Karlsruher Institut für Technologie

2. Übung – Algorithmen I

Amortisierte Analyse

Beispiel Binärzähler

$$000 \quad \leftrightarrow \quad 0$$

Amortisierte Analyse

Beispiel Binärzähler

$$000 \leftrightarrow 0$$
$$001 \leftrightarrow 1$$

Amortisierte Analyse

Beispiel Binärzähler

$$
\begin{aligned}
000 &\leftrightarrow 0 \\
001 &\leftrightarrow 1 \\
010 &\leftrightarrow 2
\end{aligned}
$$

Amortisierte Analyse

Beispiel Binärzähler

$$
\begin{aligned}
000 &\leftrightarrow 0 \\
001 &\leftrightarrow 1 \\
010 &\leftrightarrow 2 \\
011 &\leftrightarrow 3
\end{aligned}
$$

Amortisierte Analyse

Beispiel Binärzähler

$$
\begin{aligned}
000 &\leftrightarrow 0 \\
001 &\leftrightarrow 1 \\
010 &\leftrightarrow 2 \\
011 &\leftrightarrow 3 \\
100 &\leftrightarrow 4
\end{aligned}
$$

Amortisierte Analyse

Beispiel Binärzähler

$$
\begin{aligned}
000 &\leftrightarrow 0 \\
001 &\leftrightarrow 1 \\
010 &\leftrightarrow 2 \\
011 &\leftrightarrow 3 \\
100 &\leftrightarrow 4 \\
101 &\leftrightarrow 5
\end{aligned}
$$

Amortisierte Analyse

Beispiel Binärzähler

$$
\begin{aligned}
000 &\leftrightarrow 0 \\
001 &\leftrightarrow 1 \\
010 &\leftrightarrow 2 \\
011 &\leftrightarrow 3 \\
100 &\leftrightarrow 4 \\
101 &\leftrightarrow 5 \\
\ldots\beta_2\beta_1\beta_0 &\leftrightarrow \sum_i \beta_i 2^i
\end{aligned}
$$

Amortisierte Analyse

Beispiel Binärzähler

- Binärzähler: zähle von 0 bis m
- m Inkrementoperationen
- $x = \sum_i \beta_i 2^i$

Function binary-increment() :

$\beta_0 \leftarrow \beta_0 + 1$

i=0 : $\mathbb{N}_{\geq 0}$

while $\beta_i = 2$ **do**

$\beta_{i+1} \leftarrow \beta_{i+1} + 1$

$\beta_i \leftarrow 0$

$i \leftarrow i + 1$

$$
\begin{aligned}
000 &\leftrightarrow 0 \\
001 &\leftrightarrow 1 \\
010 &\leftrightarrow 2 \\
011 &\leftrightarrow 3 \\
100 &\leftrightarrow 4 \\
101 &\leftrightarrow 5 \\
\ldots\beta_2\beta_1\beta_0 &\leftrightarrow \sum_i \beta_i 2^i
\end{aligned}
$$

Beispiel Binärzähler

binary-increment

0011111111
0011111111

Beispiel Binärzähler

binary-increment

0011111111
0011111111

Beispiel Binärzähler

binary-increment

$$001111111$$
$$001111112$$

Beispiel Binärzähler

binary-increment

001111111
001111120

Beispiel Binärzähler

binary-increment

001111111
001111200

Beispiel Binärzähler

binary-increment

001111111
001112000

Beispiel Binärzähler

binary-increment

001111111
001120000

Beispiel Binärzähler

binary-increment

0011111111
0012000000

Beispiel Binärzähler

binary-increment

001111111
002000000

Beispiel Binärzähler

binary-increment

001111111
010000000

Beispiel Binärzähler

binary-increment

001111111
010000000

- Kostenmodell: Kosten 1 pro Bitflip

Beispiel Binärzähler

binary-increment

001111111
010000000

- Kostenmodell: Kosten 1 pro Bitflip
- $\Longrightarrow$ increment worst case $O(\log m)$

Amortisierte Analyse

Beispiel Binärzähler

- Kostenmodell: Kosten 1 pro Bitflip
- $T(m)$ seien Gesamtkosten für m Operationen
- amortisierten Kosten pro Operation definiert als $T(m)/m$

amortisierten Kosten einer Inkrementoperation?

a) $O(1)$ b) $O(\log m)$

Beispiel Binärzähler

Aggregatmethode

- Kochrezept: Schätze $T(m)$ direkt ab

Operation	Kosten
$0000 \to 0001$	1
$0001 \to 0010$	2
$0010 \to 0011$	1
$0011 \to 0100$	3
$0100 \to 0101$	1
$0101 \to 0110$	2
$0110 \to 0111$	1
$0111 \to 1000$	4
....	

- Beobachtung (m Ops):
  - β_0 jedes mal gekippt $\quad \sum = m$
  - β_1 jedes zweite mal gekippt $\quad \sum = \frac{m}{2}$
  - β_2 jedes vierte mal gekippt $\quad \sum = \frac{m}{4}$
  - β_3 jedes achte mal gekippt $\quad \sum = \frac{m}{8}$
  - ....

$$T(m) \leq m + \frac{m}{2} + \frac{m}{4} + \frac{m}{8} \ldots \leq \sum_{i=0}^{\infty} \frac{m}{2^i} = 2m$$

Beispiel Binärzähler

Aggregatmethode

- Kochrezept: Schätze $T(m)$ direkt ab

Operation	Kosten
$0000 \to 0001$	1
$0001 \to 0010$	2
$0010 \to 0011$	1
$0011 \to 0100$	3
$0100 \to 0101$	1
$0101 \to 0110$	2
$0110 \to 0111$	1
$0111 \to 1000$	4
....	

- Beobachtung (m Ops):
  - β_0 jedes mal gekippt $\quad \sum = m$
  - β_1 jedes zweite mal gekippt $\quad \sum = \frac{m}{2}$
  - β_2 jedes vierte mal gekippt $\quad \sum = \frac{m}{4}$
  - β_3 jedes achte mal gekippt $\quad \sum = \frac{m}{8}$
  - ....

$$T(m) \leq m + \frac{m}{2} + \frac{m}{4} + \frac{m}{8} \ldots \leq \sum_{i=0}^{\infty} \frac{m}{2^i} = 2m$$

$$\Rightarrow \frac{T(m)}{m} \leq 2 = O(1)$$

Beispiel Binärzähler

Bankkontomethode

- Allgemein: $t_1, ..., t_m$ die Laufzeiten der Einzeloperationen
- t_i Gebühr für i-te Operation
- vor jeder Operation erhält Algorithmus Gehalt G
- alle Operationen müssen aus Gehältern bezahlt werden
- finde Gehalt G, sodass $T(m) \leq mG$

Beispiel Binärzähler

Bankkontomethode

- finde Gehalt G, sodass mit $T(m) \leq mG$
- Binärzähler: $G = 2$ Chips, Strategie:
- Zählvorgang setzt genau ein Bit auf 1. Dies kostet 1 Chip

Beispiel Binärzähler

Bankkontomethode

- finde Gehalt G, sodass mit $T(m) \leq mG$
- Binärzähler: $G = 2$ Chips, Strategie:

- Zählvorgang setzt genau ein Bit auf 1. Dies kostet 1 Chip
- Spare anderen Chip, um Bit auf 0 zu setzen

Beispiel Binärzähler

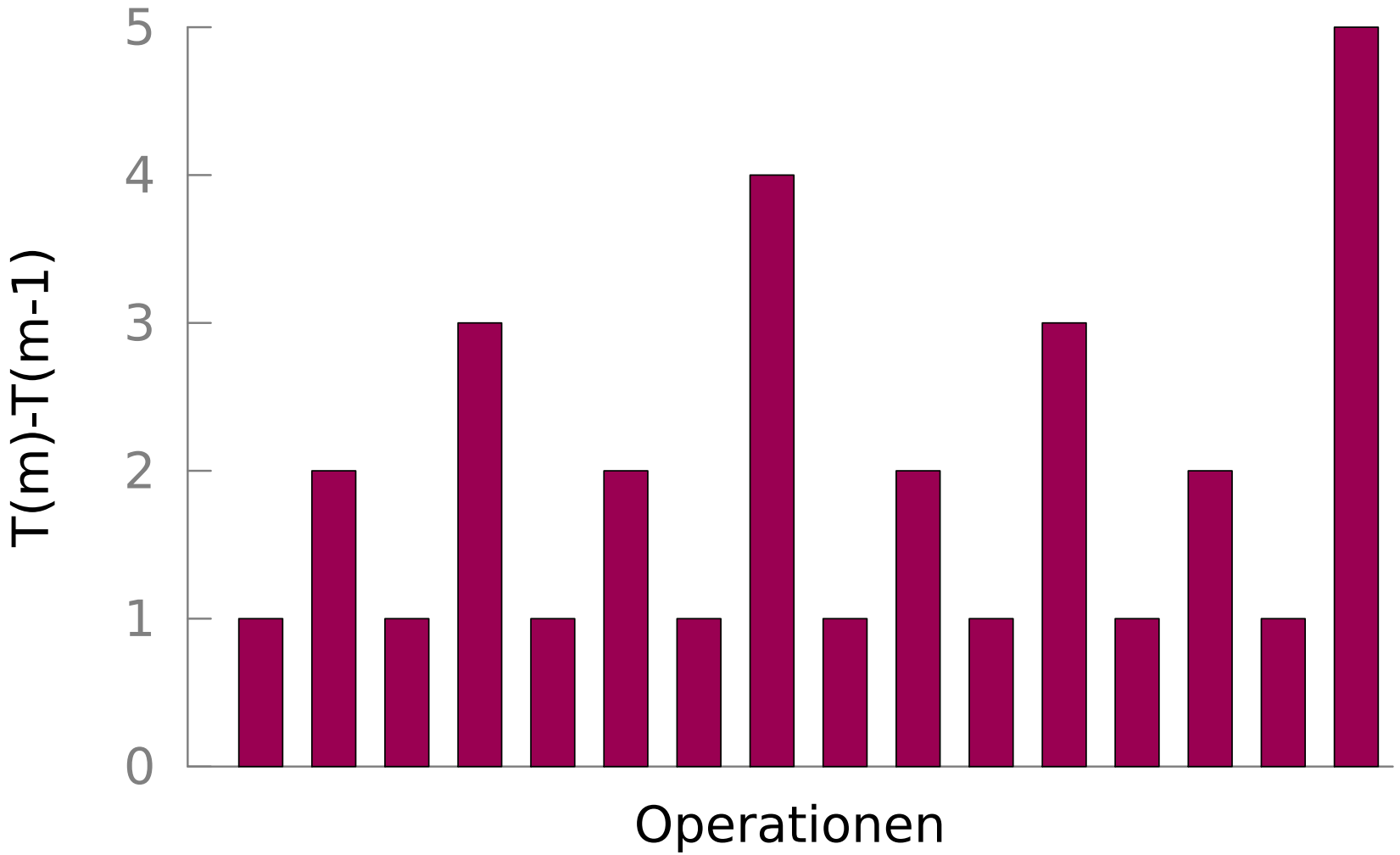
Bankkontomethode

- finde Gehalt G, sodass mit $T(m) \leq mG$
- Binärzähler: $G = 2$ Chips, Strategie:

- Zählvorgang setzt genau ein Bit auf 1. Dies kostet 1 Chip
- Spare anderen Chip, um Bit auf 0 zu setzen
- Invariante: Kontostand = Anzahl der 1-Bits

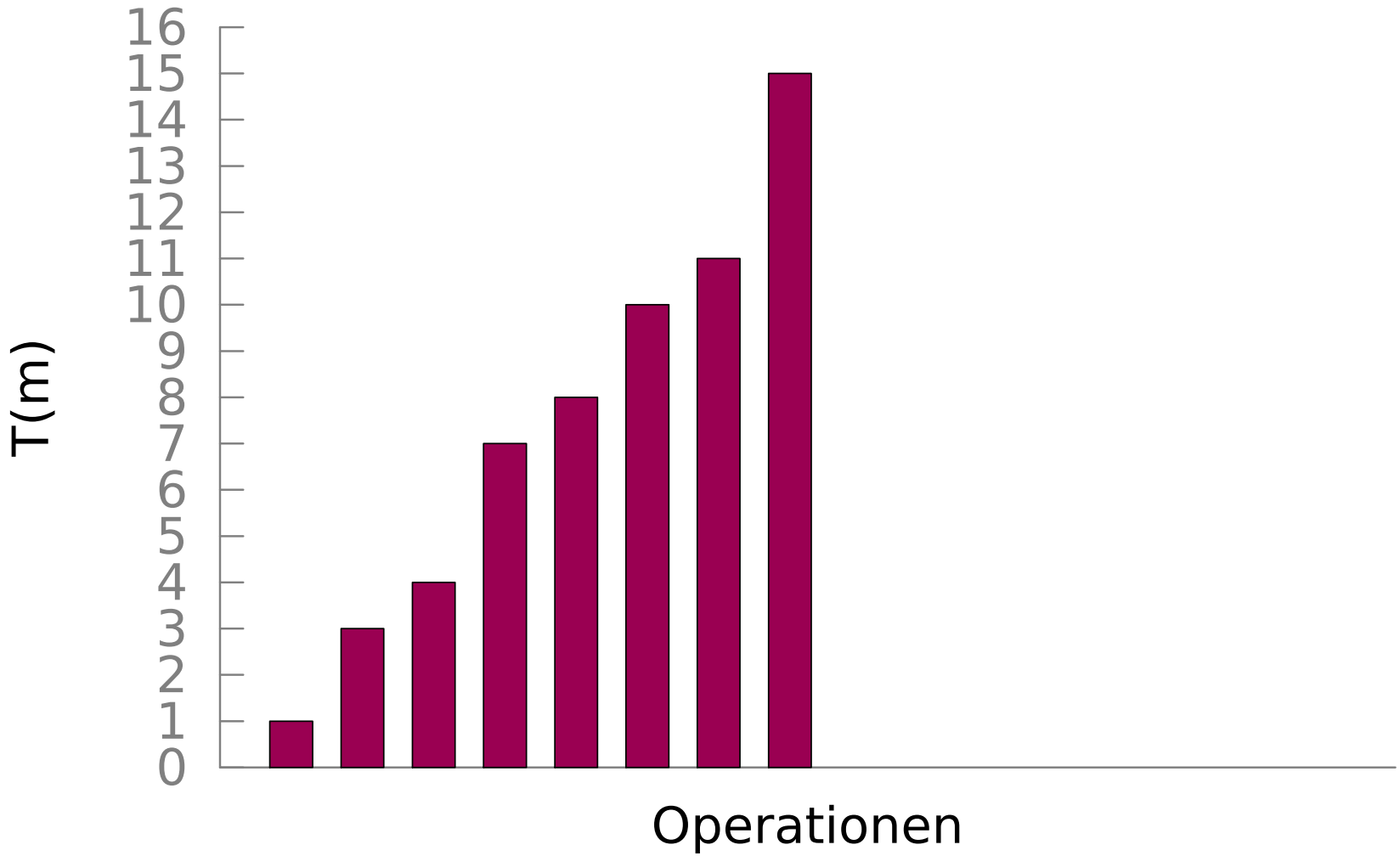
Beispiel Binärzähler

Einzelkosten



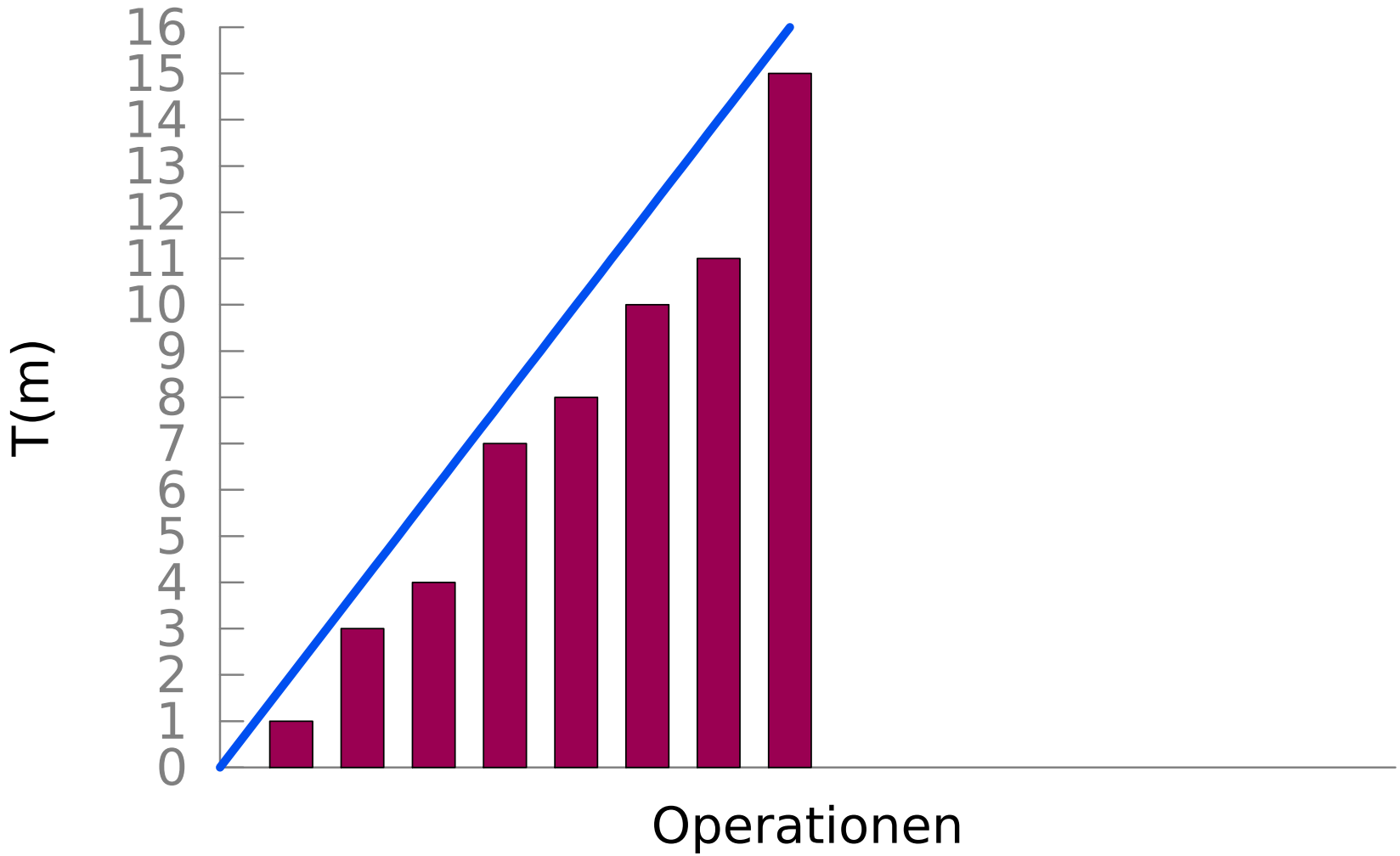
Beispiel Binärzähler

Summierte Kosten



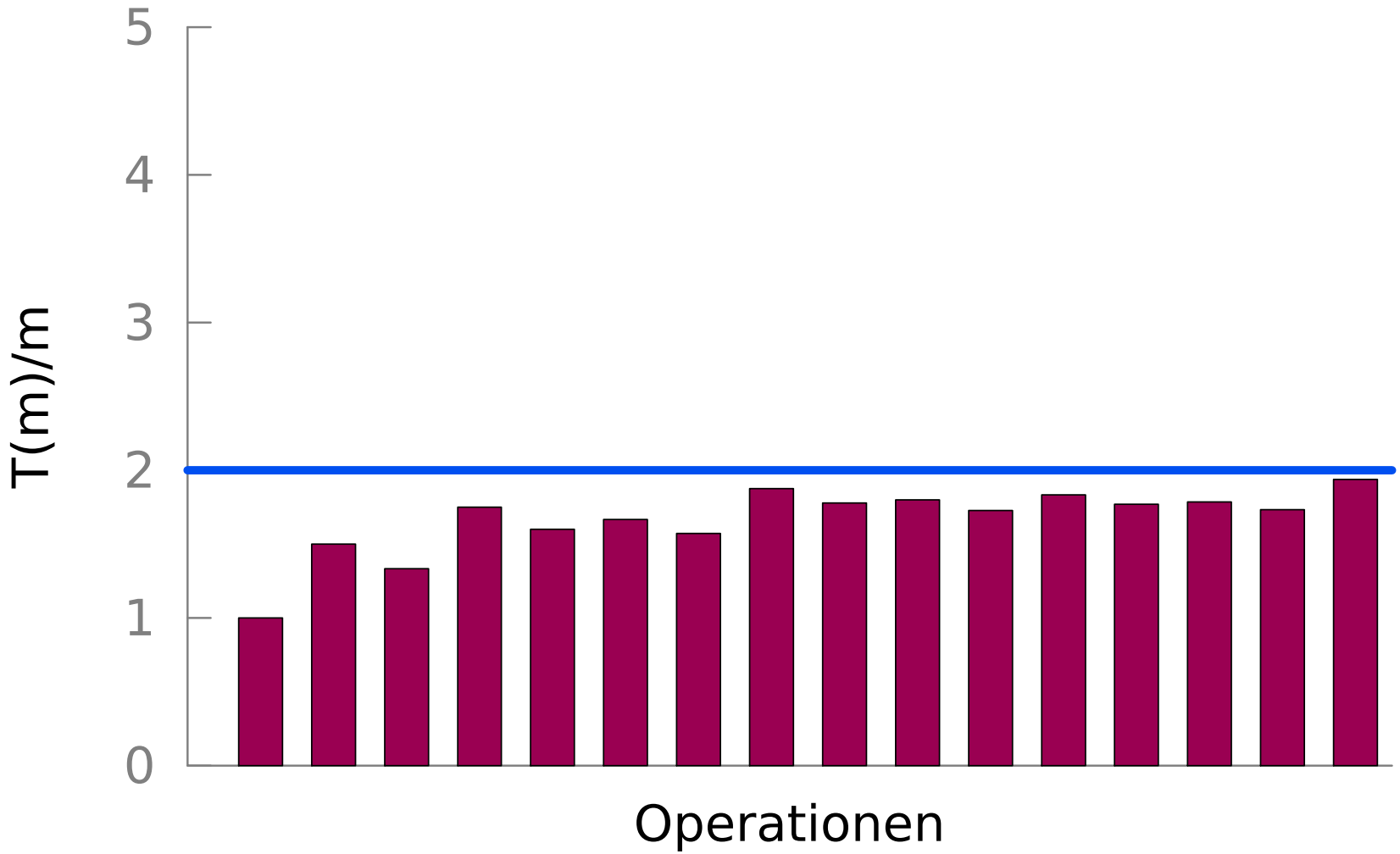
Beispiel Binärzähler

Summierte Kosten $\leq 2m$



Beispiel Binärzähler

$T(m)/m \leq 2$



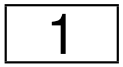
Unbounded Array

Wieviele Speicherzellen braucht
ein Unbounded Array gleichzeitig
für n Elemente?

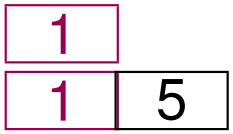
- Worst-Case-Betrachtung
- es darf eine beliebige Folge von `pushBack` und `popBack` Operationen vorausgegangen sein

a) $2n \pm c$
b) $3n \pm c$
c) $4n \pm c$
d) $6n \pm c$

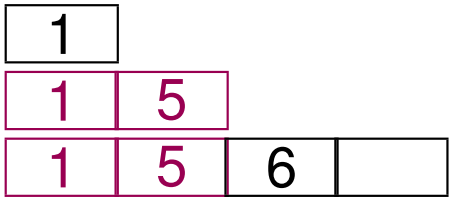
Unbounded Array live!



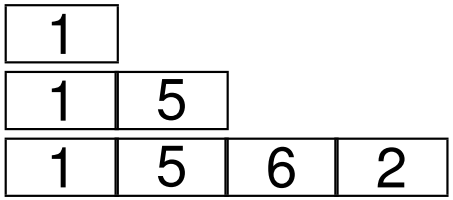
Unbounded Array live!



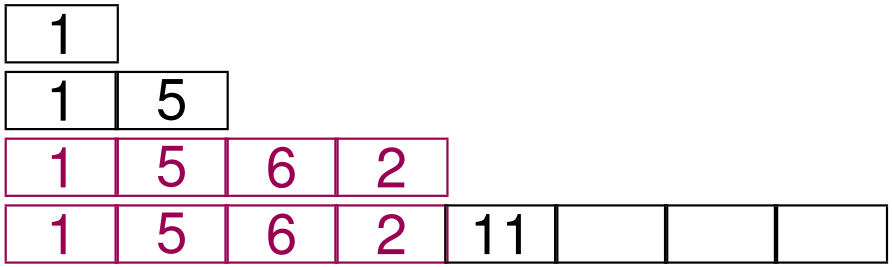
Unbounded Array live!



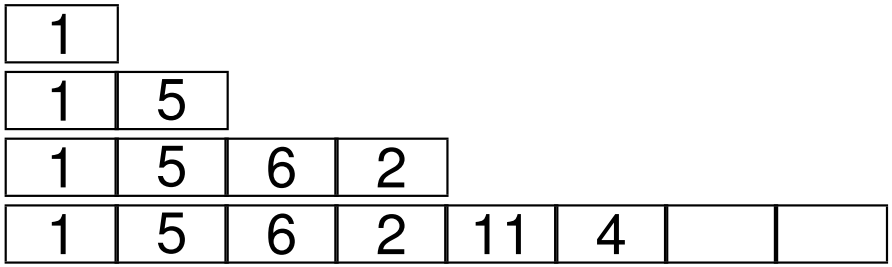
Unbounded Array live!



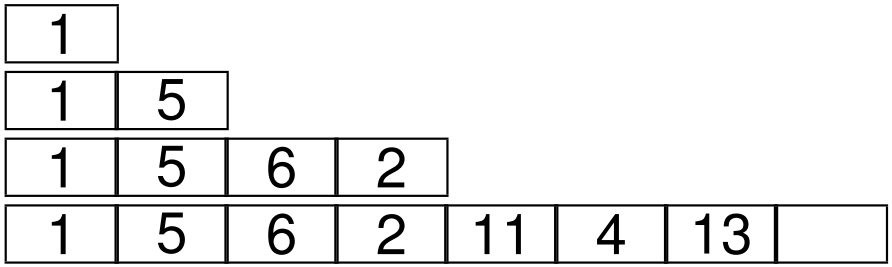
Unbounded Array live!



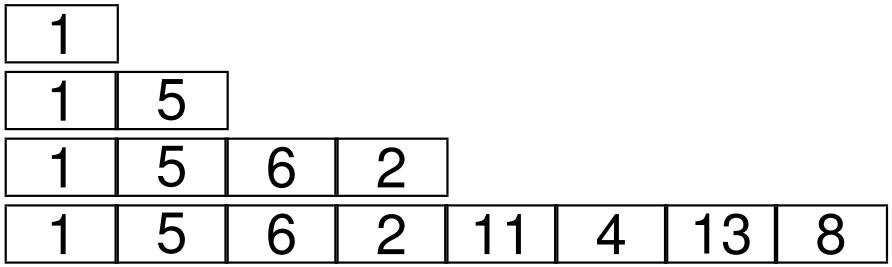
Unbounded Array live!



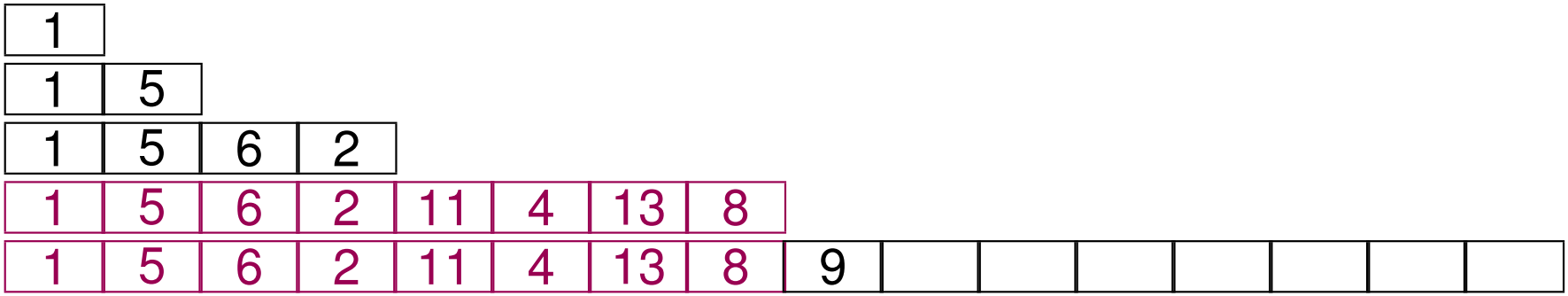
Unbounded Array live!



Unbounded Array live!

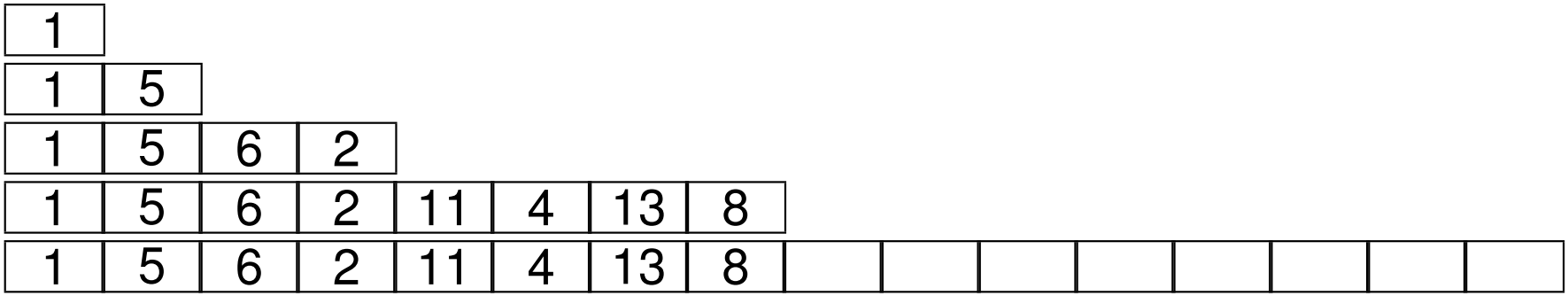


Unbounded Array live!



- Platzbedarf bei Vergrößerung: $3(n-1)$ Speicherzellen
- falls maximales n bekannt: bounded array oft besser
- $\Rightarrow$ Antwort b richtig?

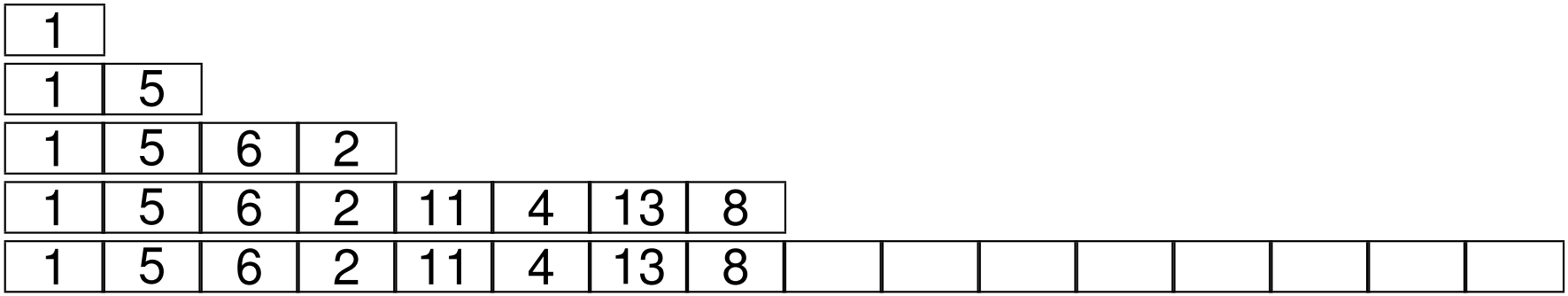
Unbounded Array live!



Warum schrumpft das Array nicht?

- die Zukunft nicht bekannt (Stichwort Online-Algorithmus)
- Problem: erneutes `pushBack` $\rightarrow$ erneutes $O(n)$ Anwachsen
- Problem: Folge von n solcher `popBack` / `pushBack` $\rightarrow$ $O(n^2)$ Laufzeit

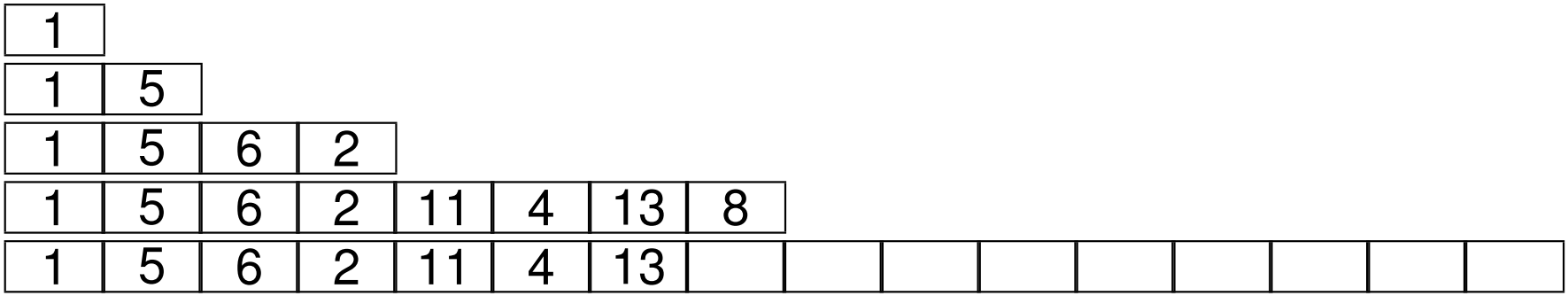
Unbounded Array live!



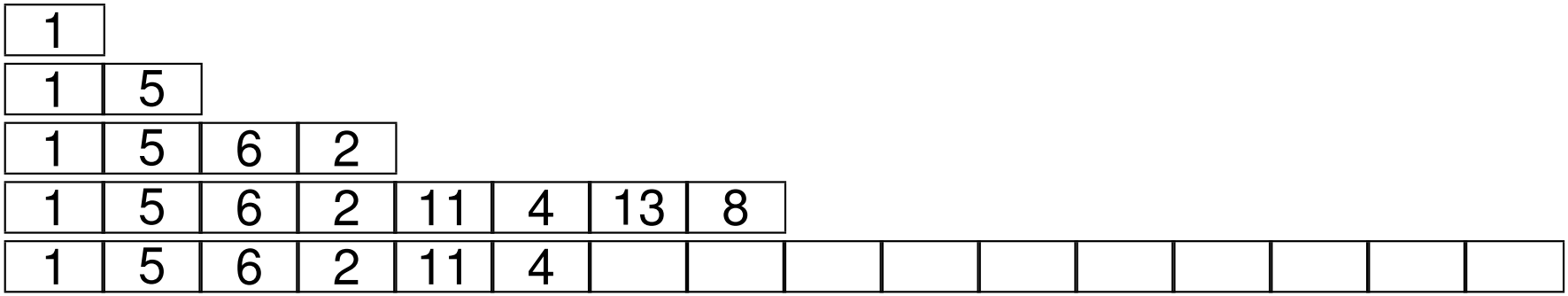
Warum schrumpft das Array nicht?

- die Zukunft nicht bekannt (Stichwort Online-Algorithmus)
- Problem: erneutes `pushBack` $\rightarrow$ erneutes $O(n)$ Anwachsen
- Problem: Folge von n solcher `popBack` / `pushBack` $\rightarrow$ $O(n^2)$ Laufzeit
- deshalb erst bei $1/4$ schrumpfen $\rightarrow$ $\Omega(n)$ konstante Operationen seit dem letzten Anwachsen/Schrumpfen

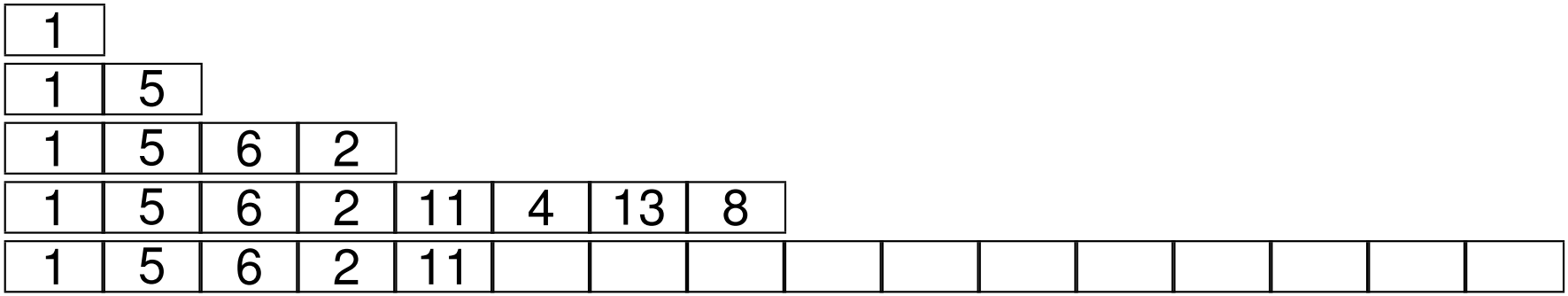
Unbounded Array live!



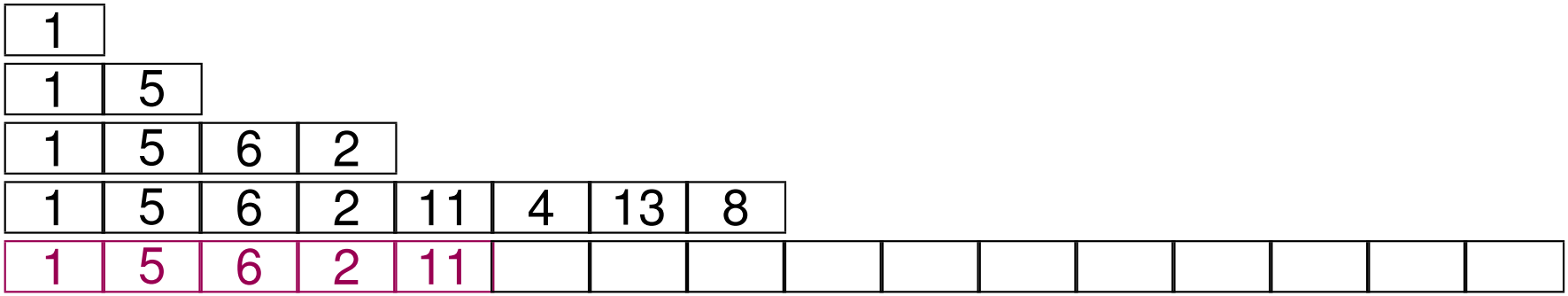
Unbounded Array live!



Unbounded Array live!

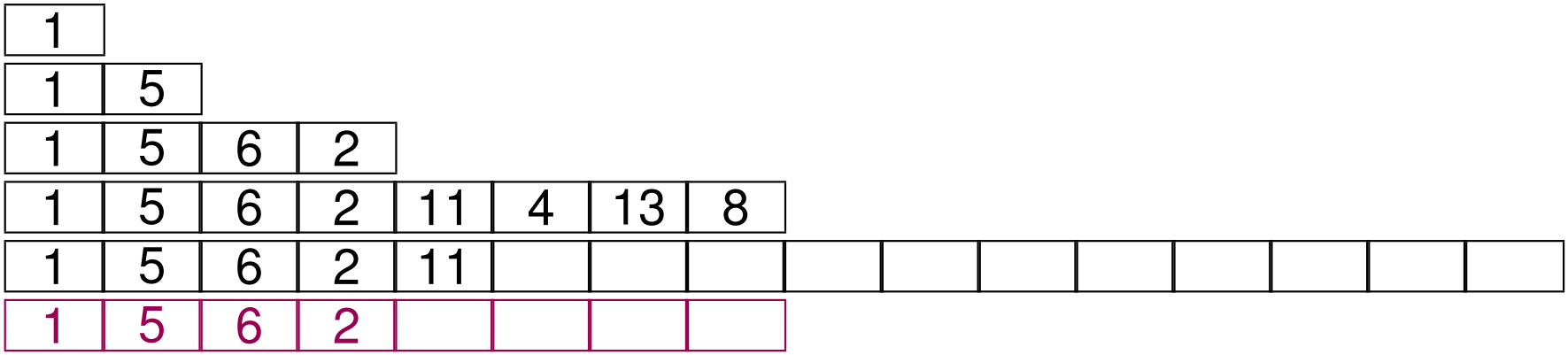


Unbounded Array live!



- Platzbedarf direkt vor Schrumpfoperation: $4(n-1)$ Speicherzellen
- $\Rightarrow$ Antwort c richtig?

Unbounded Array live!



- Wenn Feldkapazität $4n \rightarrow$ schrumpfe Feld auf $2n$
- es bleibt Platz für weitere Elemente
- während Schrumpfoperation: 6n Speicherzellen gleichzeitig belegt
- außer Speicherverwaltung bietet Schrumpfoperation ohne Kopieren
- $\Rightarrow$ **Antwort d richtig!**

Unbounded Array: Allgemeine Version

Kompromiss: im schlimmsten Fall weniger Platz, dafür langsamer?

Unbounded Array: Allgemeine Version

Kompromiss: im schlimmsten Fall weniger Platz, dafür langsamer?

Ja! Durch Wahl des:

- "'Vergrößerungsfaktors"' β (hier bisher immer 2)
- "'Schranke"' α (hier bisher immer 4)
  der verallgemeinerten Version:

Unbounded Array: Allgemeine Version

Kompromiss: im schlimmsten Fall weniger Platz, dafür langsamer?

Ja! Durch Wahl des:

- "'Vergrößerungsfaktors"' β (hier bisher immer 2)
- "'Schranke"' α (hier bisher immer 4)
  der verallgemeinerten Version:

  reallociere βn

  verkleinere bei $\alpha n \leq w \wedge n > 0$

Unbounded Array: Allgemeine Version

Kompromiss: im schlimmsten Fall weniger Platz, dafür langsamer?

Ja! Durch Wahl des:

- "'Vergrößerungsfaktors"' β (hier bisher immer 2)
- "'Schranke"' α (hier bisher immer 4)
  der verallgemeinerten Version:

  reallociere βn

  verkleinere bei $\alpha n \leq w \wedge n > 0$

- amortisierte Laufzeiten bleiben asymptotisch gleich für $\beta > 1, \alpha > \beta$.
- größeres β, α: schneller, benötigt aber mehr Speicherplatz
- kleineres β, α: langsamer, benötigt aber weniger Speicherplatz