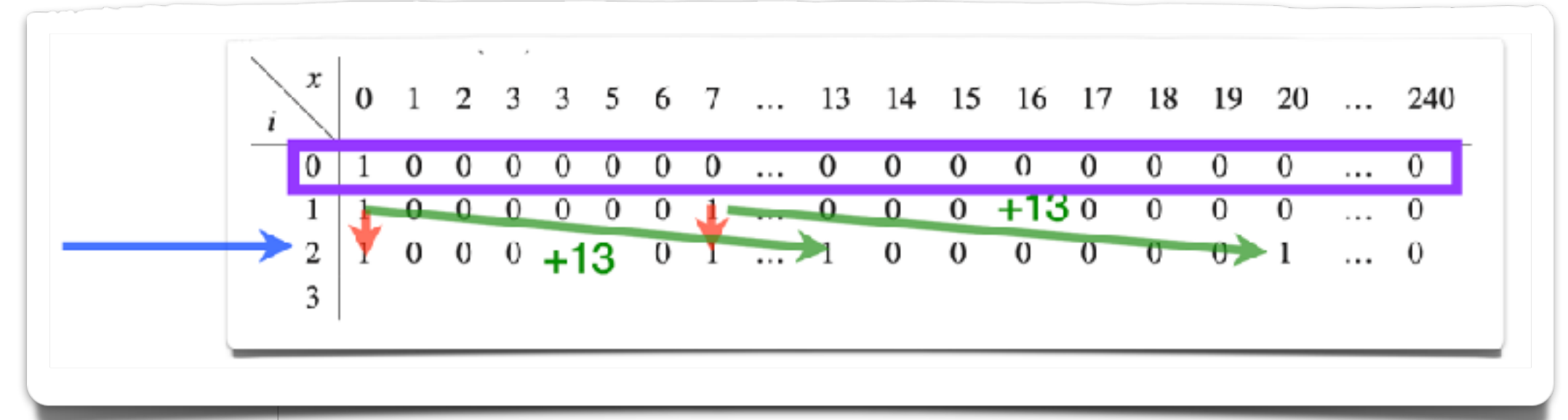
Eingabe: Zahlen z_1, \dots, z_n mit $\sum_{i=1}^n z_i = Z$

Ausgabe: Boolesche Funktion $\mathcal{S} : \{0, \dots, Z\} \times \{0, \dots, n\} \rightarrow \{0, 1\}$

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

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

4: **for** ($i = 1$) **to** n **do**
 5: **for** ($x = 0$) **to** $z_i - 1$ **do**
 6: $\mathcal{S}(x, i) := \mathcal{S}(x, i - 1);$
 7: **for** ($x = z_i$) **to** Z **do**
 8: **if** ($\mathcal{S}(x, i - 1) = 1$ **OR** ($\mathcal{S}((x - z_i), i - 1) = 1$)) **then**
 9: $\mathcal{S}(x, i) := 1;$
 10: **else**
 11: $\mathcal{S}(x, i) := 0;$
 12: **return**



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.

$i \backslash x$	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 .

$i \backslash x$	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 $n=0$.

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

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

Induktionsschritt: Erzeugung von x mit z_1, \dots, 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																				
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: $\mathcal{S}(x, i-1)$ korrekt für alle i .

Induktionsschritt: Erzeugung von x mit z_1, \dots, z_i .

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

Induktionsschritt: Erzeugung von x mit z_1, \dots, z_i .

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

Induktionsschritt: Erzeugung von x mit z_1, \dots, z_i .

(I) z_i wird nicht verwendet

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

Induktionsschritt: Erzeugung von x mit z_1, \dots, z_i .

(I) z_i wird nicht verwendet

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

Induktionsschritt: Erzeugung von x mit z_1, \dots, z_i .

(I) z_i wird nicht verwendet

(II) z_i wird verwendet

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

Induktionsschritt: Erzeugung von x mit z_1, \dots, z_i .

(I) z_i wird nicht verwendet

(II) z_i wird verwendet

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

Induktionsschritt: 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

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

Induktionsschritt: 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

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

Induktionsschritt: 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.

$x \backslash i$	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 .

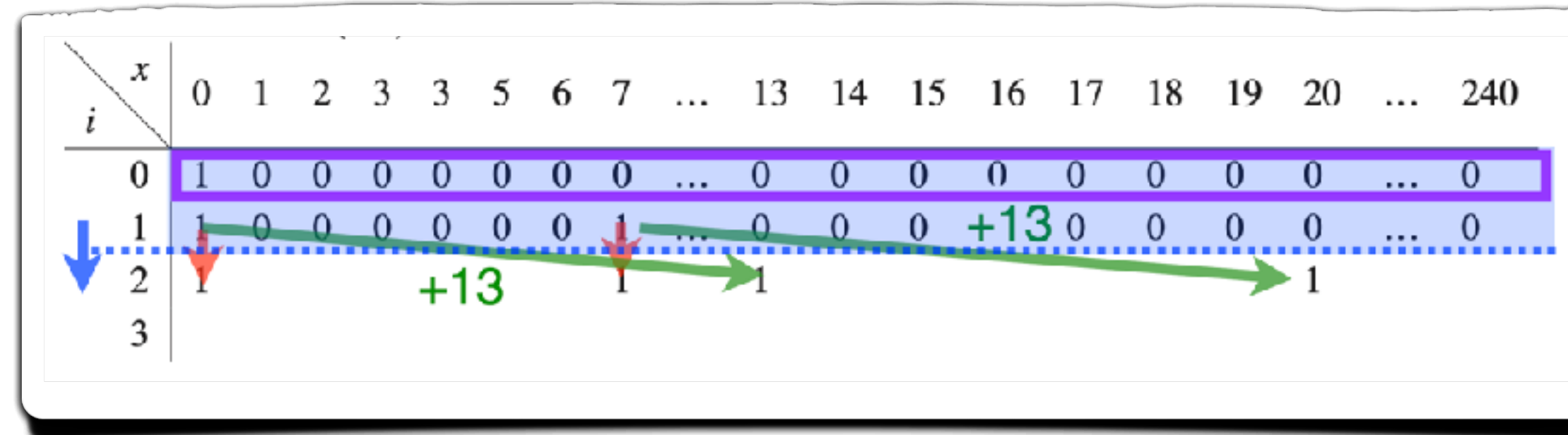
Induktionsanfang: Initialisierung korrekt für $n=0$.

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

Induktionsschritt: 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.



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, \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.

$x \backslash i$	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	0	...	0	0	0	+13	0	0	0	0	...	0
2	1	0	0	0	+13	0	1	...	1	0	0	0	0	0	0	0	0	...	1
3																			

Laufzeit

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

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

```
1:  $\mathcal{S}(0, 0) := 1$ 
2: for ( $x=1$ ) to  $Z$  do
3:    $\mathcal{S}(x, 0) := 0$ ;
4: for ( $i = 1$ ) to  $n$  do
5:   for ( $x = 0$ ) to  $z_i - 1$  do
6:      $\mathcal{S}(x, i) := \mathcal{S}(x, i - 1)$ ;
7:   for ( $x = z_i$ ) to  $Z$  do
8:     if ( $(\mathcal{S}(x, i - 1) = 1)$  OR  $(\mathcal{S}(x - z_i, i - 1) = 1)$ ) then
9:        $\mathcal{S}(x, i) := 1$ ;
10:    else
11:       $\mathcal{S}(x, i) := 0$ ;
12: 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 $\mathcal{S} : \{0, \dots, Z\} \times \{0, \dots, n\} \rightarrow \{0, 1\}$

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

```
1:  $\mathcal{S}(0, 0) := 1$ 
2: for ( $x=1$ ) to  $Z$  do
3:    $\mathcal{S}(x, 0) := 0;$ 
4: for ( $i = 1$ ) to  $n$  do
5:   for ( $x = 0$ ) to  $z_i - 1$  do
6:      $\mathcal{S}(x, i) := \mathcal{S}(x, i - 1);$ 
7:   for ( $x = z_i$ ) to  $Z$  do
8:     if ( $(\mathcal{S}(x, i - 1) = 1)$  OR  $(\mathcal{S}(x - z_i, i - 1) = 1)$ ) then
9:        $\mathcal{S}(x, i) := 1;$ 
10:    else
11:       $\mathcal{S}(x, i) := 0;$ 
12: 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 $\mathcal{S} : \{0, \dots, Z\} \times \{0, \dots, n\} \rightarrow \{0, 1\}$

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

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

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

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

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

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

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

7: **for** ($x = z_i$) **to** Z **do**

8: **if** ($(\mathcal{S}(x, i - 1) = 1)$ **OR** $(\mathcal{S}((x - z_i), i - 1) = 1)$) **then**

9: $\mathcal{S}(x, i) := 1;$

10: **else**

11: $\mathcal{S}(x, i) := 0;$

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

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

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

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

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

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

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

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

7: **for** ($x = z_i$) **to** Z **do**

8: **if** ($(\mathcal{S}(x, i - 1) = 1)$ **OR** $(\mathcal{S}((x - z_i), i - 1) = 1)$) **then**

9: $\mathcal{S}(x, i) := 1;$

10: **else**

11: $\mathcal{S}(x, i) := 0;$

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

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

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

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

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

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

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

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

7: **for** ($x = z_i$) **to** Z **do**

8: **if** ($(\mathcal{S}(x, i - 1) = 1)$ **OR** $(\mathcal{S}((x - z_i), i - 1) = 1)$) **then**

9: $\mathcal{S}(x, i) := 1;$

10: **else**

11: $\mathcal{S}(x, i) := 0;$

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

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

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

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

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

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

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

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

7: **for** ($x = z_i$) **to** Z **do**

8: **if** ($(\mathcal{S}(x, i - 1) = 1)$ **OR** $(\mathcal{S}((x - z_i), i - 1) = 1)$) **then**

9: $\mathcal{S}(x, i) := 1;$

10: **else**

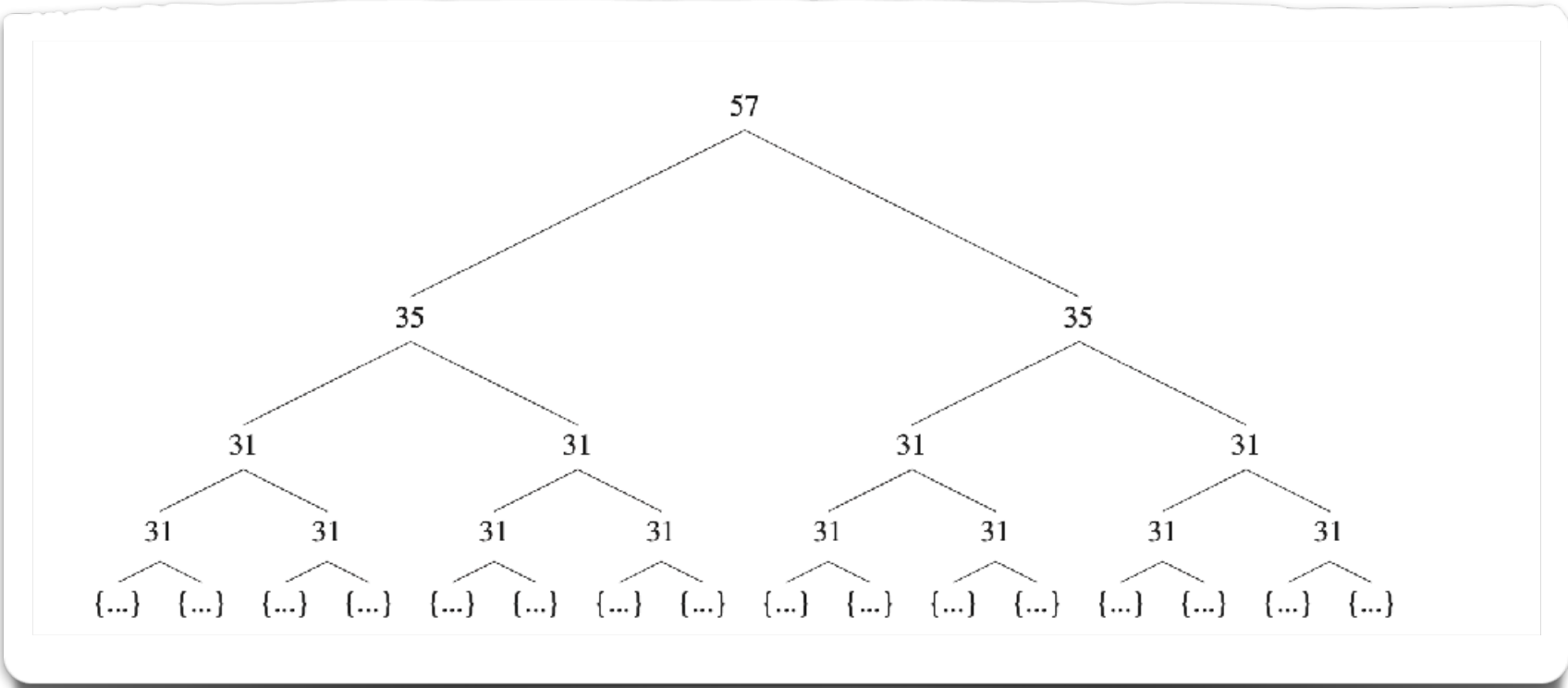
11: $\mathcal{S}(x, i) := 0;$

12: **return**

$O(nZ)$

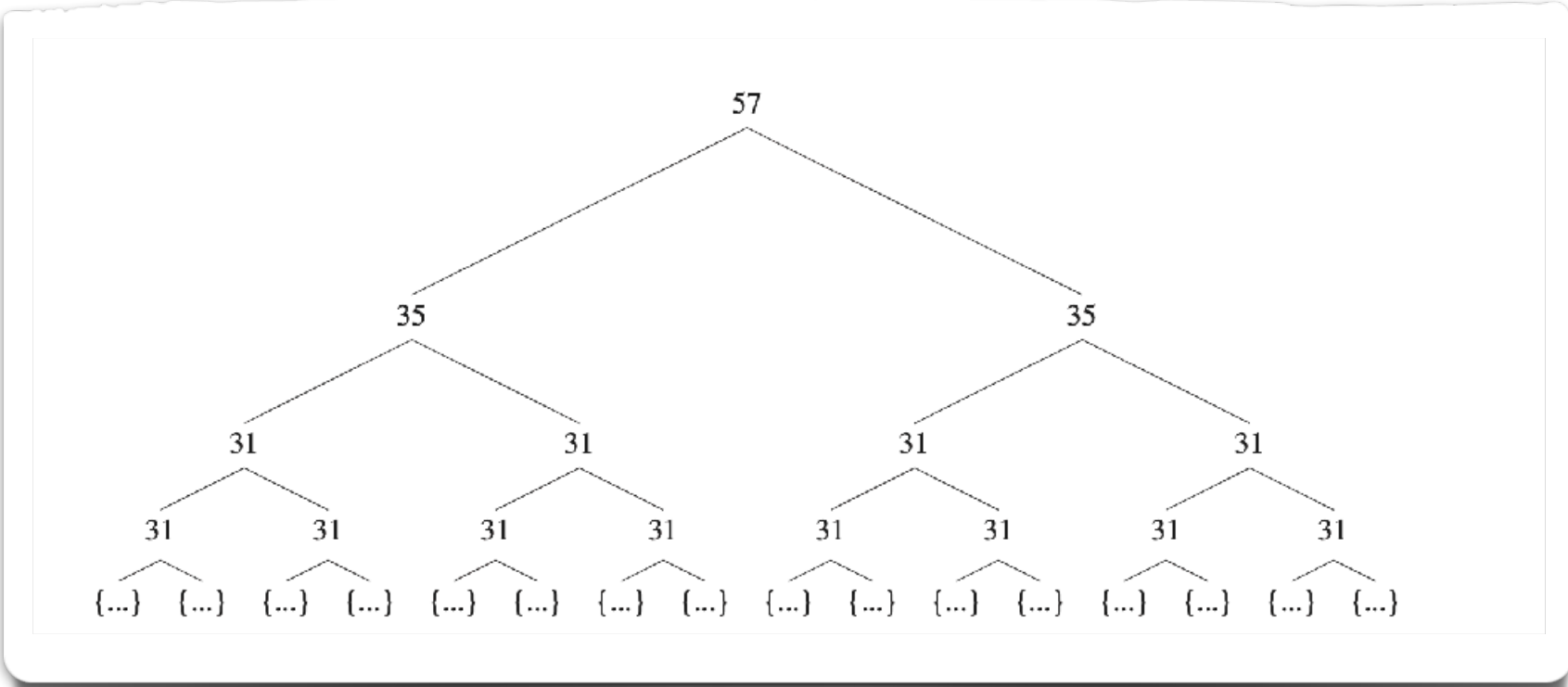
Effizienz?!

Effizienz?!



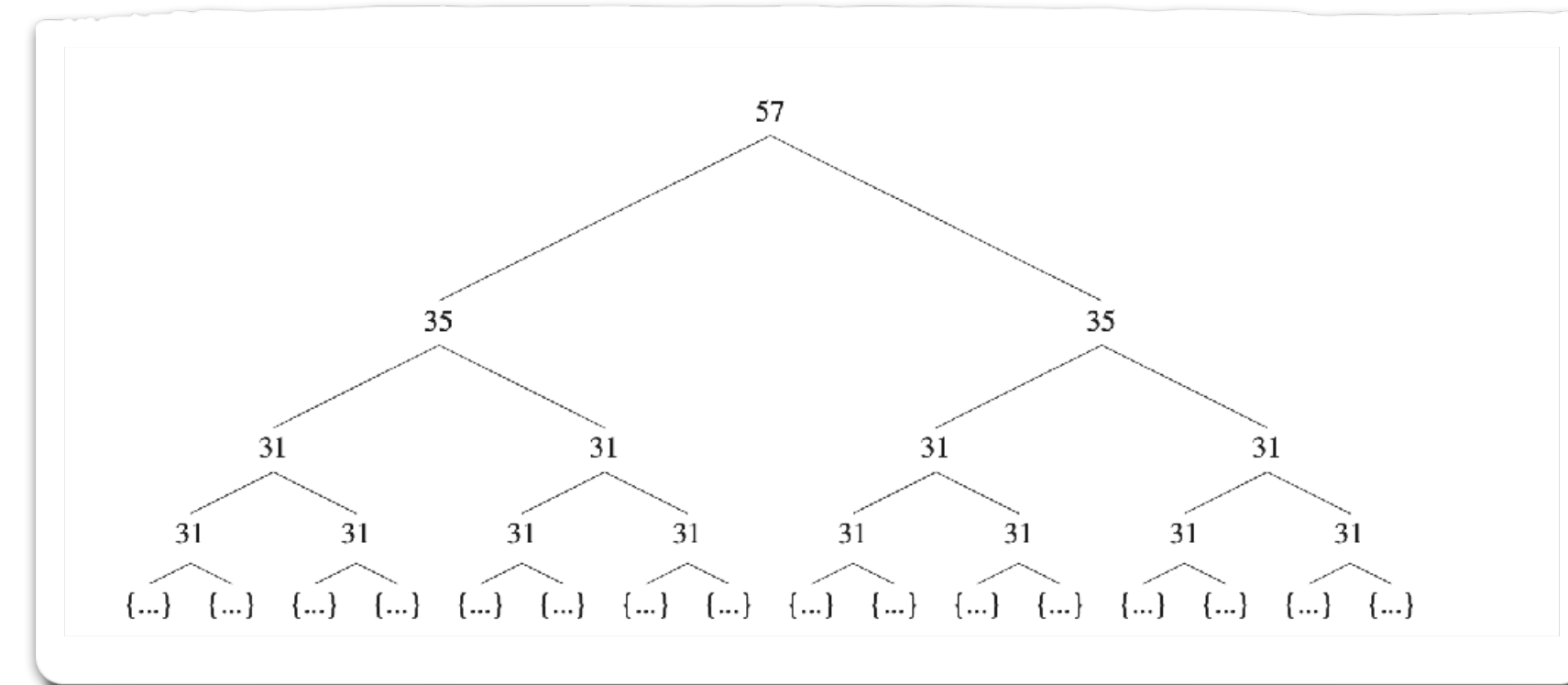
Effizienz?!

$$2^n$$



Effizienz?!

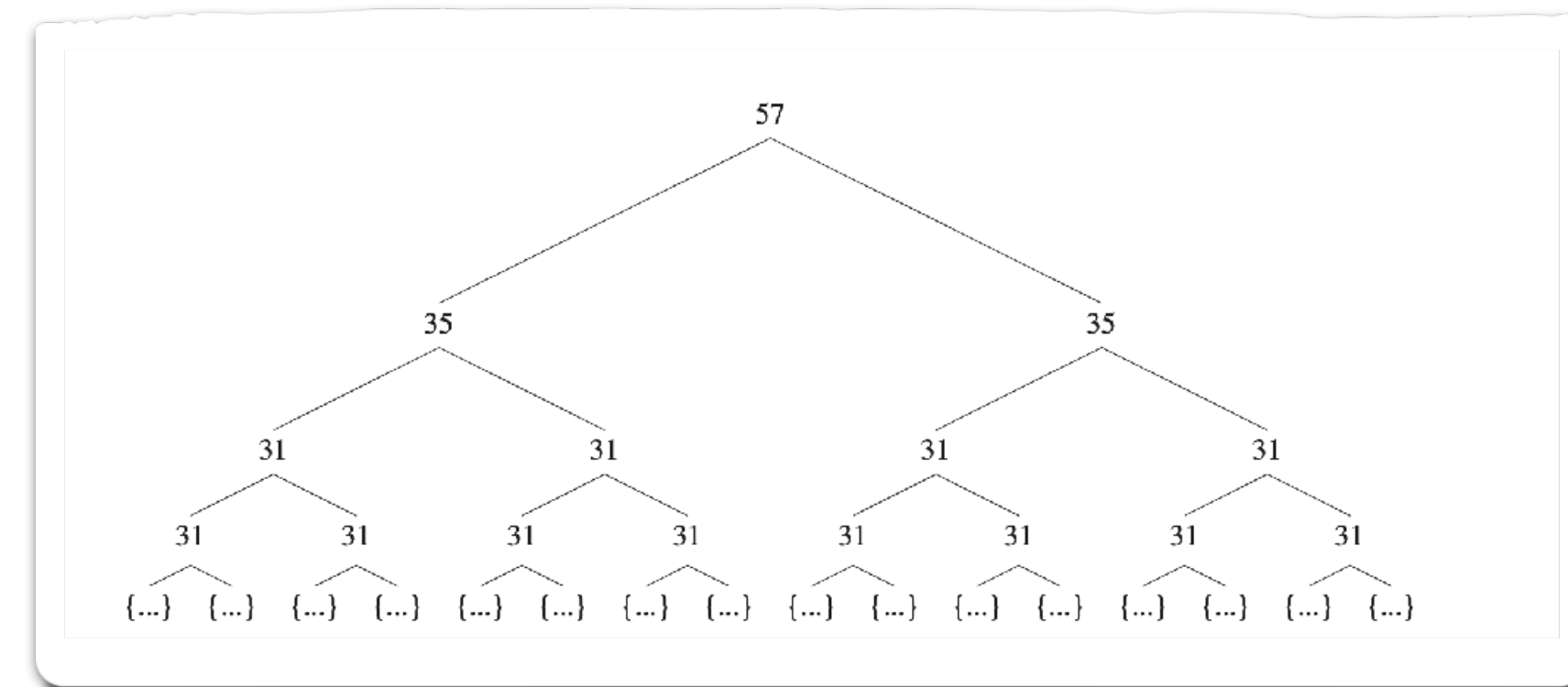
$$2^n$$



- Exponentiell viele Fälle!

Effizienz?!

$$2^n$$

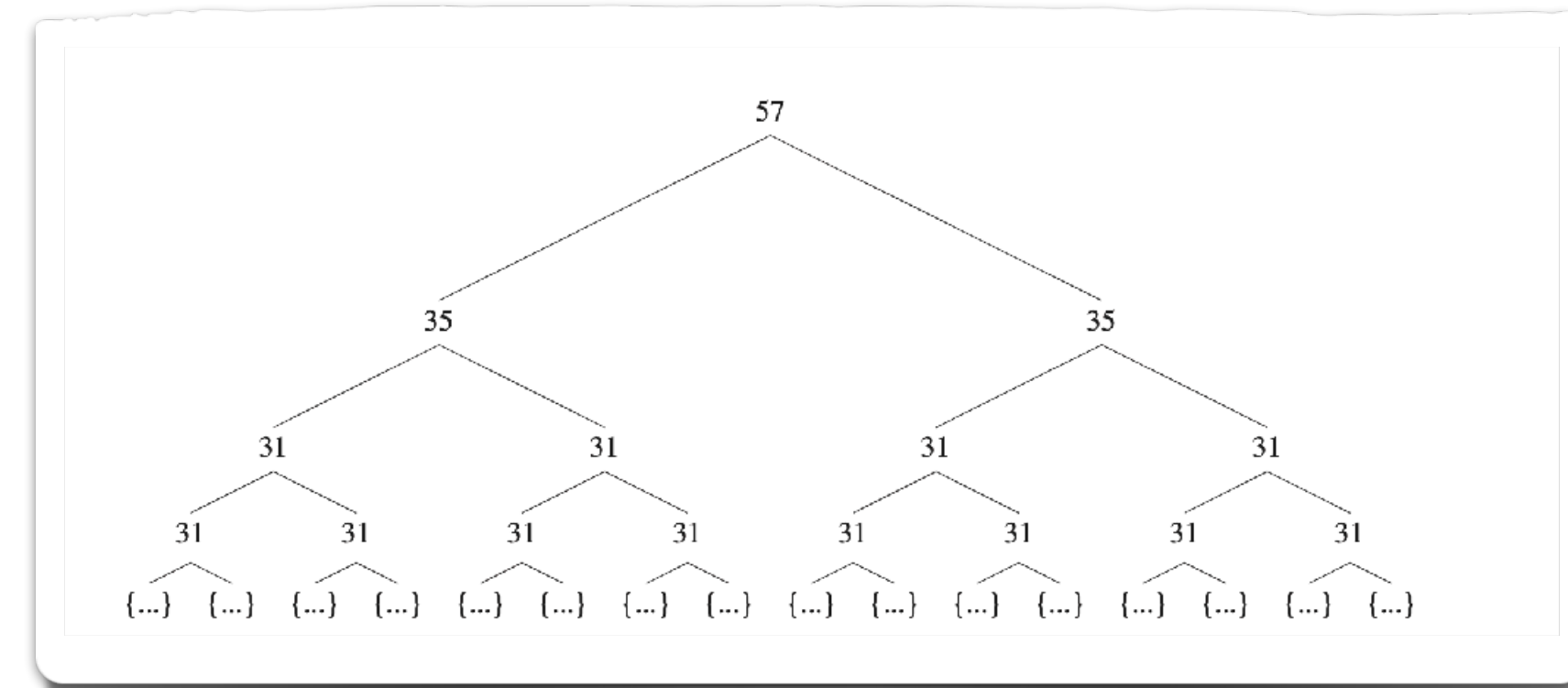


- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

$$O(nZ)$$

$$2^n$$



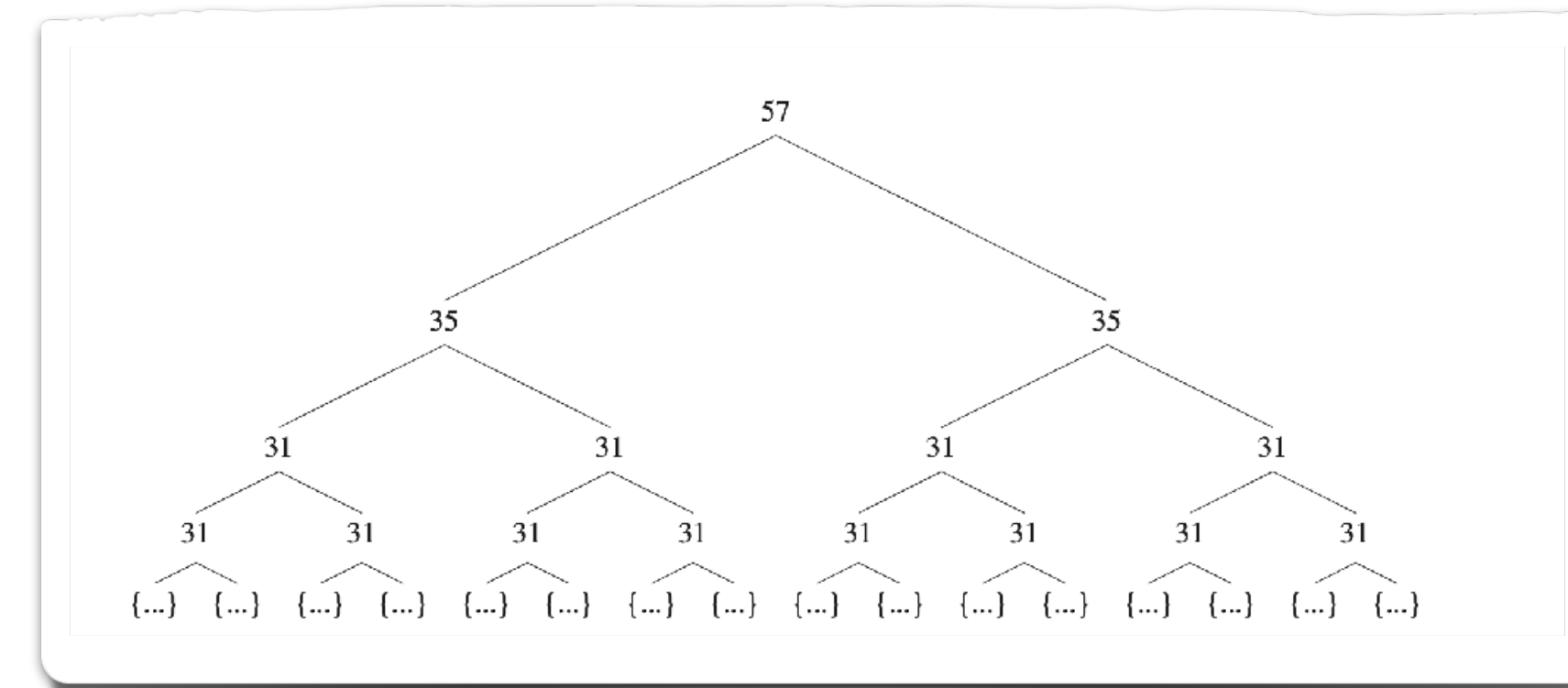
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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																				

$$2^n$$



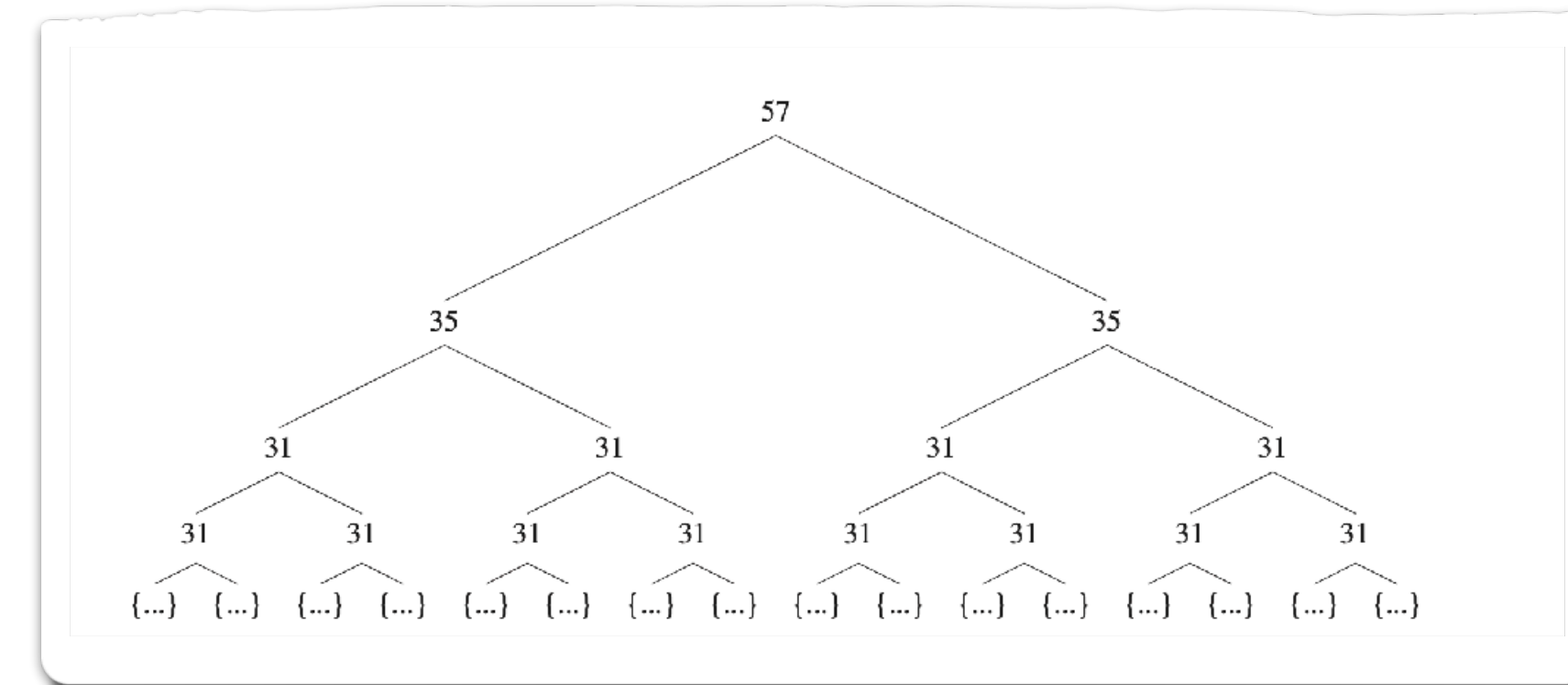
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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																				

$$2^n$$



1

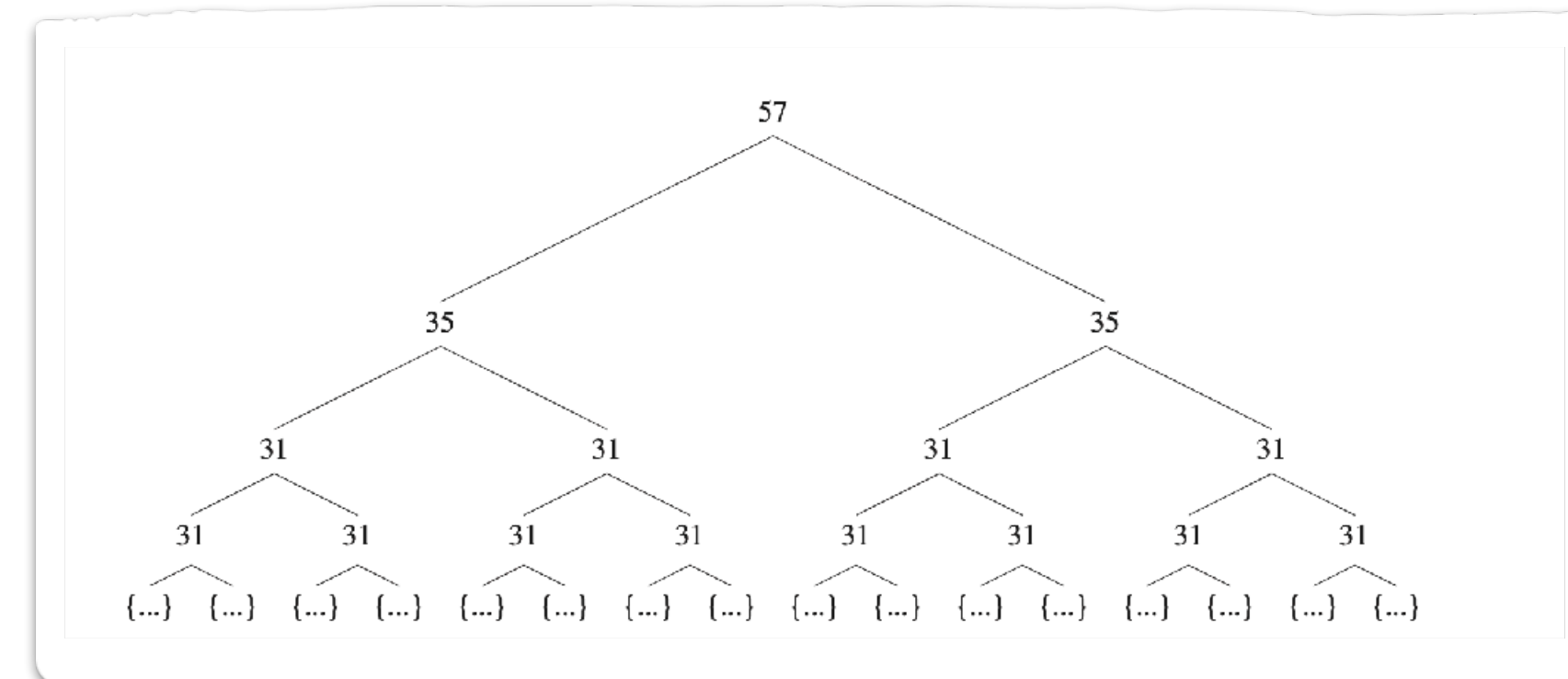
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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	+7	0	...	0	0	0	0	0	0	0	0	...	0
1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	...	0
2																			
3																			

$$2^n$$



1

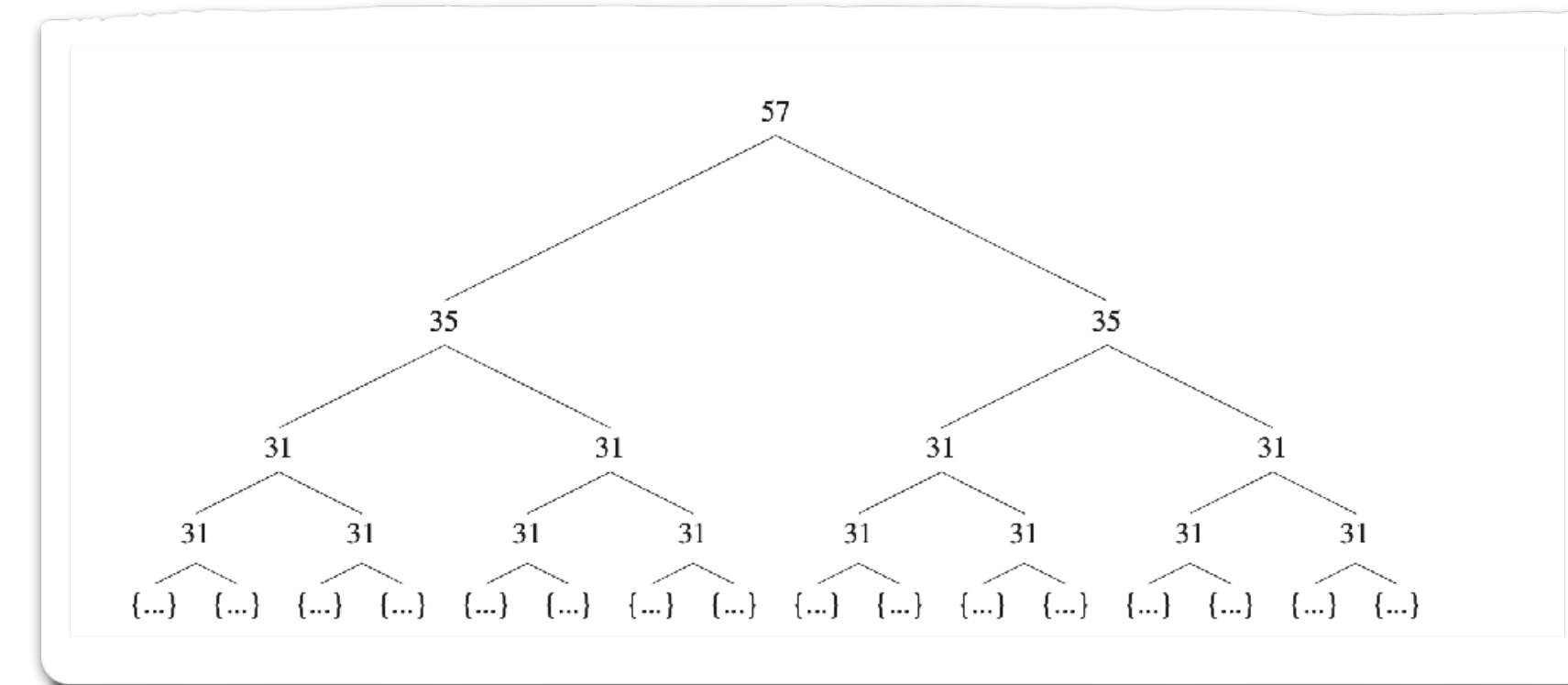
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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	+7	0	0	...	0	0	0	0	0	0	0	...	0	
1	1	0	0	0	0	0	0	0	1											
2																				
3																				

$$2^n$$



2

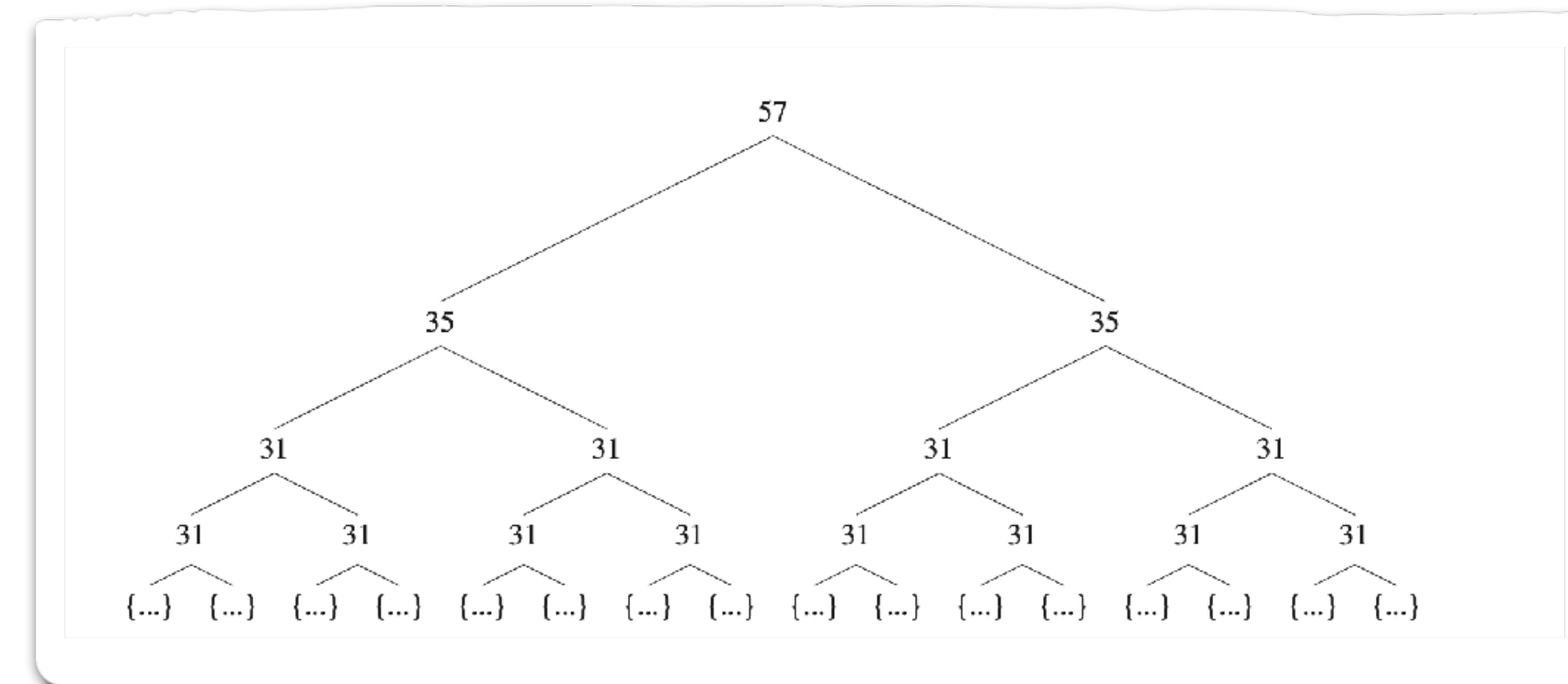
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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	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	0	1	...	0
3																			

$$2^n$$



2

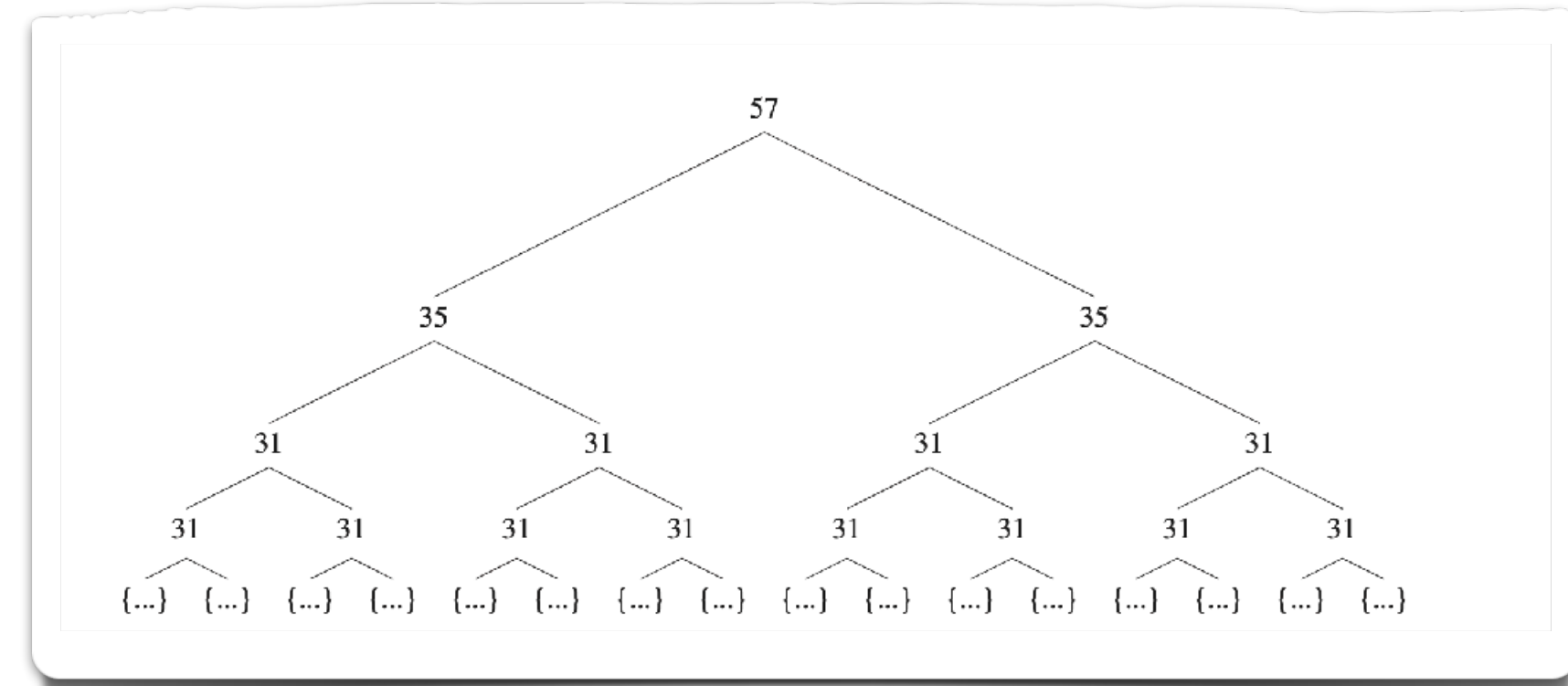
- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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	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	0	1	...	0
3																			

$$2^n$$



4

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

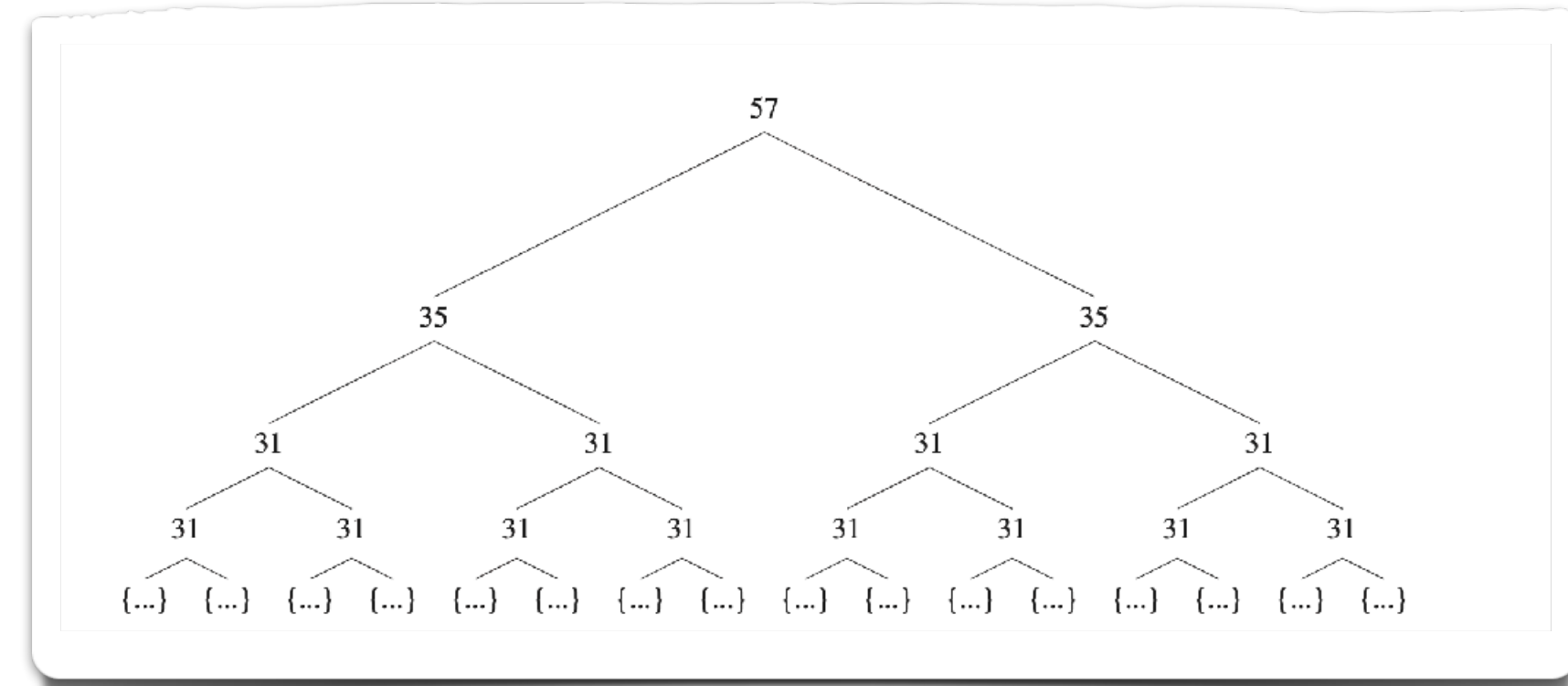
Effizienz?!

$$O(nZ)$$

$i \backslash x$	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	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	1	0	0	0	1	...	0

→ (at i=3, x=0)
 ↓ (at i=2, x=0)
 ↓ (at i=2, x=7)
 ↓ (at i=2, x=13)
 ↓ (at i=2, x=20)
 → (at i=2, x=17)
 → (at i=2, x=240)

$$2^n$$



4

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

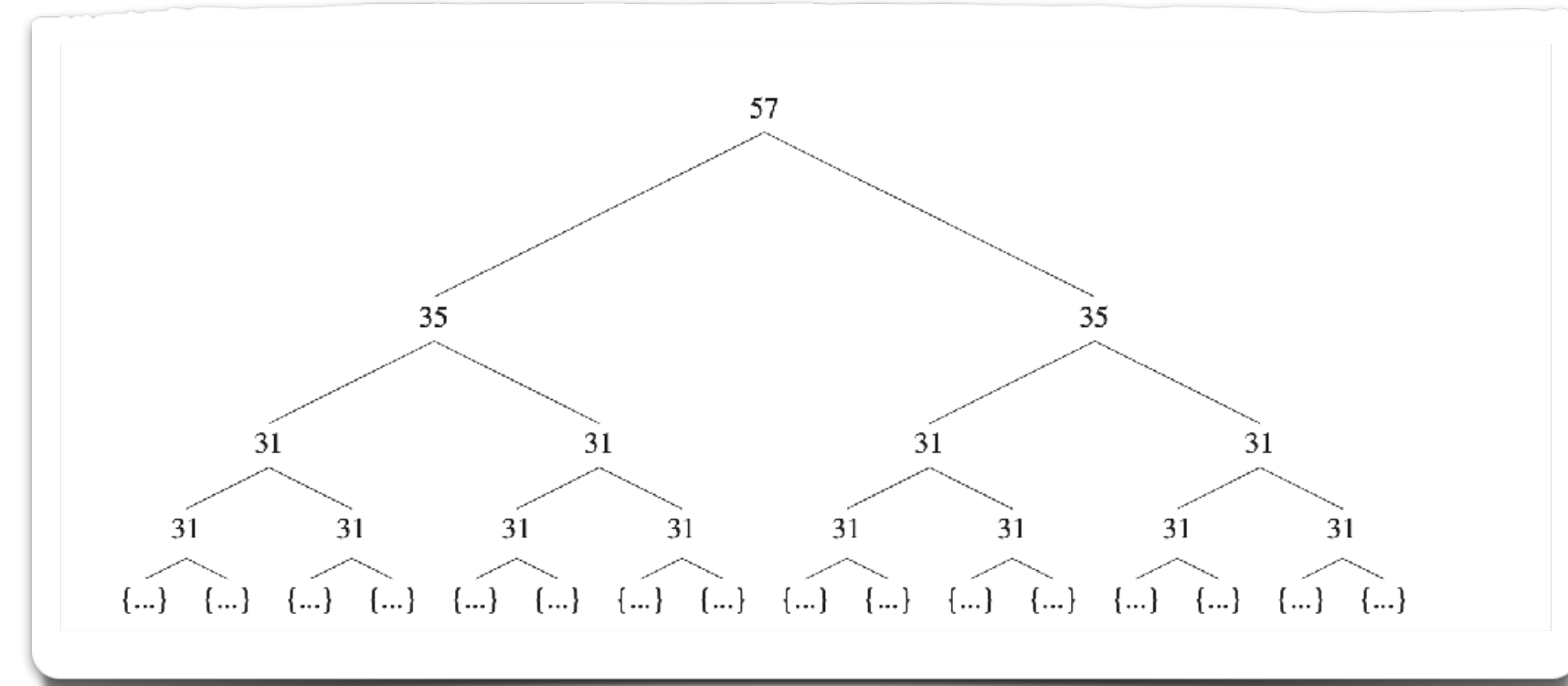
$$O(nZ)$$

$i \backslash x$	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	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	1	0	0	0	1	...	0

→ (at i=3, x=0)
 ↓ (at i=2, x=0)
 ↓ (at i=2, x=7)
 ↓ (at i=2, x=13)
 ↓ (at i=2, x=20)
 → (at i=2, x=17)
 → (at i=2, x=240)

+17

$$2^n$$



8

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

Effizienz?!

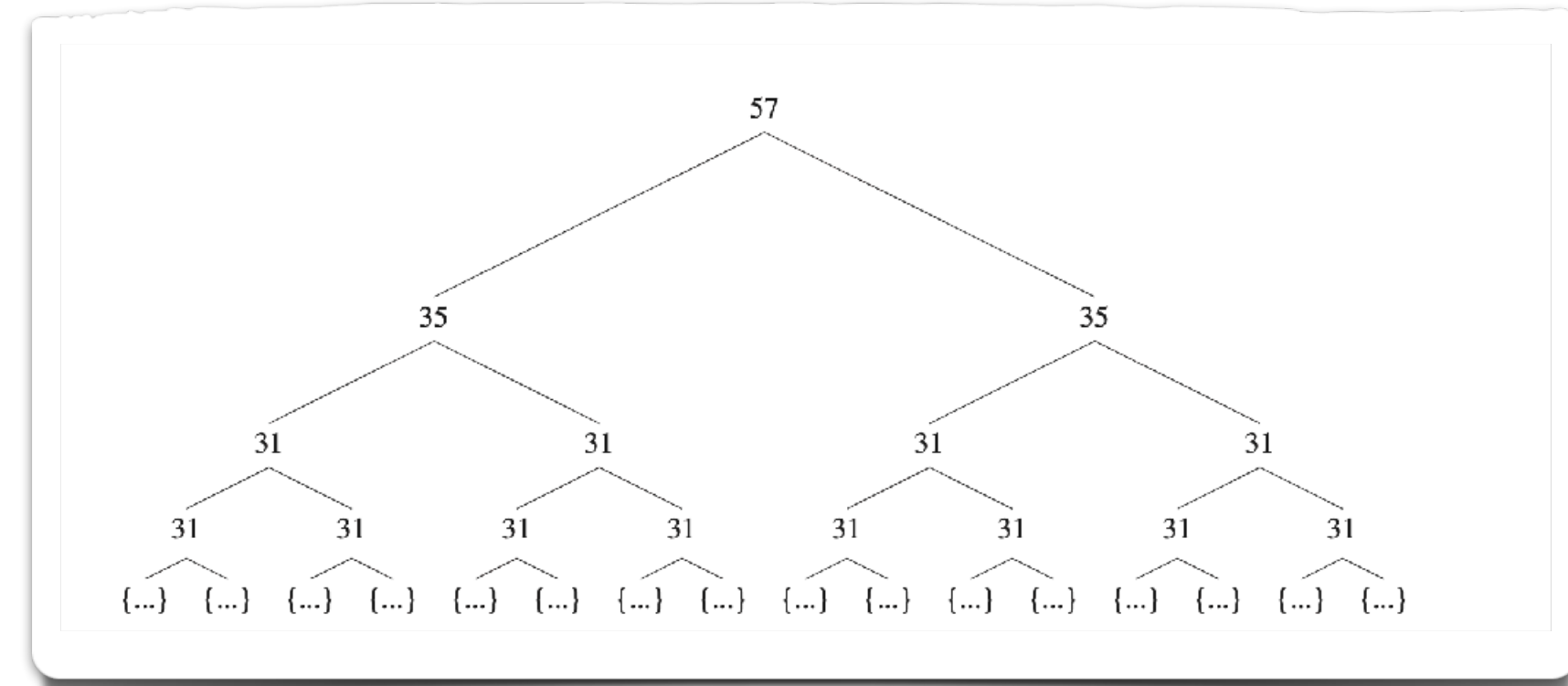
$$O(nZ)$$

$i \backslash x$	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	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	1	0	0	0	1	...	0

→ (at i=3, x=0)
 ↓ (at i=2, x=0)
 ↓ (at i=2, x=7)
 ↓ (at i=2, x=13)
 ↓ (at i=2, x=20)
 → (at i=2, x=17)
 → (at i=2, x=24)
 → (at i=2, x=30)
 → (at i=2, x=37)
 → (at i=2, x=240)

+17

$$2^n$$



16

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

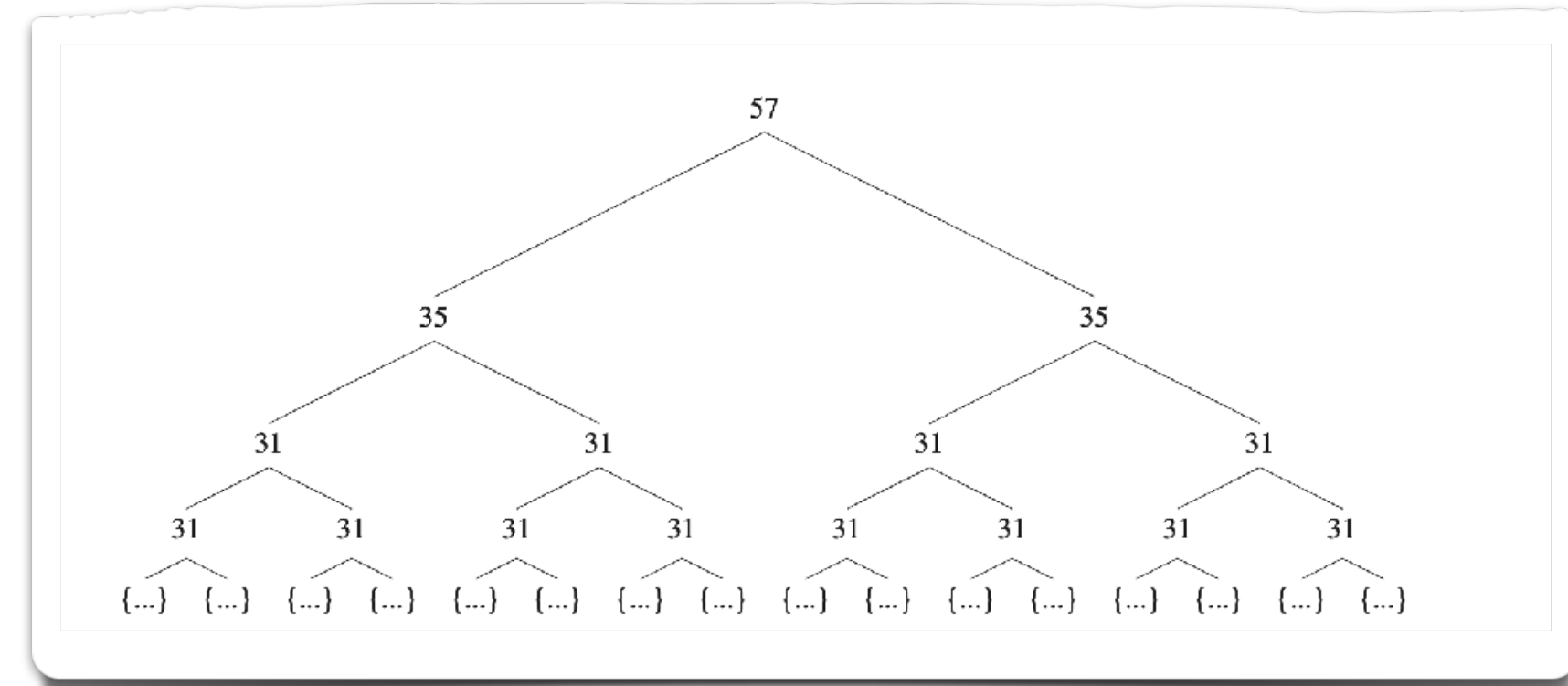
Effizienz?!

$$O(nZ)$$

$i \backslash x$	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	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	1	0	0	0	1	...	0

→ (at row 3, column 0)
 ↓ (at row 2, column 0)
 ↓ (at row 2, column 7)
 ↓ (at row 2, column 10)
 ↓ (at row 2, column 17)
 → (from row 2, column 0 to row 2, column 17)
 → (from row 2, column 17 to row 2, column 20)
 → (from row 2, column 20 to row 2, column 240)

$$2^n$$



32

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

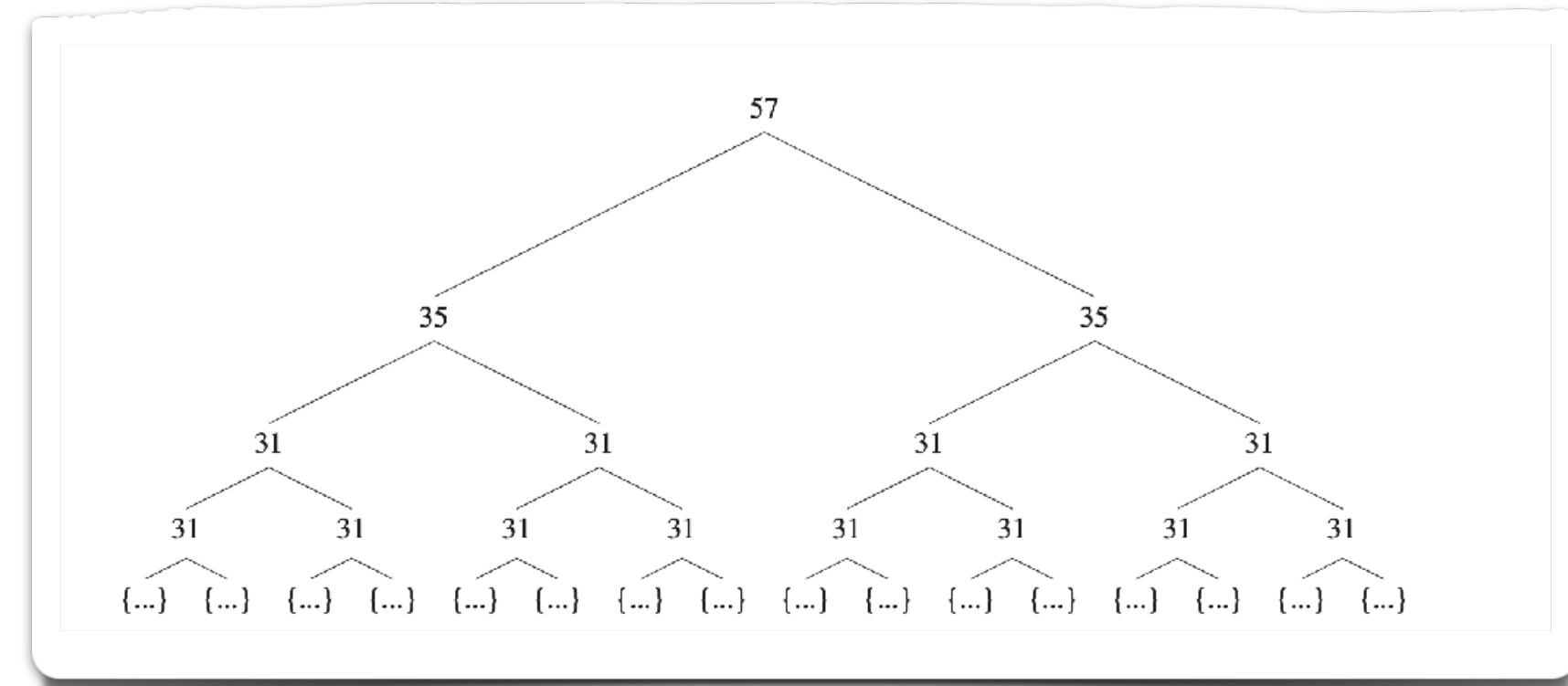
Effizienz?!

$$O(nZ)$$

$i \backslash x$	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	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	1	0	0	0	1	...	0

Note: A blue arrow points to row 3. Red arrows point to the '1' in column 7 of rows 2 and 3. Green arrows point from the '1' in row 2, column 7 to the '1' in row 3, column 17. A green '+17' is written below the arrow.

$$2^n$$



64

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

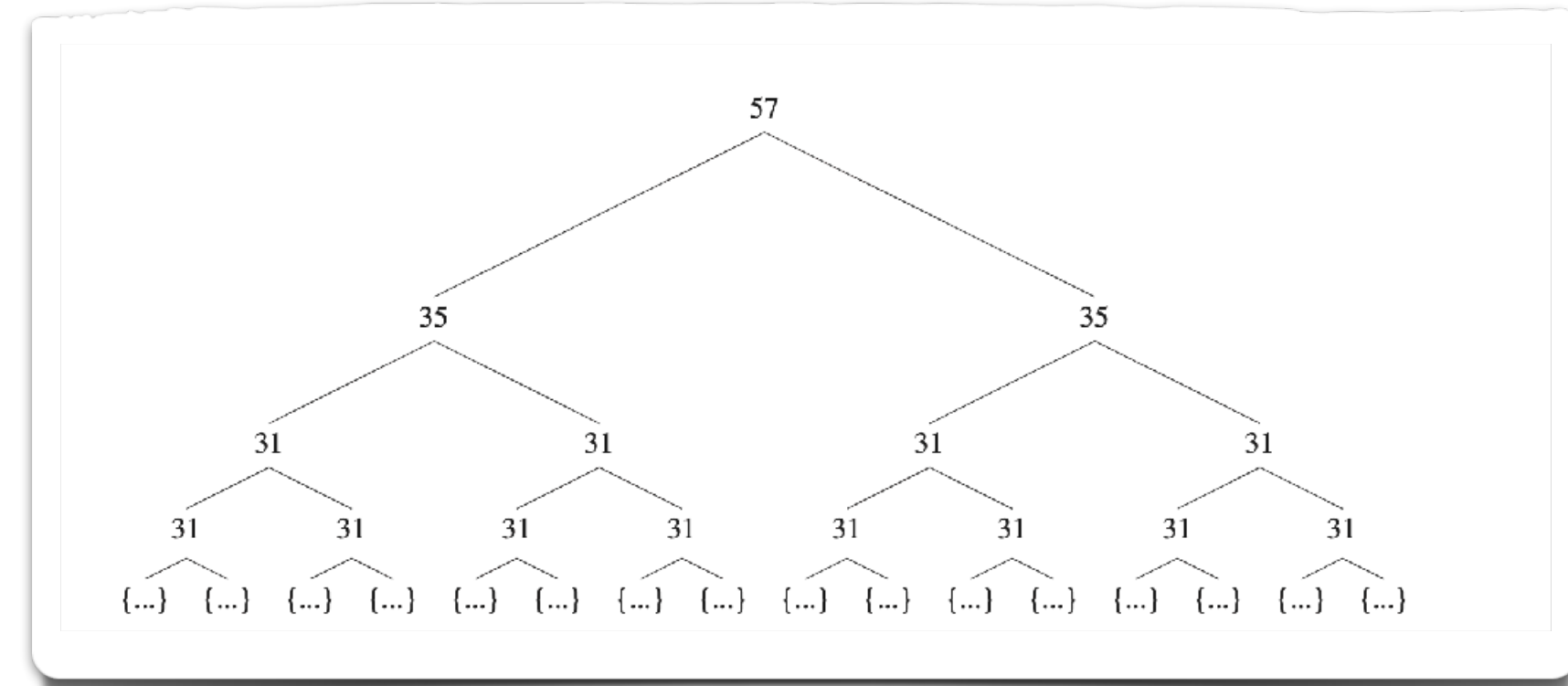
Effizienz?!

$$O(nZ)$$

$i \backslash x$	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	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	1	0	0	0	1	...	0

Note: A blue arrow points to row 3. Red arrows point to the '1' in column 7 of rows 2 and 3. Green arrows point from the '1' in row 2, column 7 to the '1' in row 3, column 17. A green '+17' is written below the '1' in row 3, column 17.

$$2^n$$



128

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

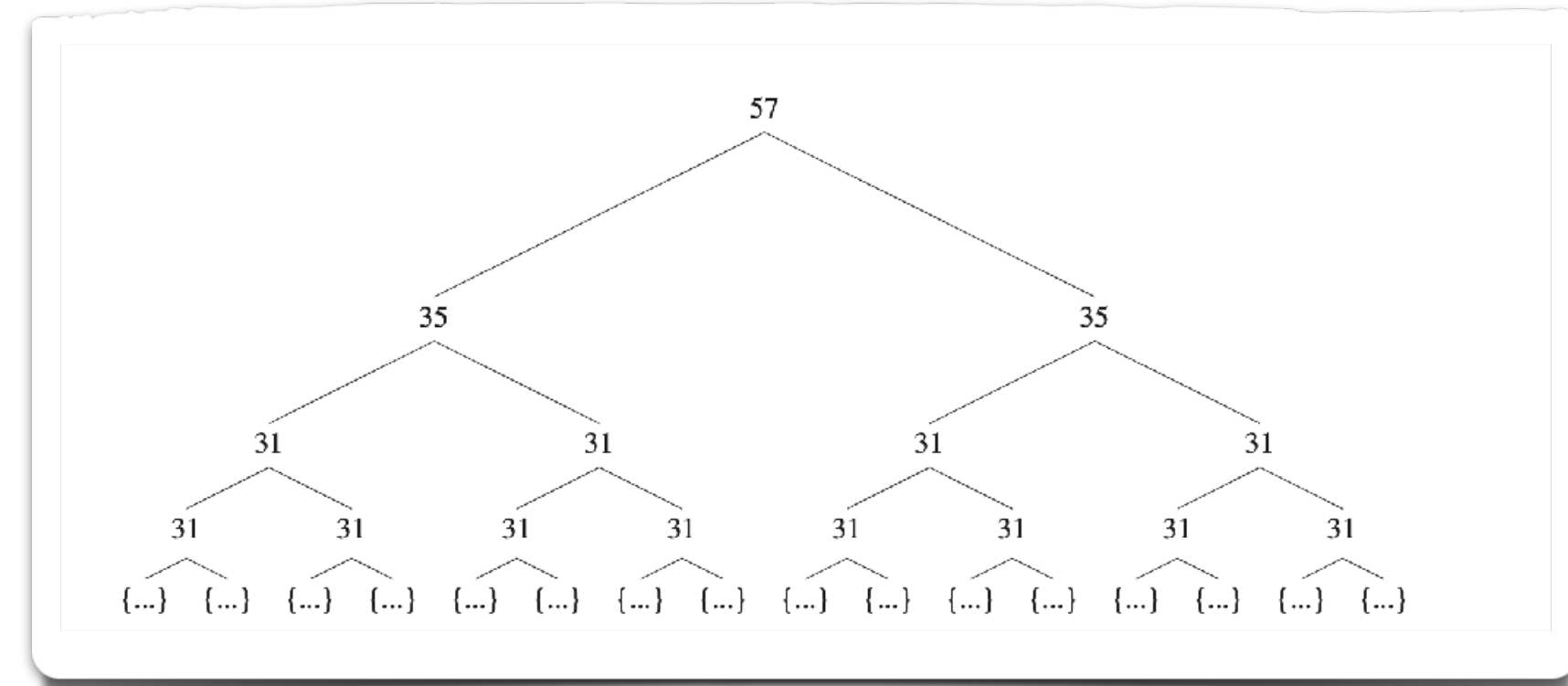
Effizienz?!

$$O(nZ)$$

$i \backslash x$	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	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	1	0	0	0	1	...	0

Note: A blue arrow points to row 3. Red arrows point to the first '1' in each row. Green arrows point from the '1' in row 2 to the '1' in row 3 at column 17, and from the '1' in row 3 to the '1' in row 2 at column 20. A green '+17' is written below the '1' in row 3 at column 17.

$$2^n$$



256?!

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

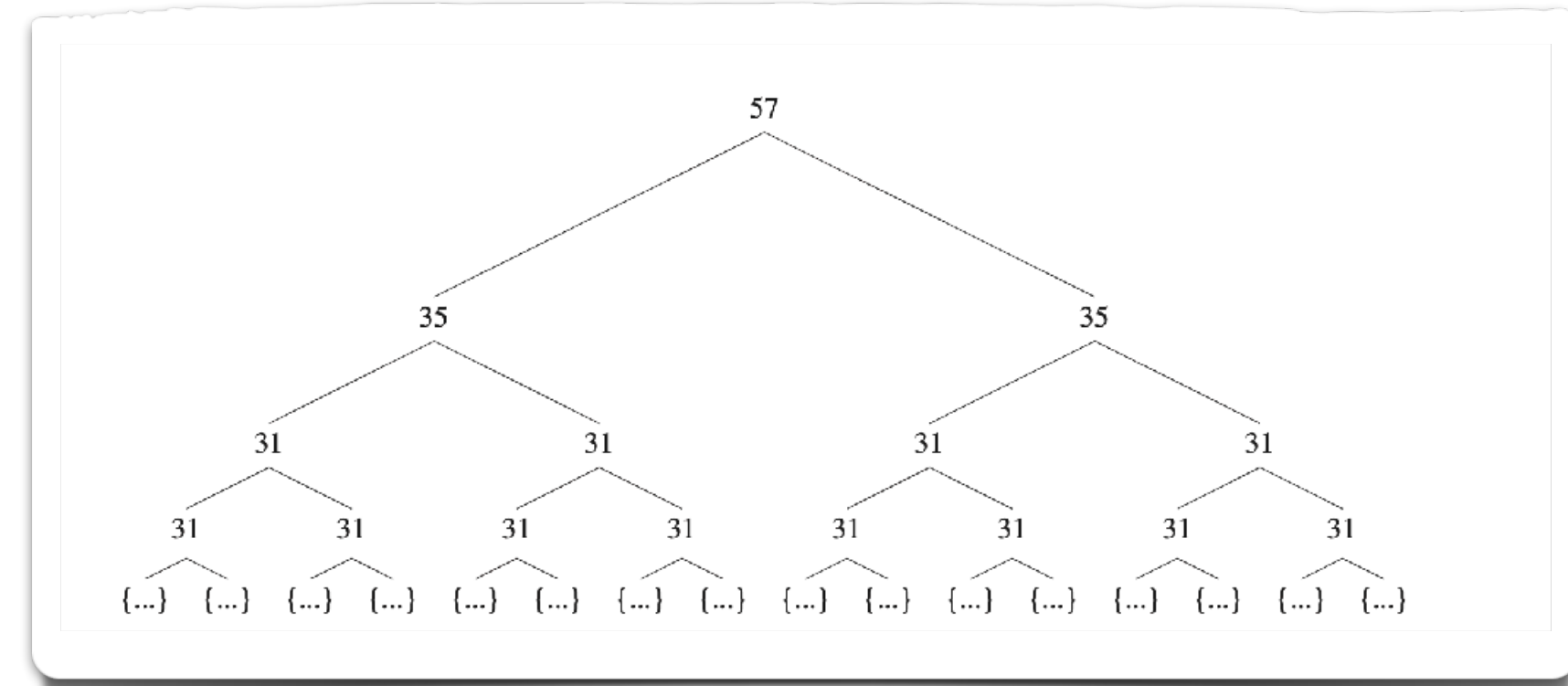
Effizienz?!

$$O(nZ)$$

$$2^n$$

$i \backslash x$	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	1	0	0	0	1	...	0				

Note: A blue arrow points to row 3. Red arrows point to the first '1' in each row. A green arrow points from the first '1' in row 2 to the '1' in row 3 at column 13. A green '+17' is written below the '1' in row 3 at column 13.



256?!

- Exponentiell viele Fälle!
- Wo sparen wir Arbeit?

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

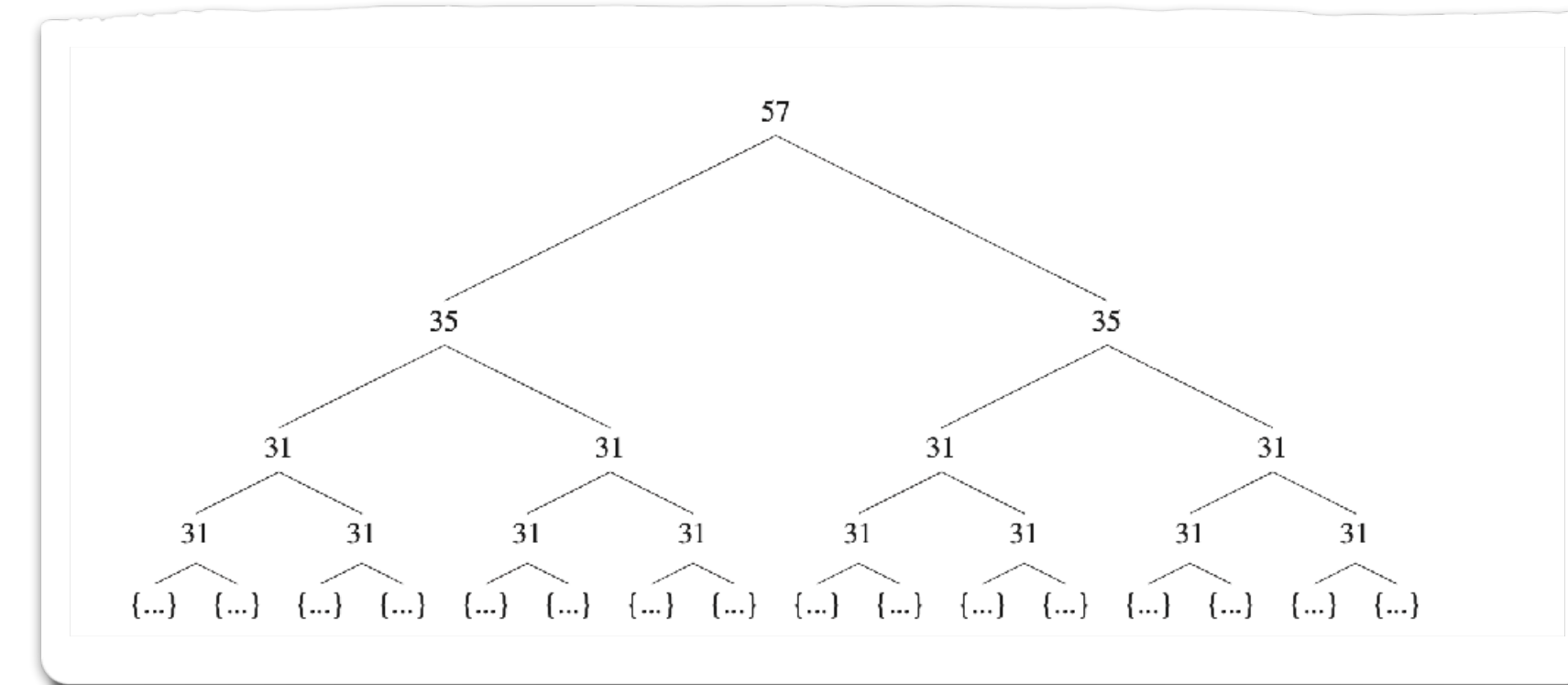
Effizienz?!

$$O(nZ)$$

$i \backslash x$	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	1	0	0	0	1	...	0			

Note: A blue arrow points to row 3. Red arrows point to the first '1' in each row. A green arrow points from the first '1' in row 2 to the '1' in row 3 at column 13. A green '+17' is written below the '1' in row 3 at column 13.

$$2^n$$



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

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