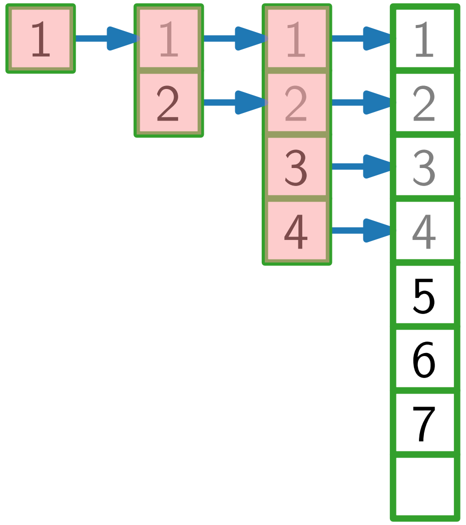
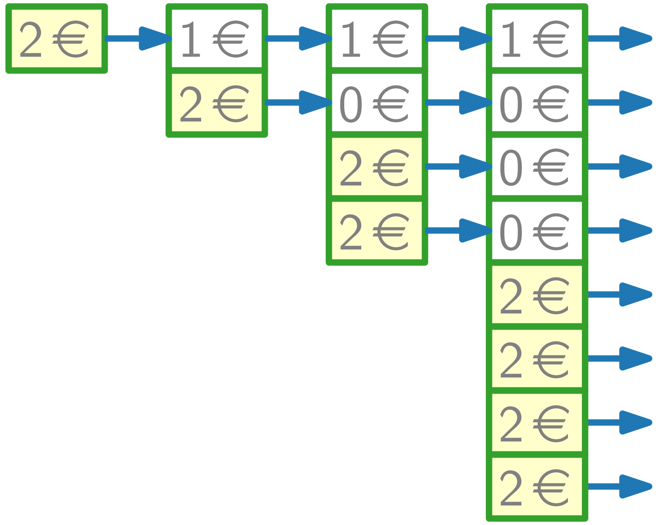
Algorithmen und Datenstrukturen



Vorlesung 16:
Amortisierte Analyse

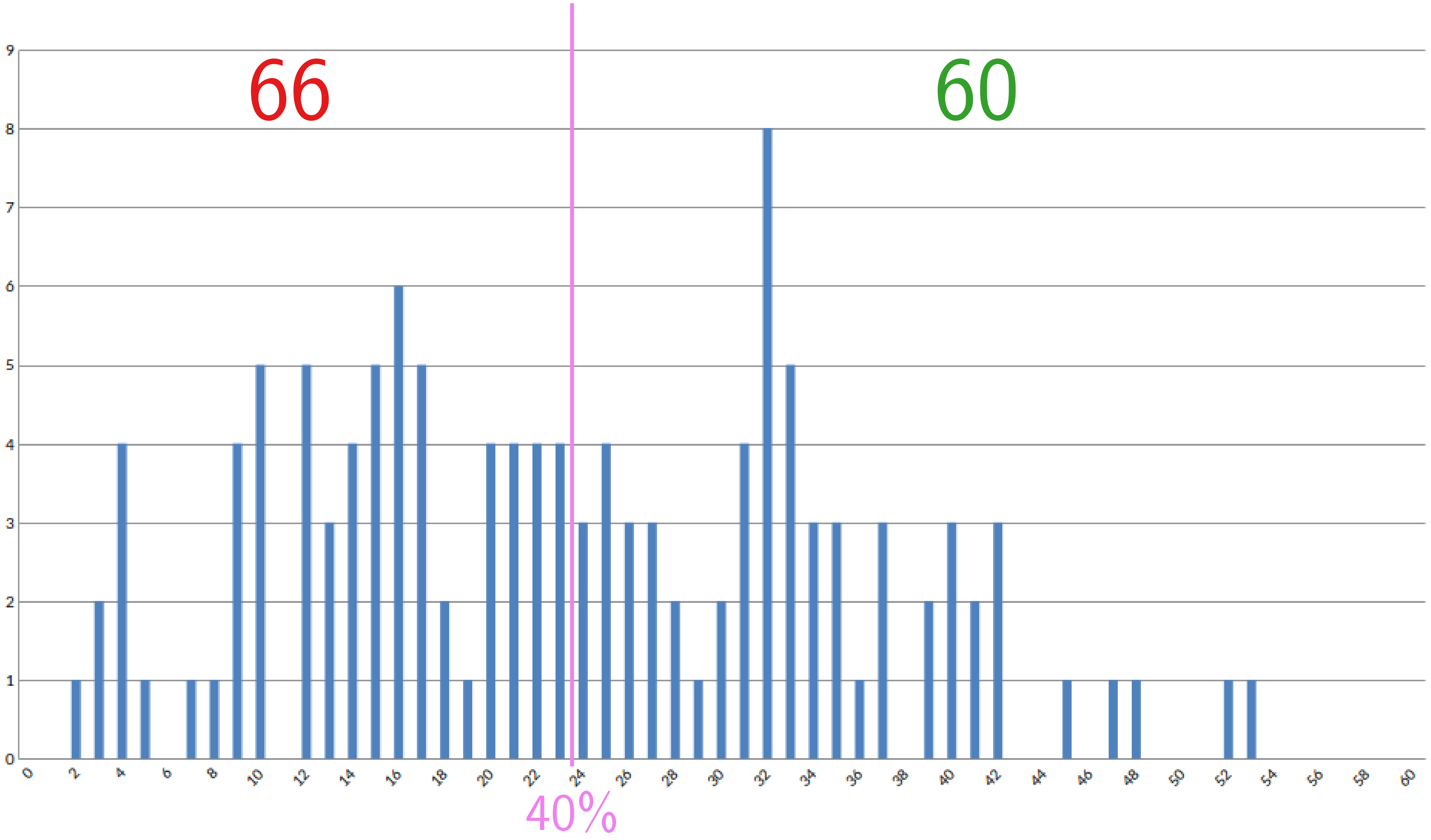


Alexander Wolff

Wintersemester 2025

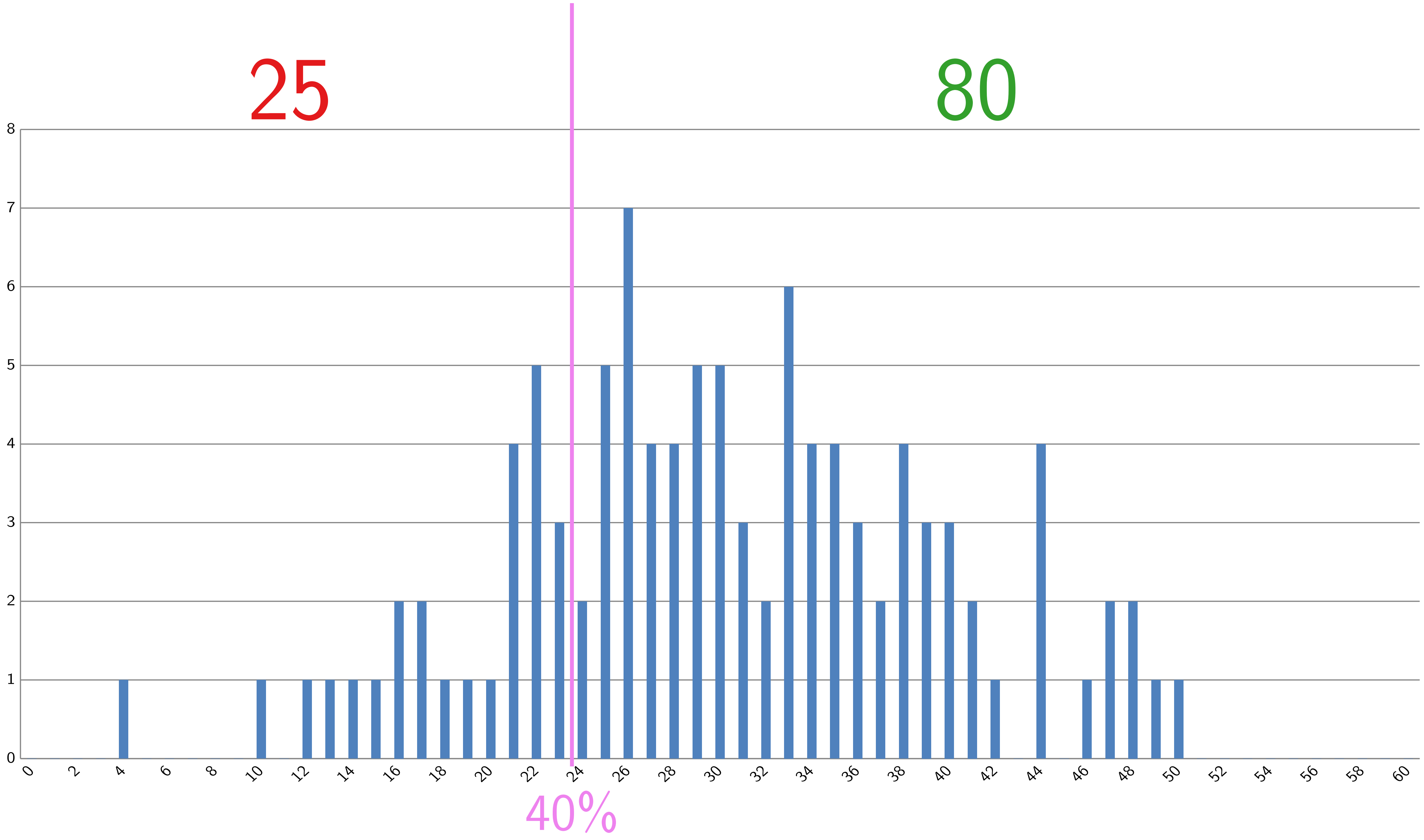
1. Zwischentest

$n = 126$; Durchschnitt $= 23{,}7$; Median $= 21{,}5$.



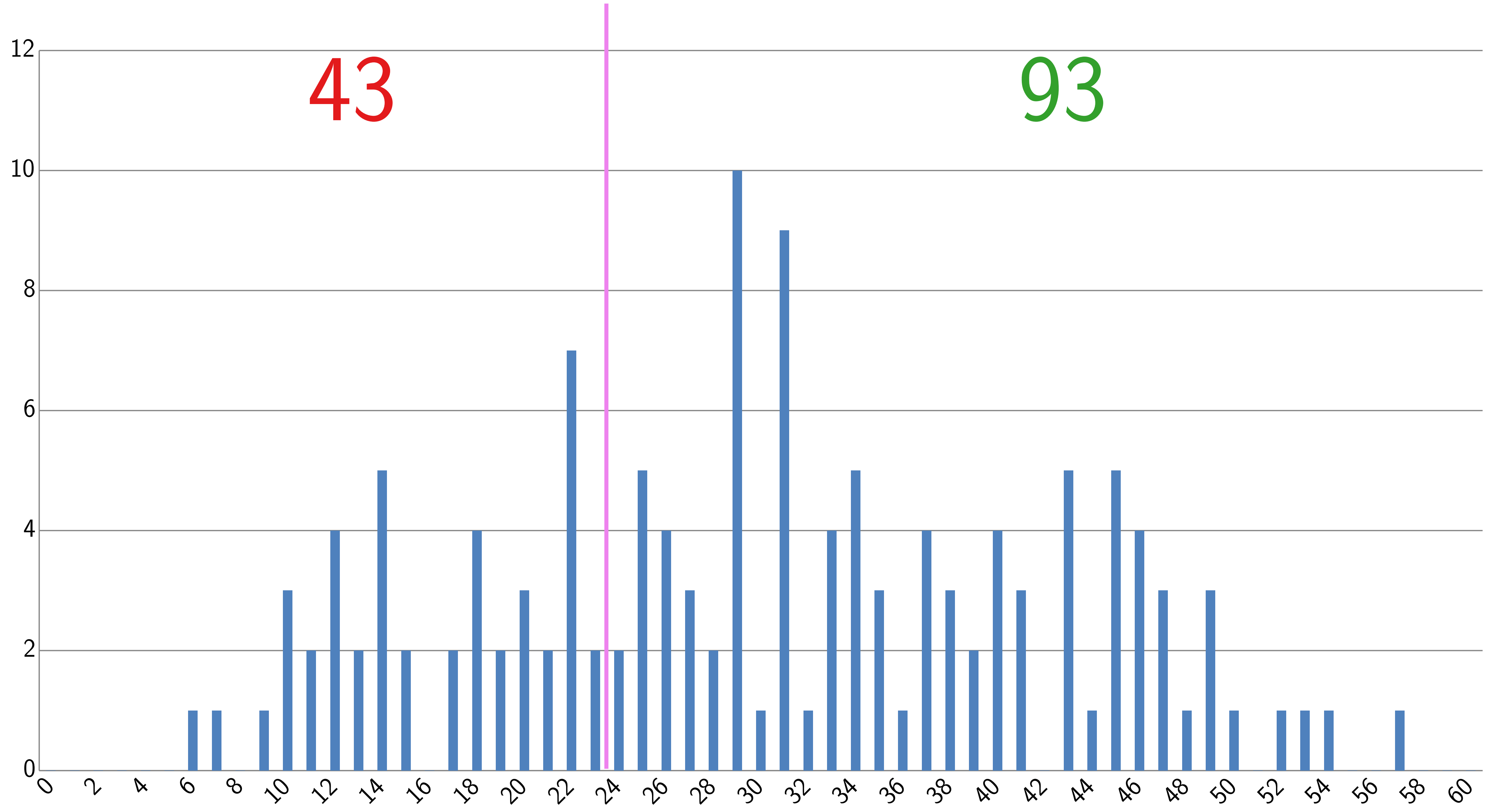
2. Zwischentest

$n = 105$; Durchschnitt $= 30{,}4$; Median $= 30$.

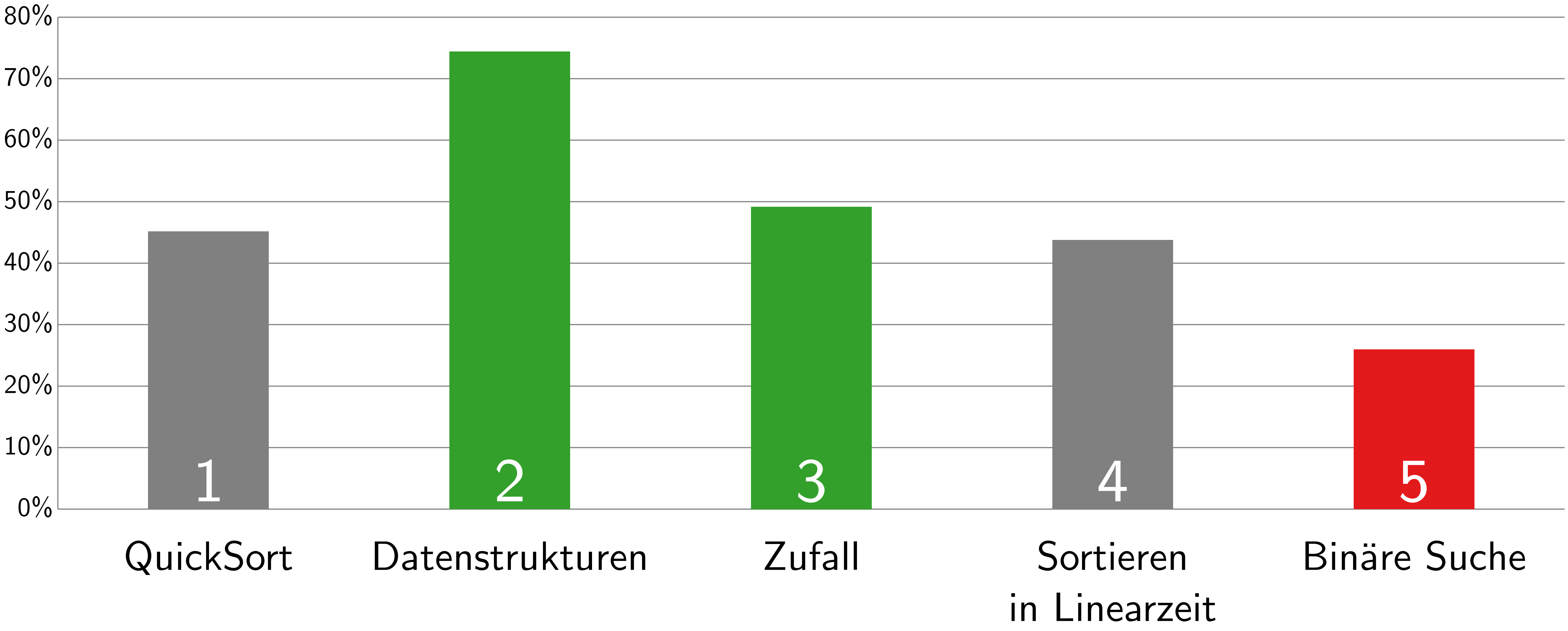


2. Zwischentest (WS 2024)

$n = 136$; Durchschnitt $= 30{,}2$; Median $= 29{,}5$



2. Zwischentest: Aufgabenübersicht



Einstiegsbeispiel: Hash-Tabellen

Frage: Wie groß macht man eine Hash-Tabelle?

Einstiegsbeispiel: Hash-Tabellen

Frage: Wie groß macht man eine Hash-Tabelle?

Ziel: So groß wie nötig, so klein wie möglich...

Einstiegsbeispiel: Hash-Tabellen

Frage: Wie groß macht man eine Hash-Tabelle?

Ziel: So groß wie nötig, so klein wie möglich...

Einstiegsbeispiel: Hash-Tabellen

Frage: Wie groß macht man eine Hash-Tabelle?

Ziel: So groß wie nötig, so klein wie möglich...

Verhindere, dass die Tabelle überläuft oder dass Operationen ineffizient werden.

Einstiegsbeispiel: Hash-Tabellen

Frage: Wie groß macht man eine Hash-Tabelle?

Ziel: So groß wie nötig, so klein wie möglich...

Verhindere, dass die Tabelle überläuft oder dass Operationen ineffizient werden.

Problem: Was tun, wenn man die maximale Anzahl zu speichernder Elemente vorab nicht kennt?

Einstiegsbeispiel: Hash-Tabellen

Frage: Wie groß macht man eine Hash-Tabelle?

Ziel: So groß wie nötig, so klein wie möglich...

Verhindere, dass die Tabelle überläuft oder dass Operationen ineffizient werden.

Problem: Was tun, wenn man die maximale Anzahl zu speichernder Elemente vorab nicht kennt?

Lösung:

Einstiegsbeispiel: Hash-Tabellen

Frage: Wie groß macht man eine Hash-Tabelle?

Ziel: So groß wie nötig, so klein wie möglich...

Verhindere, dass die Tabelle überläuft oder dass Operationen ineffizient werden.

Problem: Was tun, wenn man die maximale Anzahl zu speichernder Elemente vorab nicht kennt?

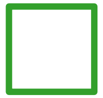
Lösung: **Dynamische** Tabellen!

Dynamische Tabellen

Idee.

Dynamische Tabellen

Idee.



Dynamische Tabellen

Idee.

$\textsc{Insert}(1)$ $\boxed{}$

Dynamische Tabellen

Idee.

$\textsc{Insert}(1)$ $\boxed{1}$

Dynamische Tabellen

Idee.

$\textsc{Insert}(1)$ $\boxed{1}$

$\textsc{Insert}(2)$

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.

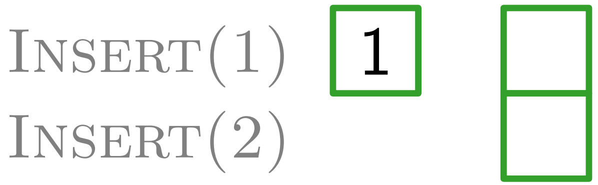
$\textsc{Insert}(1)$ $\boxed{1}$

$\textsc{Insert}(2)$

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.

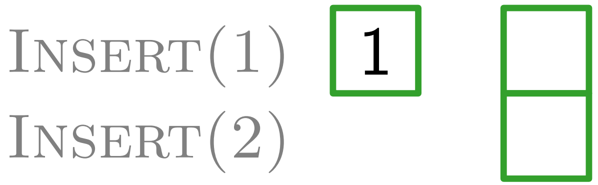


Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.

- Kopiere alle Einträge von alter in neue Tabelle.

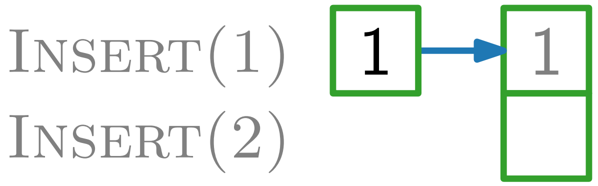


Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.

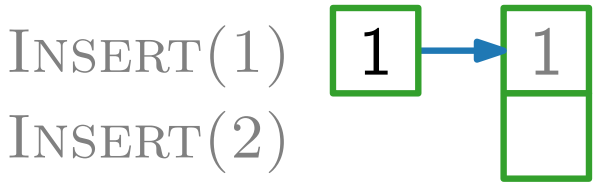
- Kopiere alle Einträge von alter in neue Tabelle.



Dynamische Tabellen

Idee.

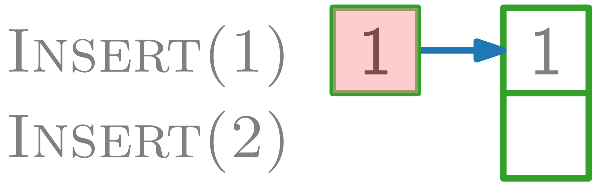
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

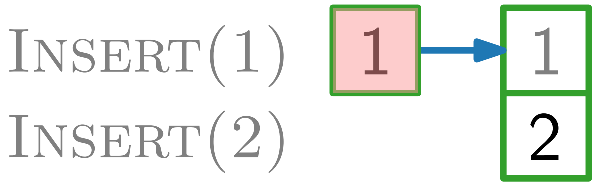
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

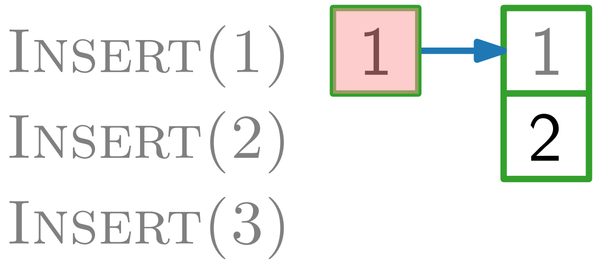
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

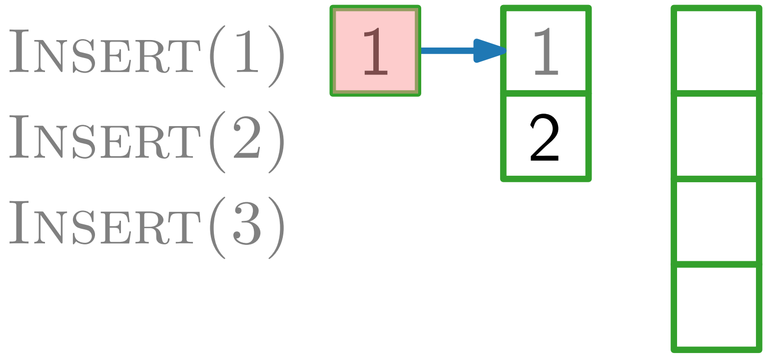
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

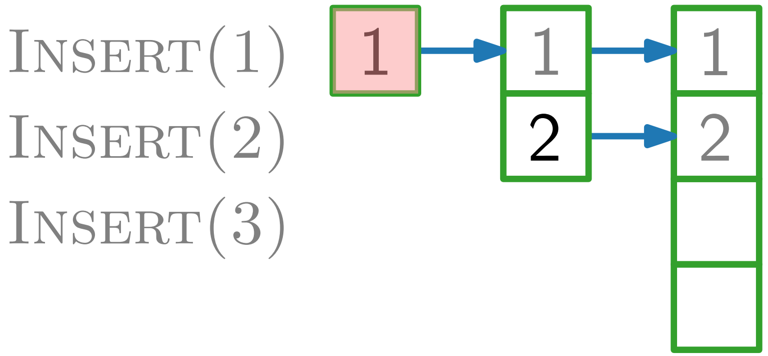
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

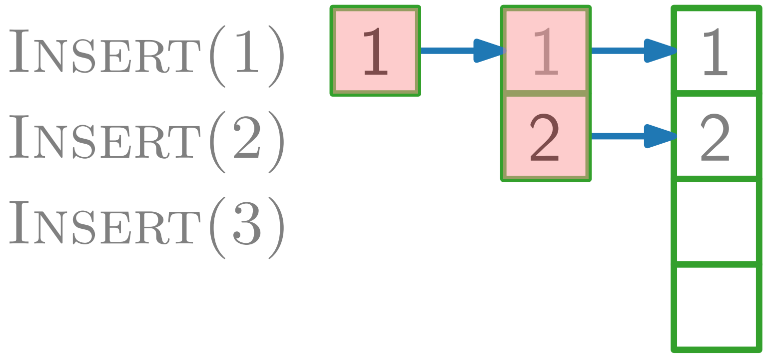
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

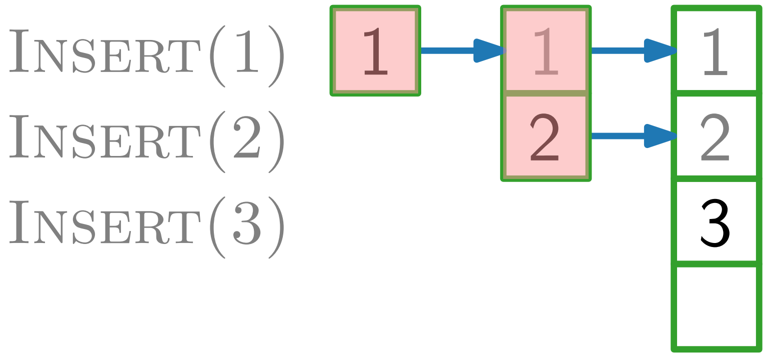
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

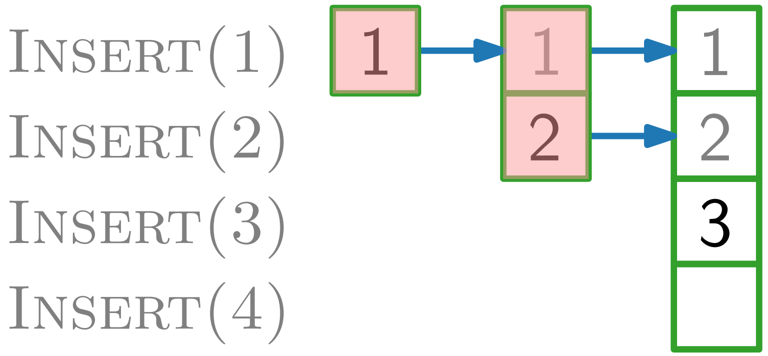
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

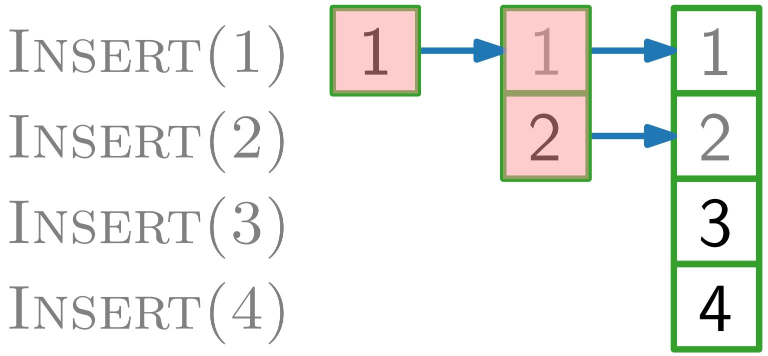
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

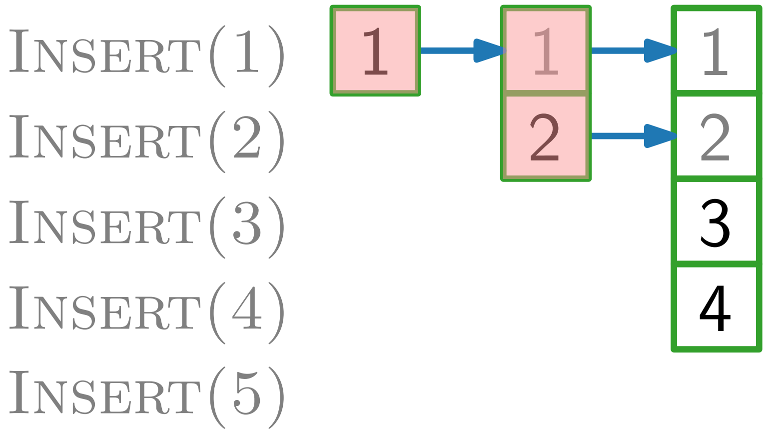
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

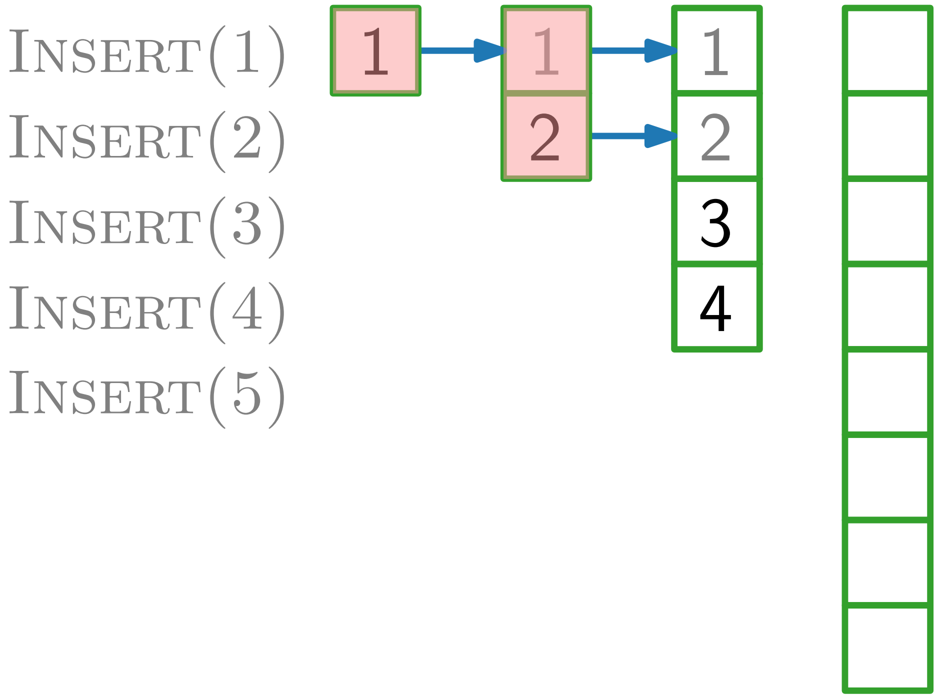
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

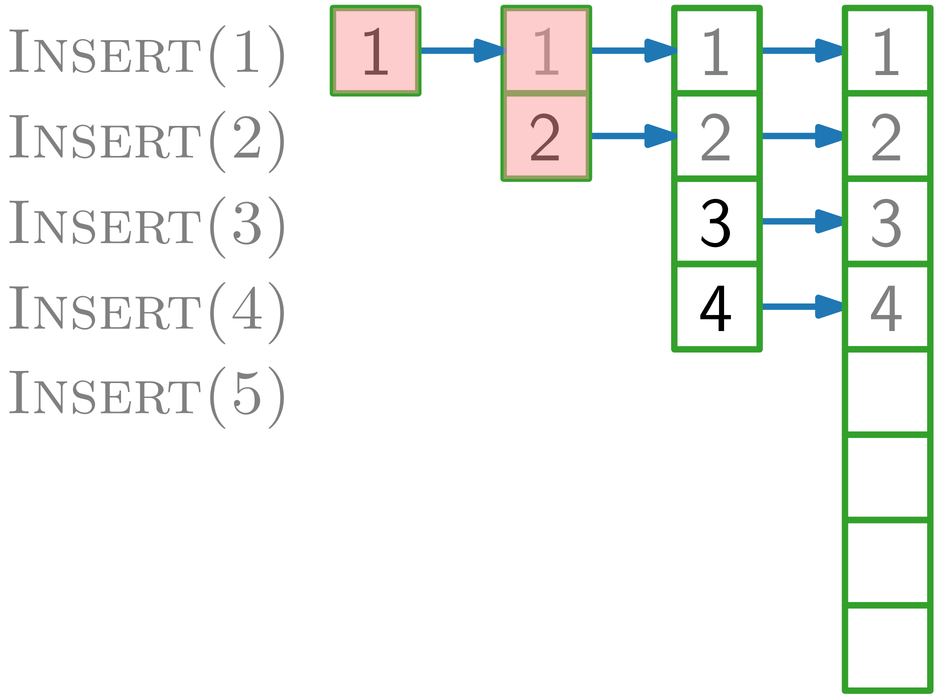
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

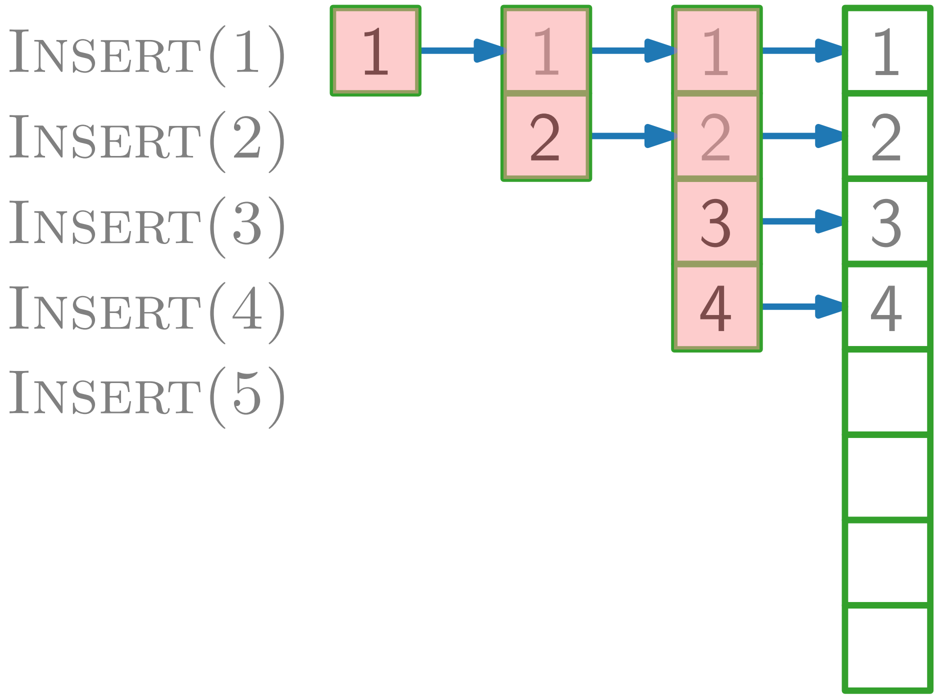
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

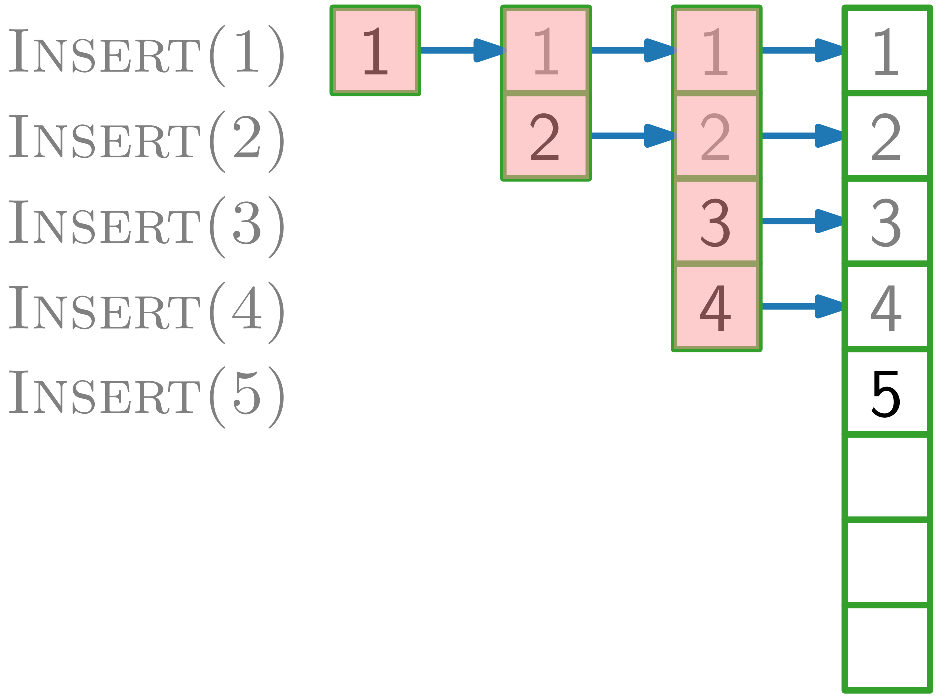
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

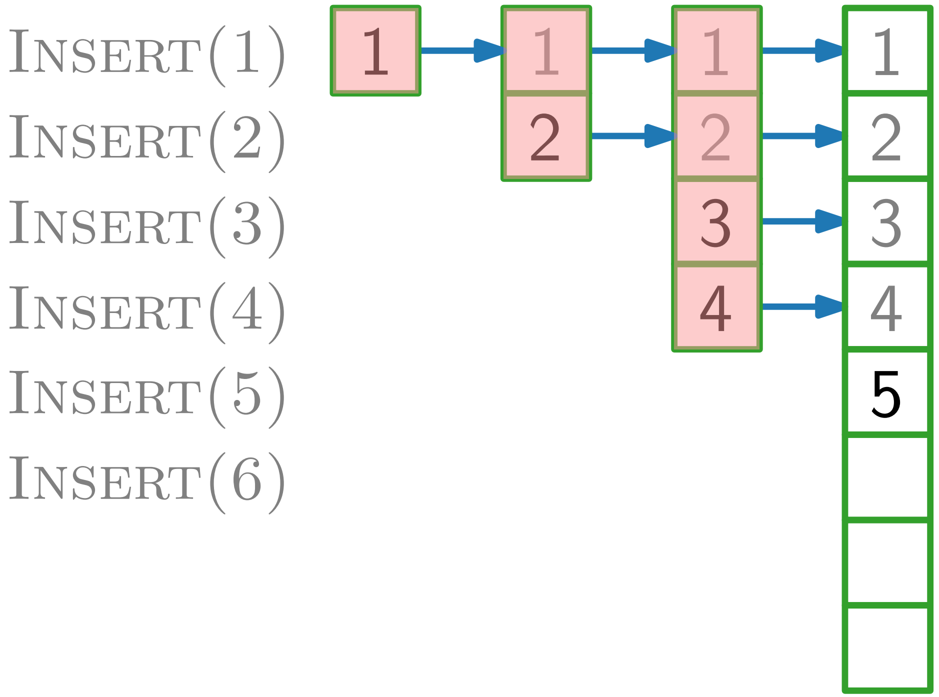
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

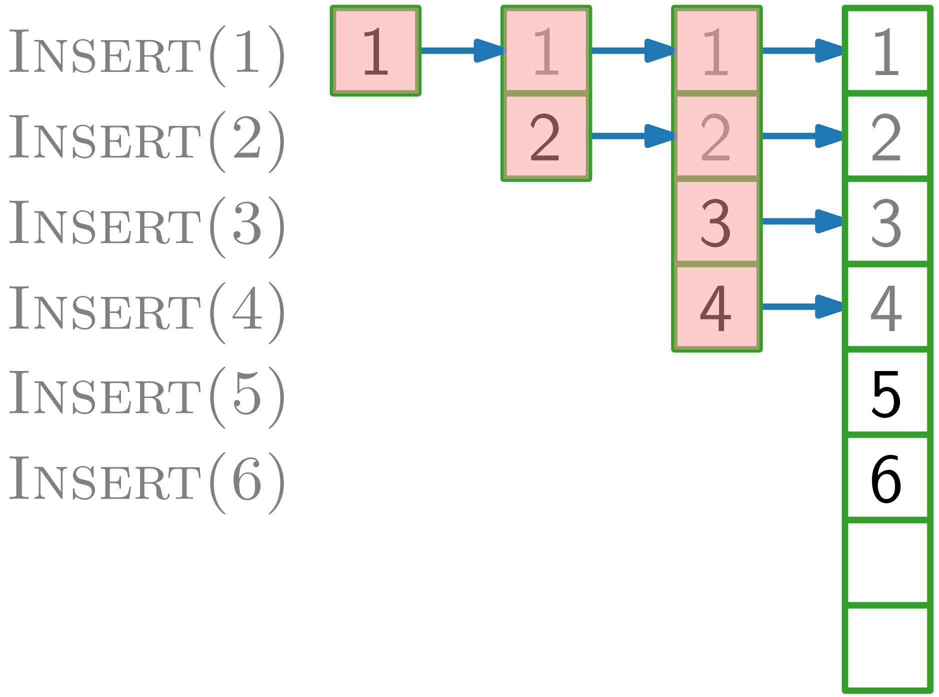
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

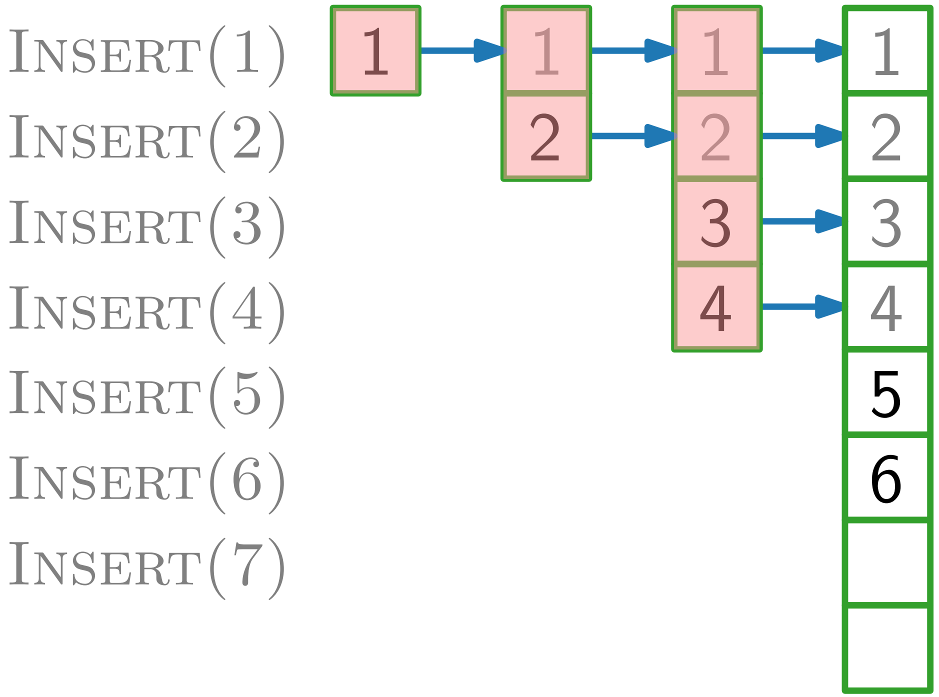
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



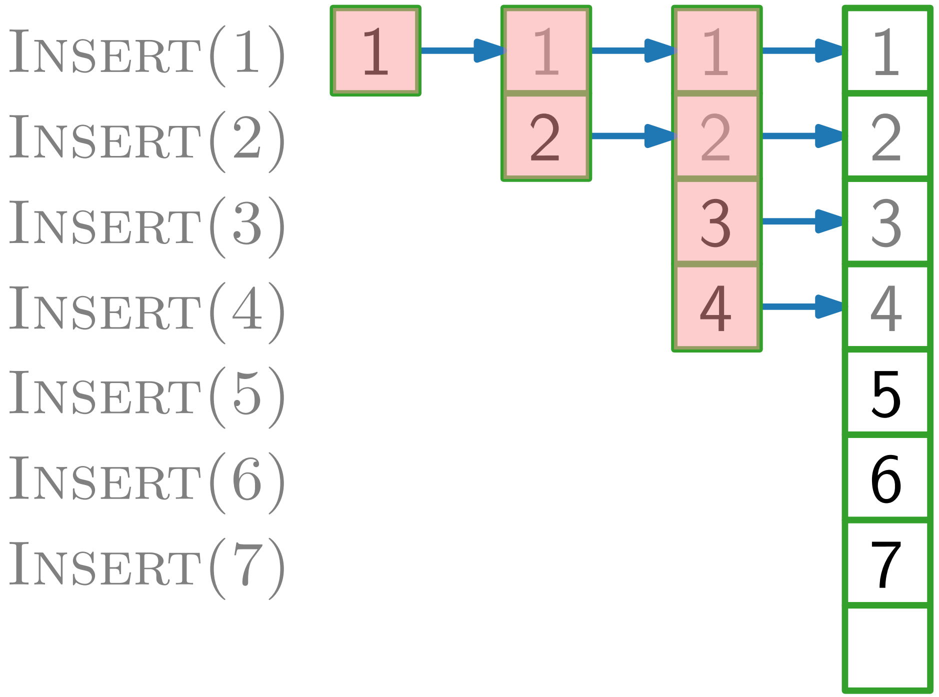
Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.

- Kopiere alle Einträge von alter in neue Tabelle.

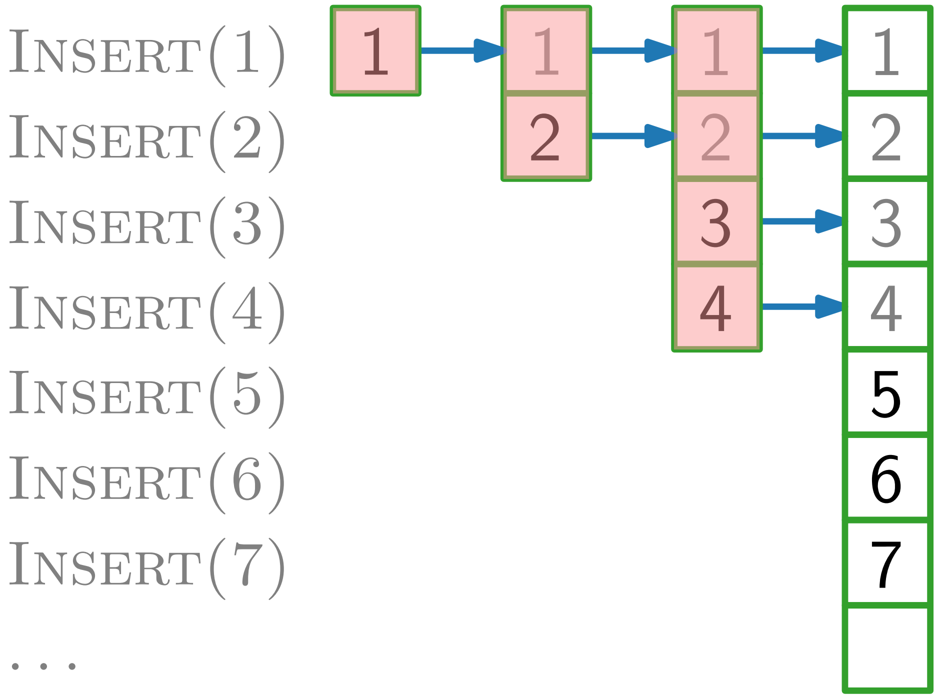
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

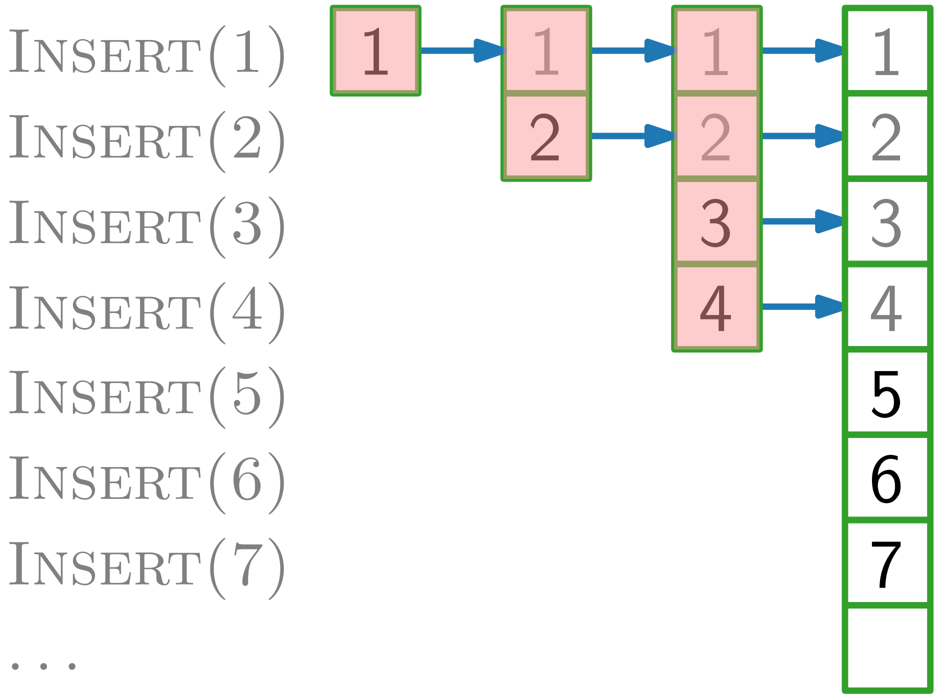
- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.

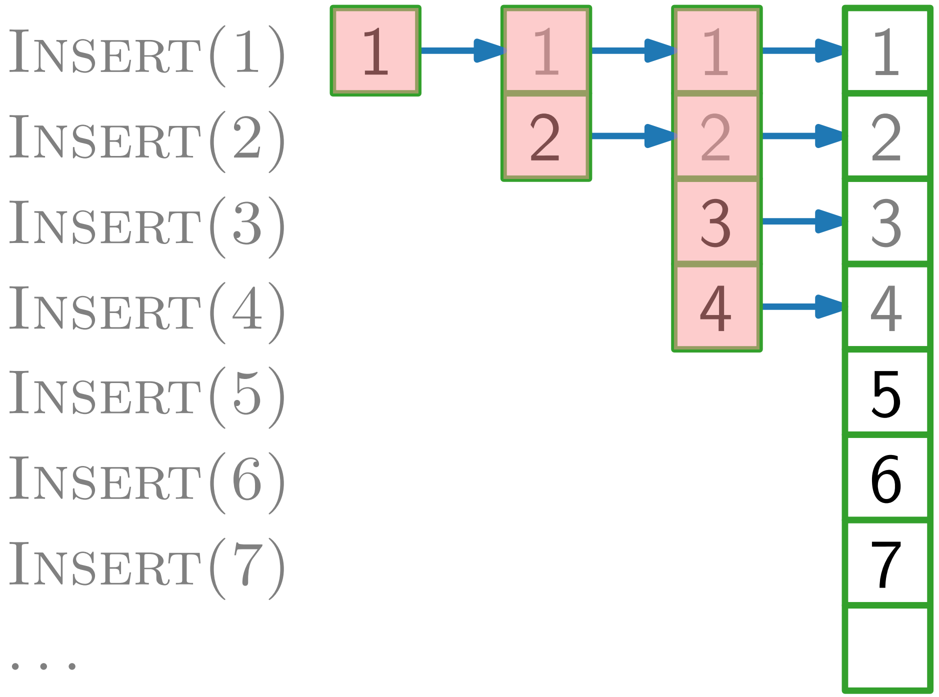


Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

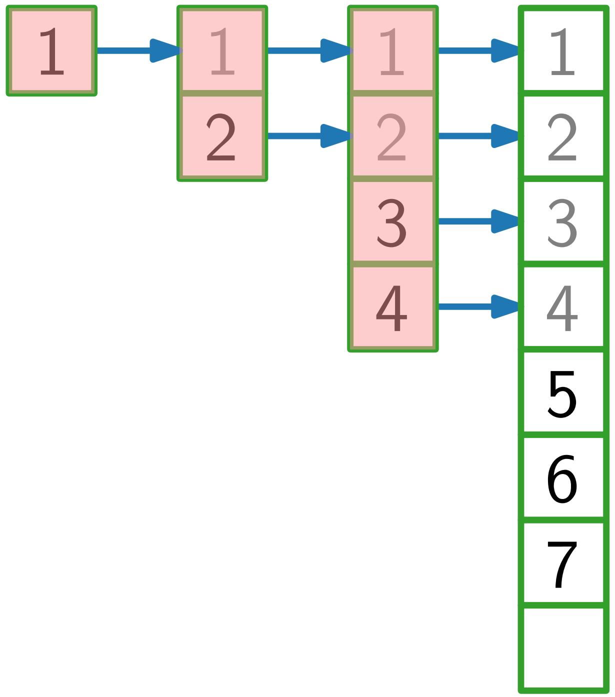
Antwort.

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)
INSERT(6)
INSERT(7)
...



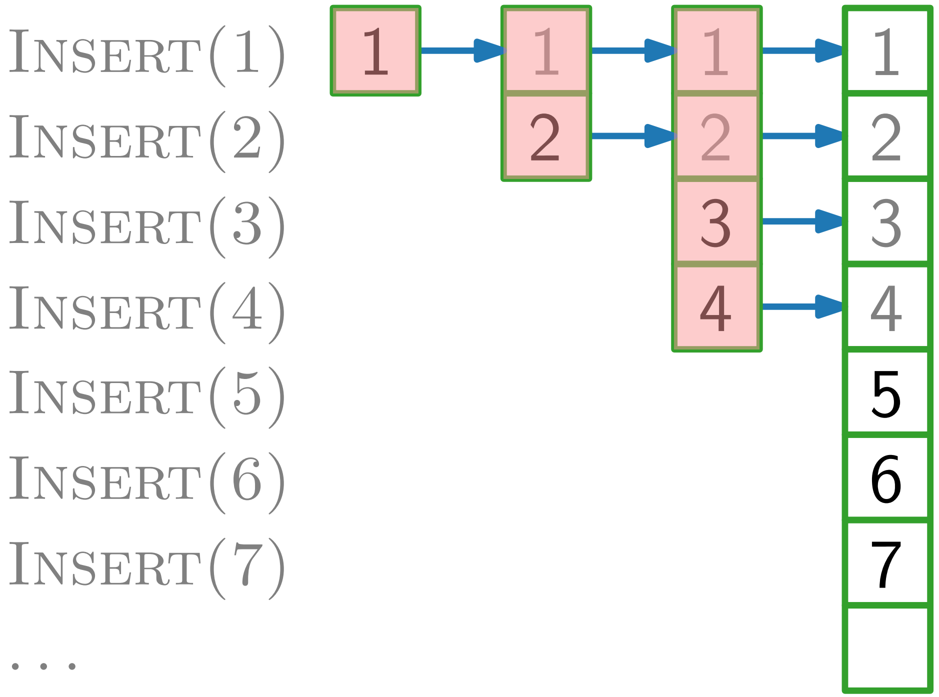
Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

Antwort.
- Tabelle wird genau mal kopiert.

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



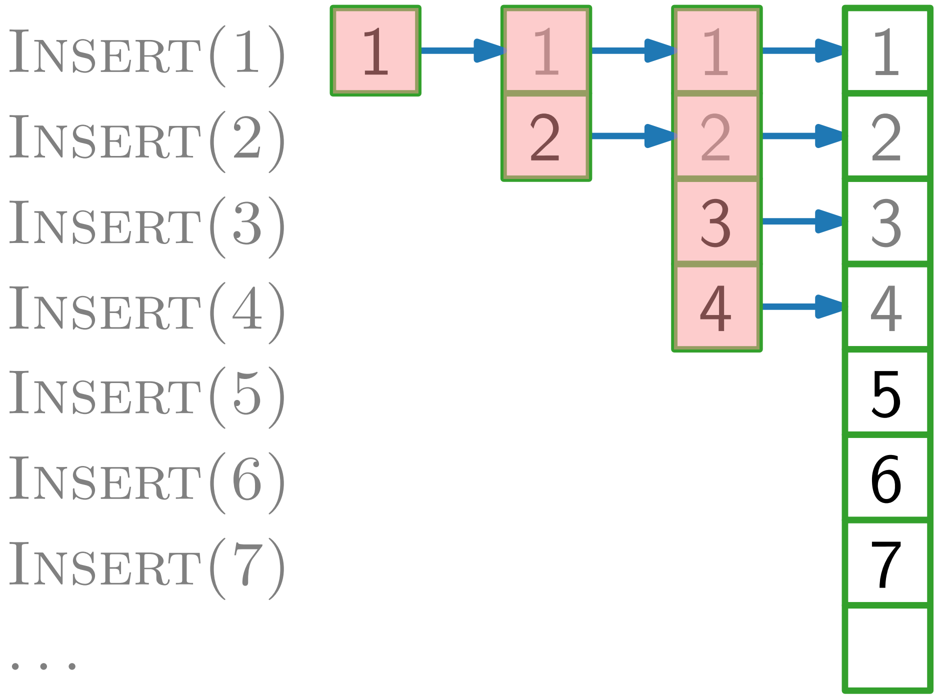
Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

Antwort.
- Tabelle wird genau mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

Antwort.
- Tabelle wird genau mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand

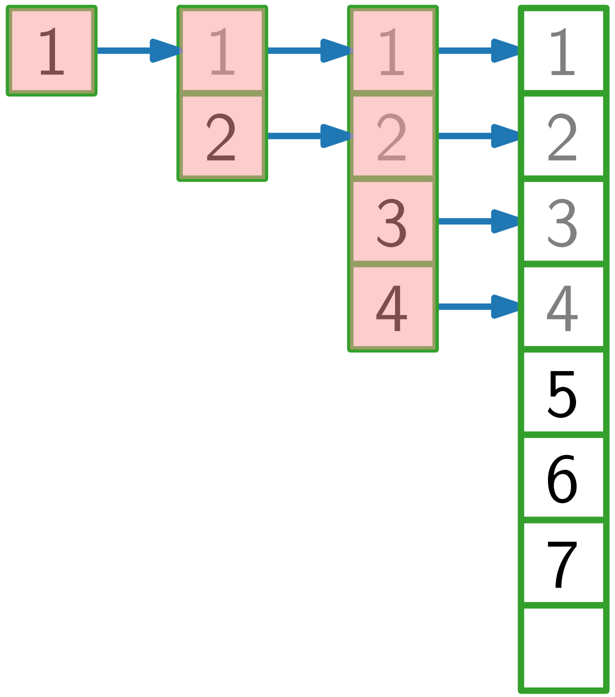
Also ist der Gesamtaufwand

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist, fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)
INSERT(6)
INSERT(7)
...



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

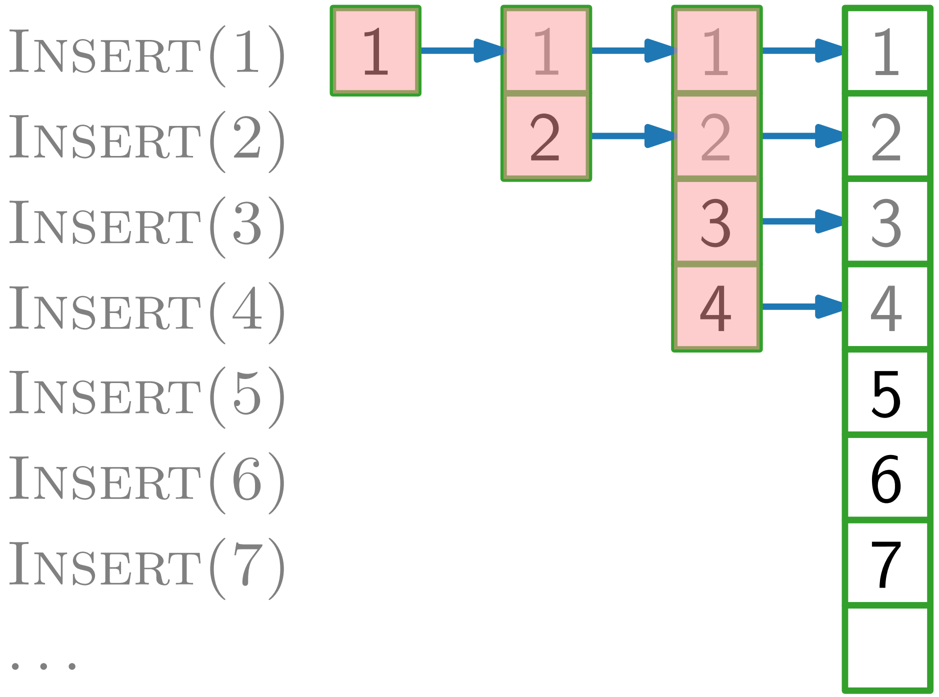
Antwort.
- Tabelle wird genau $\lceil \log_2 n \rceil$ mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand

Also ist der Gesamtaufwand

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

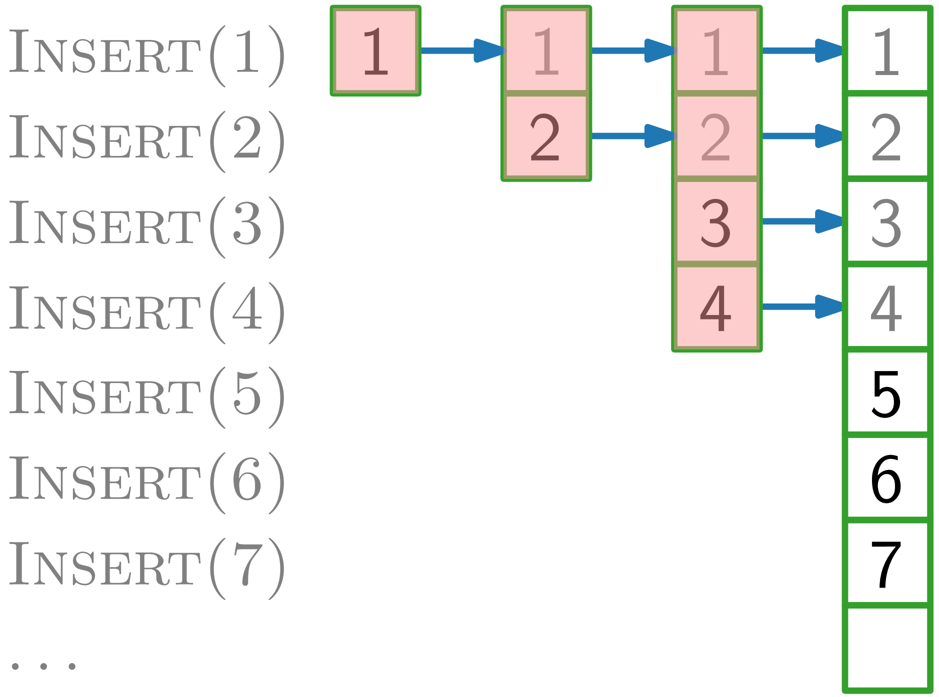
Antwort.
- Tabelle wird genau $\lceil \log_2 n \rceil$ mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand $\Theta(n)$.

Also ist der Gesamtaufwand

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

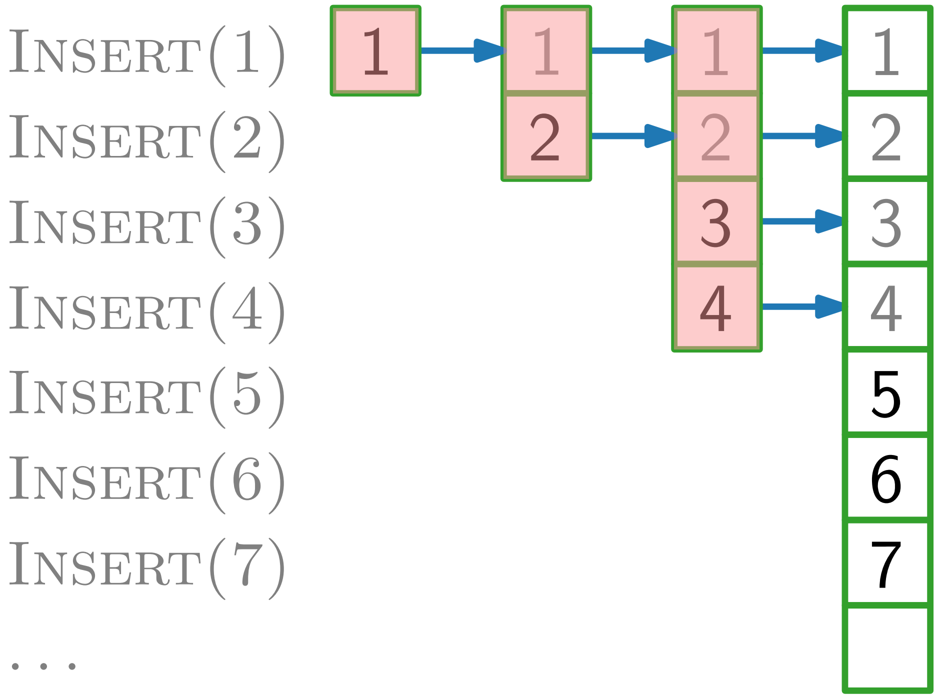
Antwort.
- Tabelle wird genau $\lceil \log_2 n \rceil$ mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand $\Theta(n)$.

Also ist der Gesamtaufwand $\Theta(n \log n)$.

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

Antwort.

- Tabelle wird genau $\lceil \log_2 n \rceil$ mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand $\Theta(n)$.

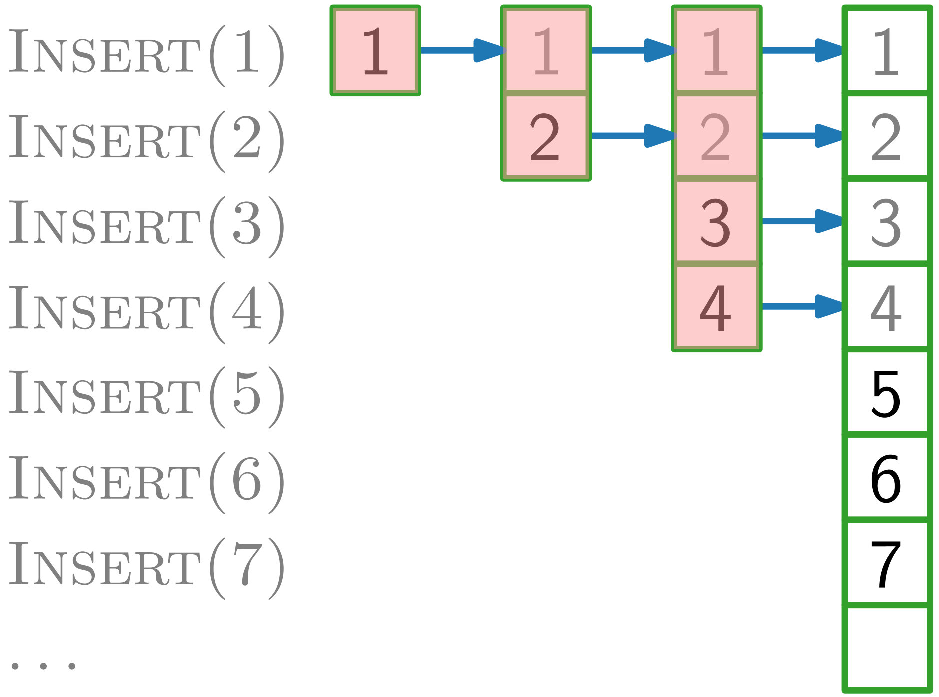
Also ist der Gesamtaufwand $\Theta(n \log n)$.

falsch!

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

Antwort.

- Tabelle wird genau $\lceil \log_2 n \rceil$ mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand $\Theta(n)$.

Also ist der Gesamtaufwand $\cancel{\Theta}(n \log n)$.

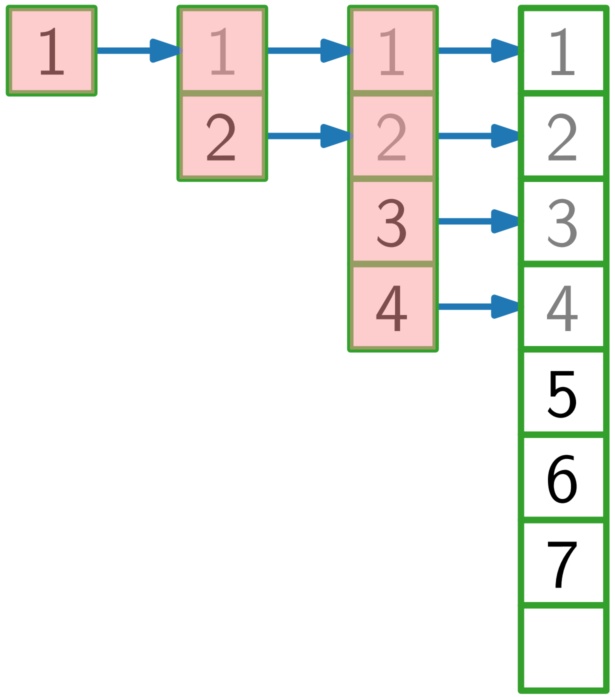
falsch! $\mathcal{O}$

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)
INSERT(6)
INSERT(7)
...



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

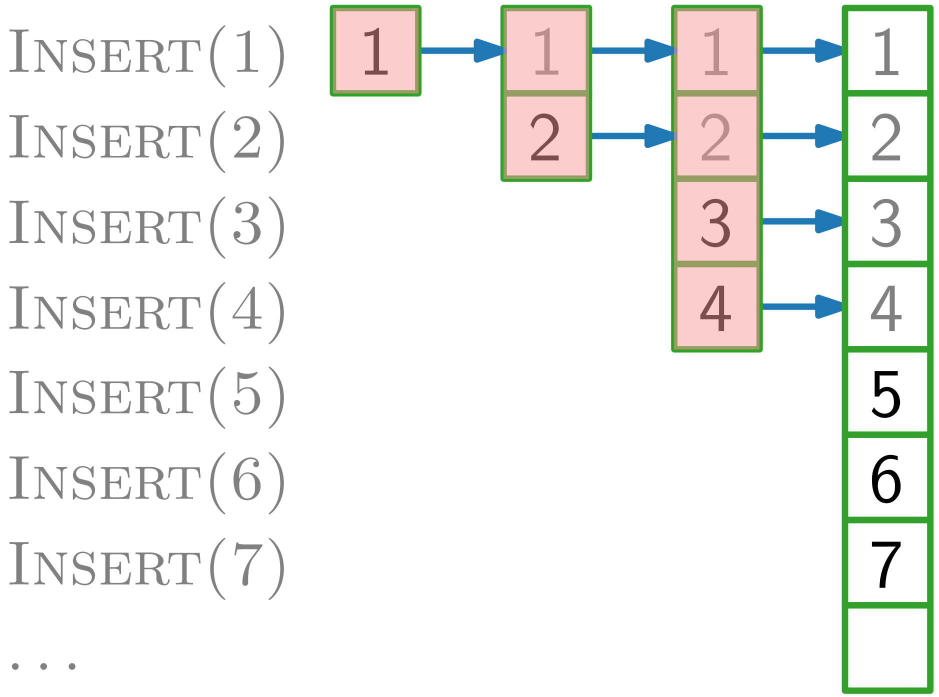
Antwort.
- Tabelle wird genau $\lceil \log_2 n \rceil$ mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand $\Theta(n)$.

Also ist der Gesamtaufwand $\cancel{\Theta}(n \log n)$, genauer
falsch! $\mathcal{O}$

Dynamische Tabellen

Idee.

- Wenn die Tabelle voll ist,
  fordere eine doppelt so große an.
- Kopiere alle Einträge von alter in neue Tabelle.
- Gib Speicher für alte Tabelle frei.



Analyse. Welche Laufzeit benötigen n Einfügeoperationen im schlimmsten Fall?

Antwort.

- Tabelle wird genau $\boxed{\lceil \log_2 n \rceil}$ mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand $\boxed{\Theta(n)}$.

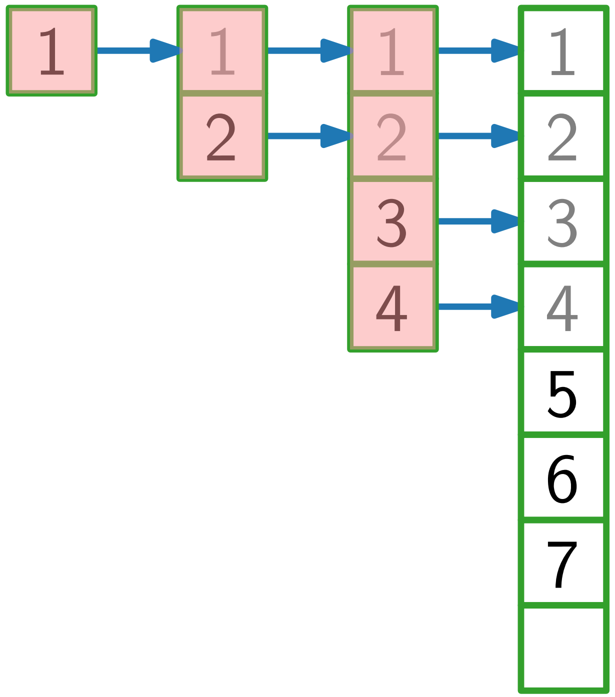
Also ist der Gesamtaufwand $\boxed{\cancel{\Theta}(n \log n)}$, genauer $\Theta(n)$.

$\mathcal{O}$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

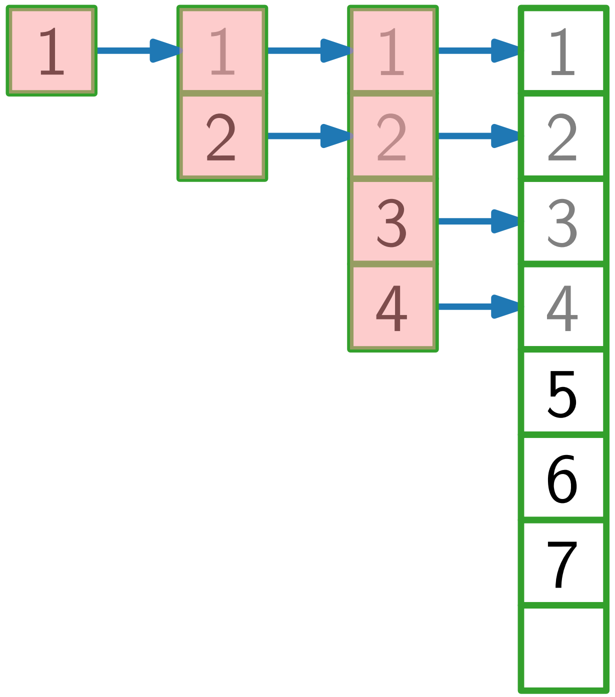
$c_i =$ Kosten fürs i-te Einfügen.



Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei
$c_i =$ Kosten fürs i-te Einfügen.

i									
$size_i$									

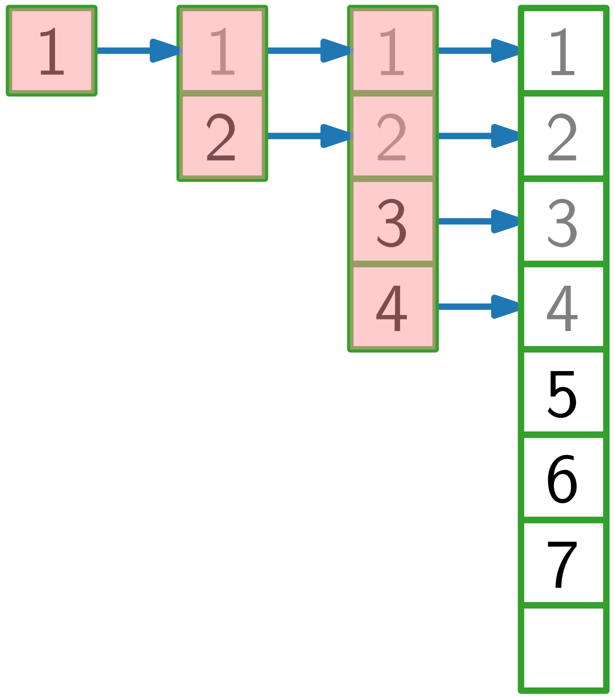


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1								
$size_i$	1								

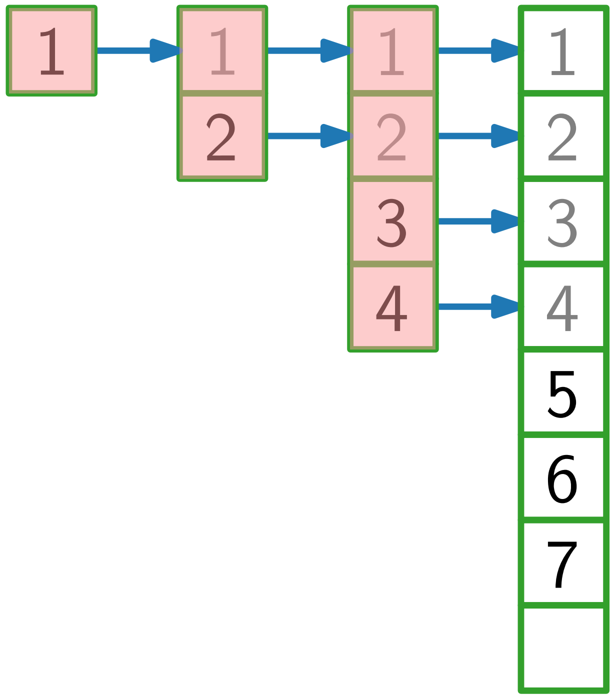


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

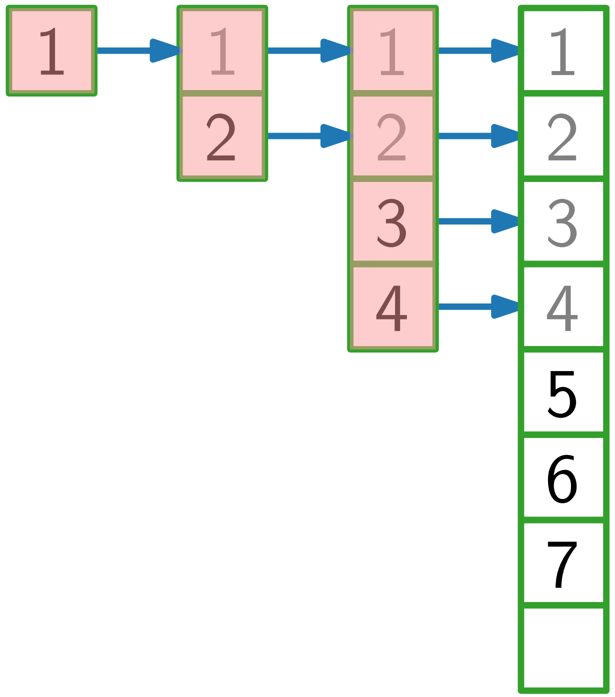
i	1	2							
$size_i$	1								



Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei
$c_i =$ Kosten fürs i-te Einfügen.

i	1	2							
$size_i$	1	2							

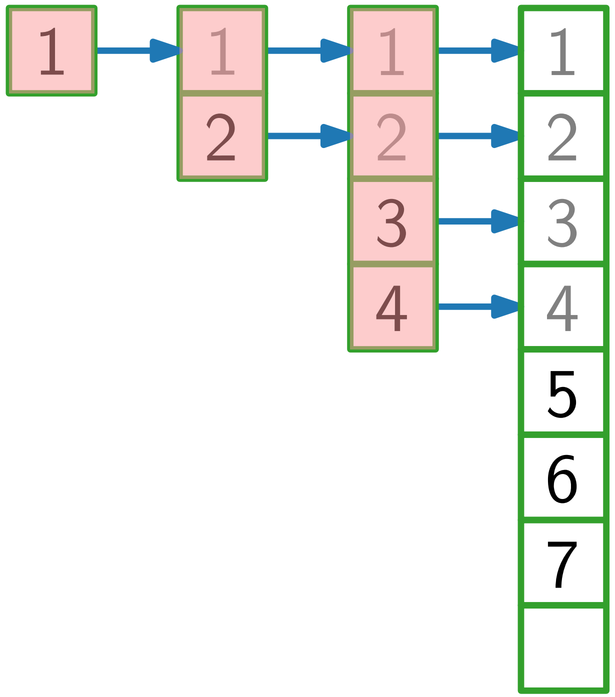


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3						
$size_i$	1	2							

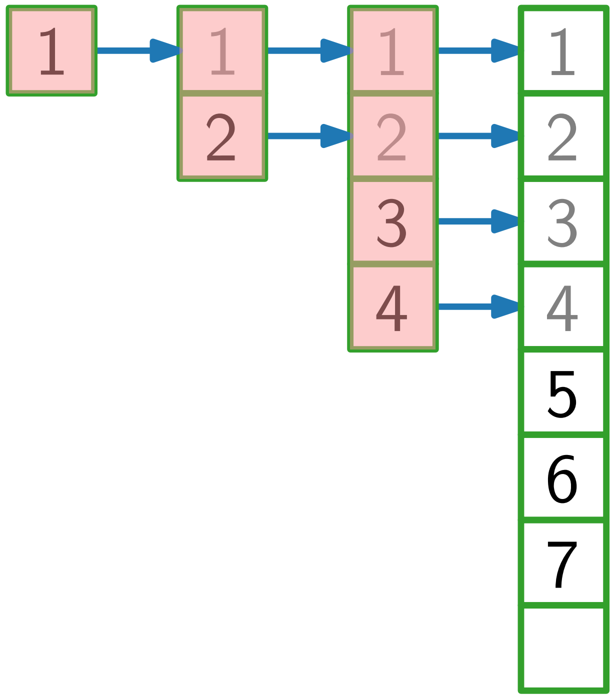


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

c_i = Kosten fürs i-te Einfügen.

i	1	2	3						
$size_i$	1	2	4						

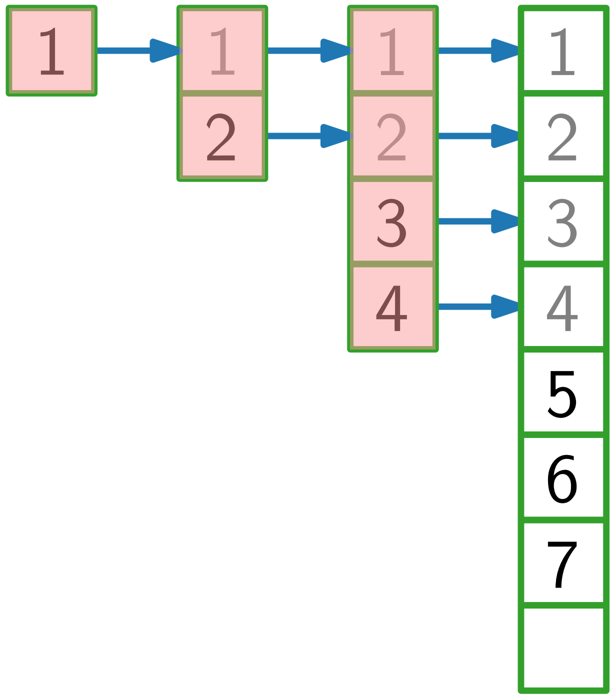


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4					
$size_i$	1	2	4	4					

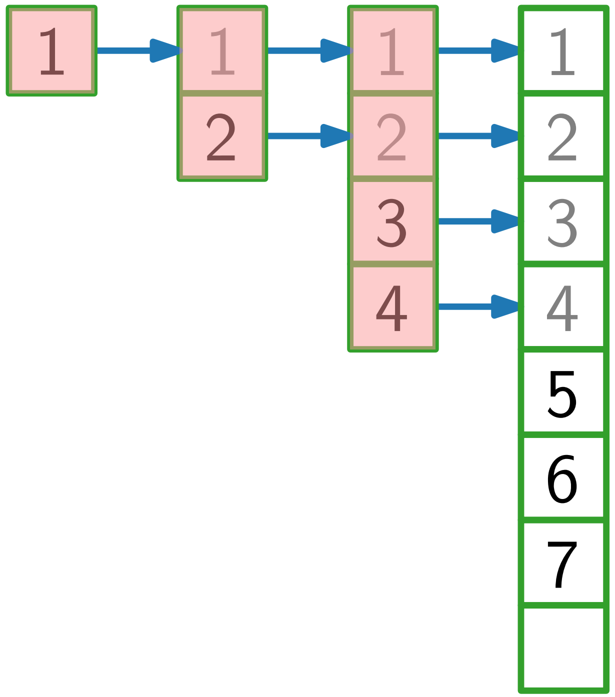


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	
$size_i$	1	2	4	4	8	8	8	8	

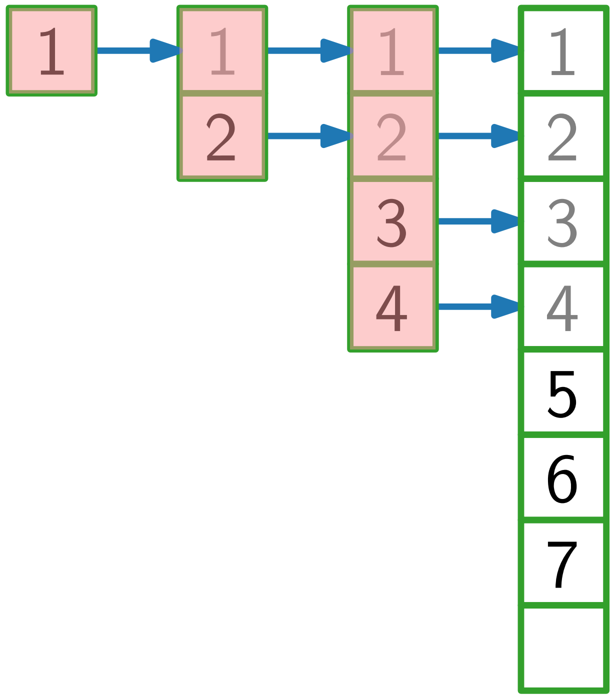


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9
$size_i$	1	2	4	4	8	8	8	8	16

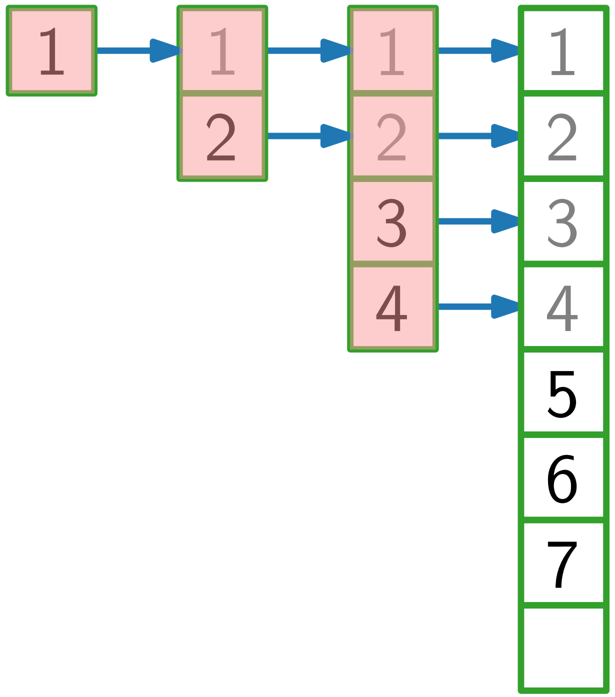


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9
$size_i$	1	2	4	4	8	8	8	8	16
c_i									



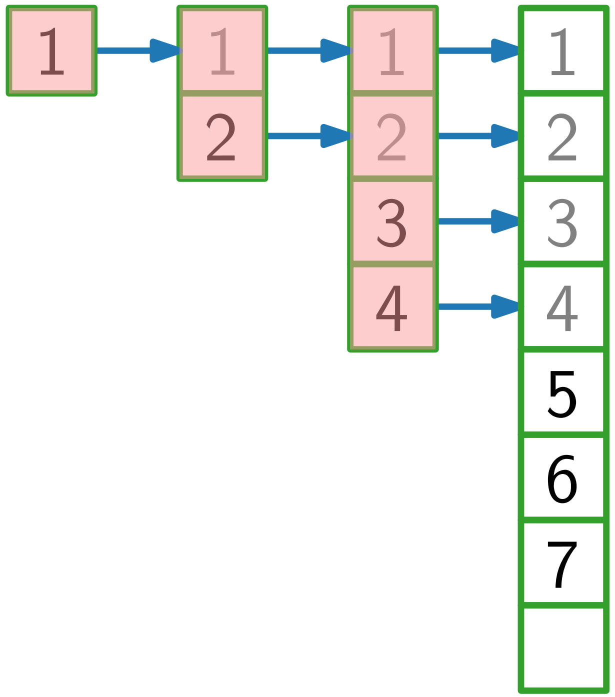
Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9
$size_i$	1	2	4	4	8	8	8	8	16
c_i									

$\leftarrow$ Einfügen



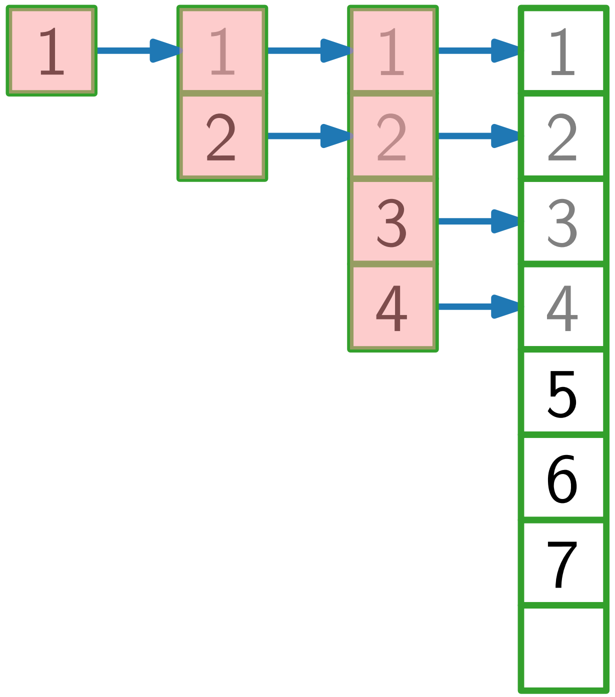
Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9
$size_i$	1	2	4	4	8	8	8	8	16
c_i	1	1	1	1	1	1	1	1	1

$\leftarrow$ Einfügen

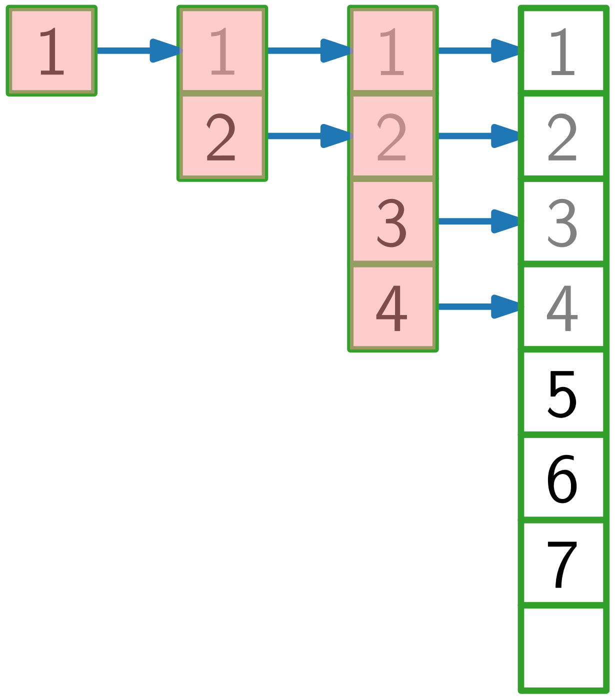


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

c_i = Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	$\leftarrow$ Einfügen
										$\leftarrow$ Kopieren

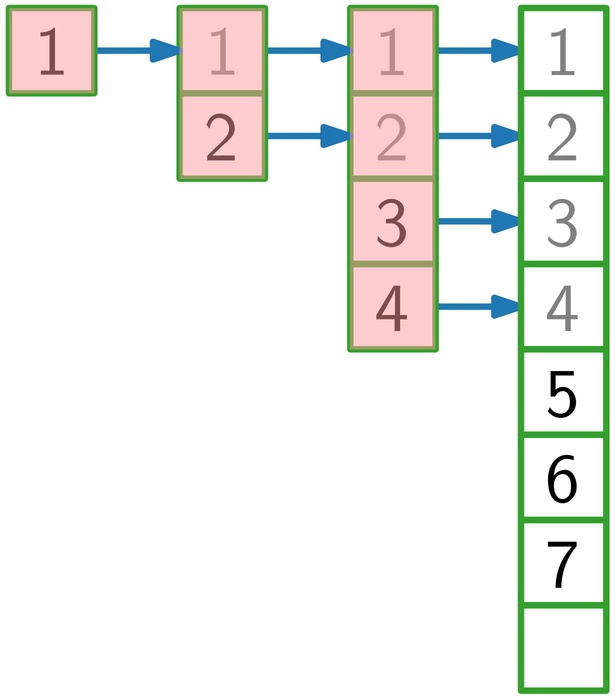


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	$\leftarrow$ Einfügen
		1	2		4				8	$\leftarrow$ Kopieren

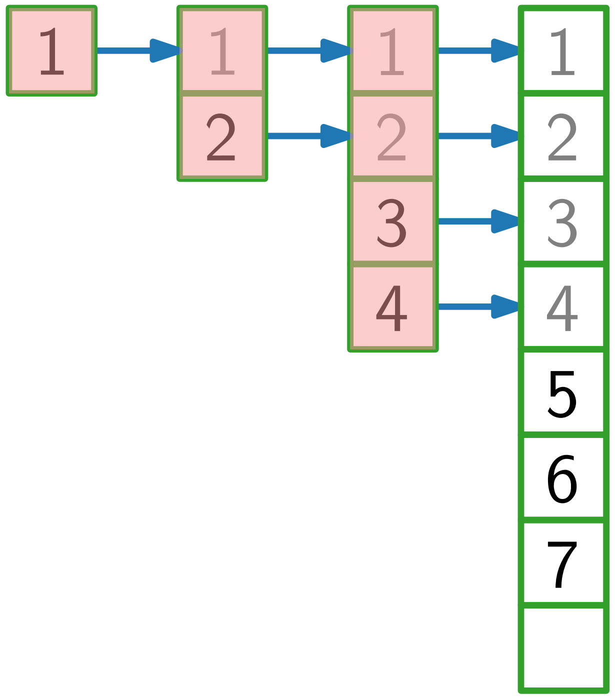


Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	$\leftarrow$ Einfügen
		1	2		4				8	$\leftarrow$ Kopieren



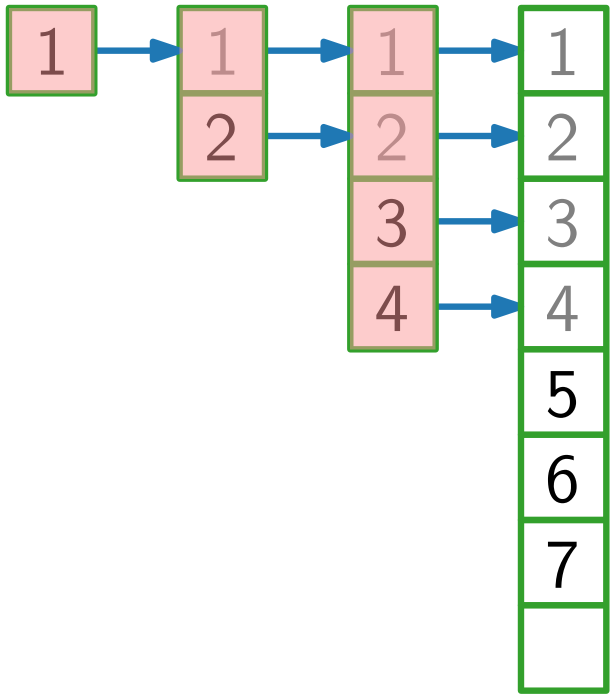
Also betragen die Kosten für n Einfügeoperationen

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	$\leftarrow$ Einfügen
		1	2		4				8	$\leftarrow$ Kopieren



Also betragen die Kosten für n Einfügeoperationen

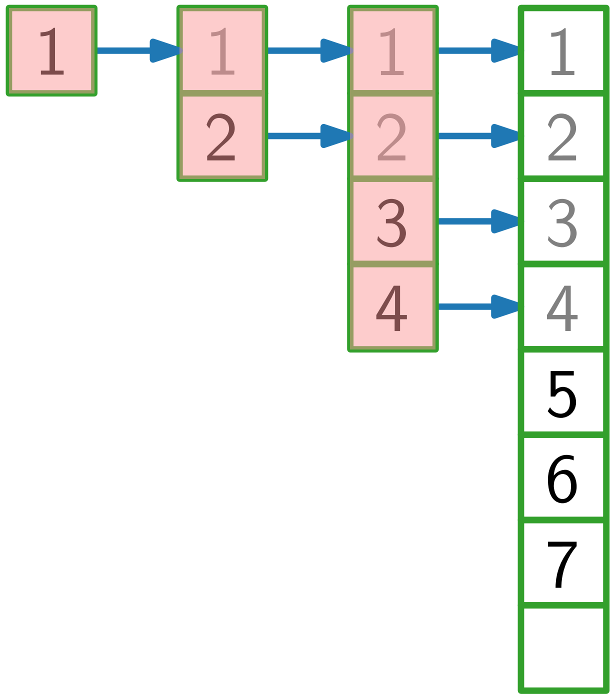
$$\sum_{1}^{n} c_i =$$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

c_i = Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren



Also betragen die Kosten für n Einfügeoperationen

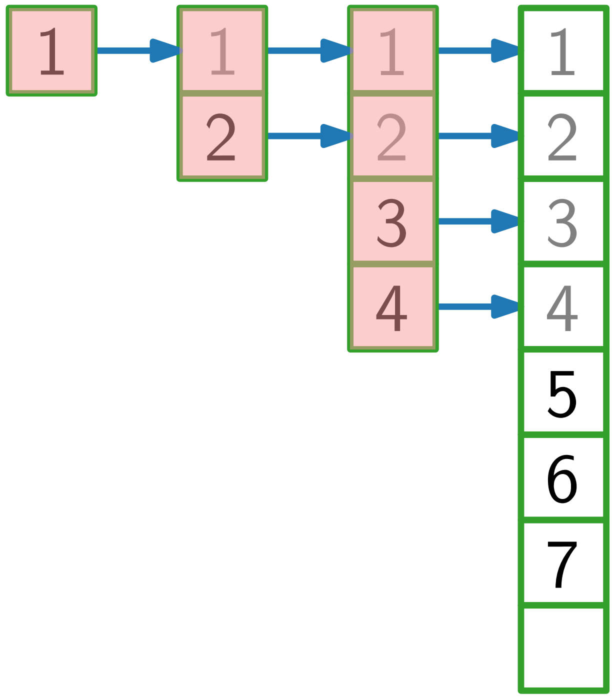
$$\sum_{1}^{n} c_i = \square + \sum_{j=0}$$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

c_i = Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren



Also betragen die Kosten für n Einfügeoperationen

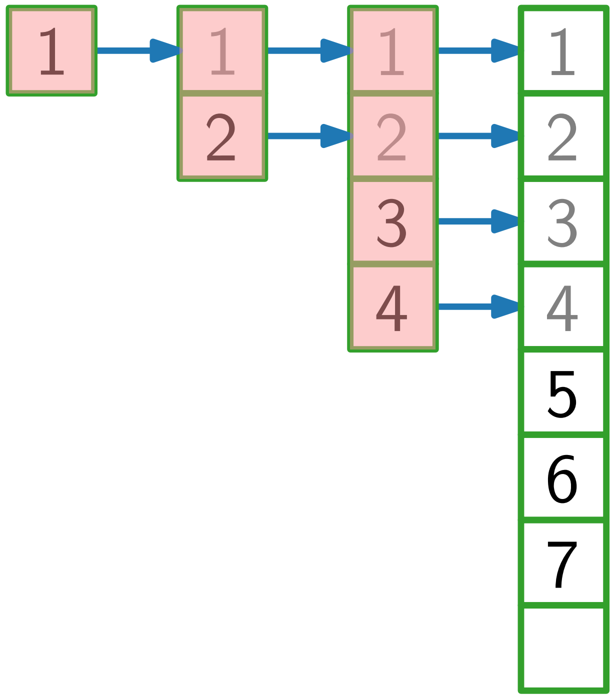
$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}$$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

c_i = Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren



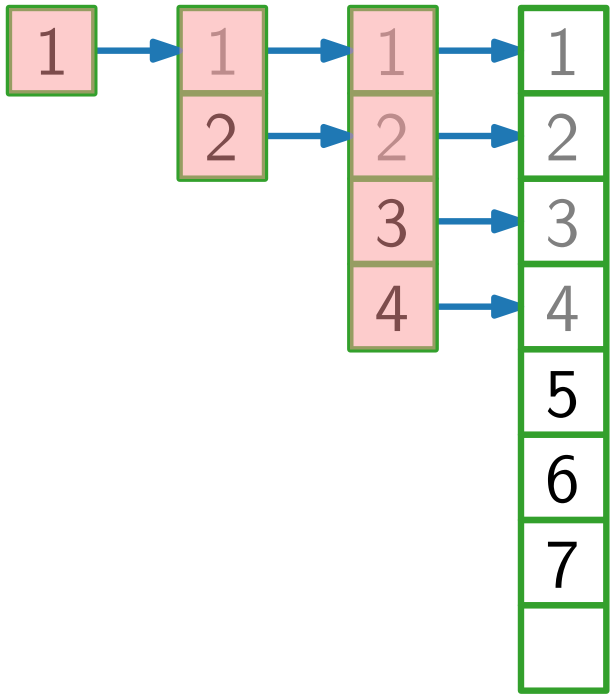
Also betragen die Kosten für n Einfügeoperationen

$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0} 2^j$$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei
c_i = Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren



Also betragen die Kosten für n Einfügeoperationen

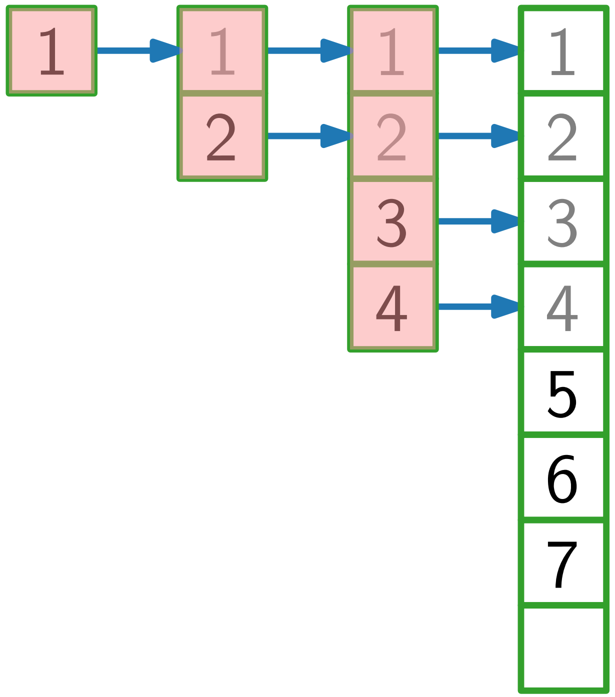
$$\sum_{1}^{n} c_i = n + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j$$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	$\leftarrow$ Einfügen
		1	2		4				8	$\leftarrow$ Kopieren



Also betragen die Kosten für n Einfügeoperationen

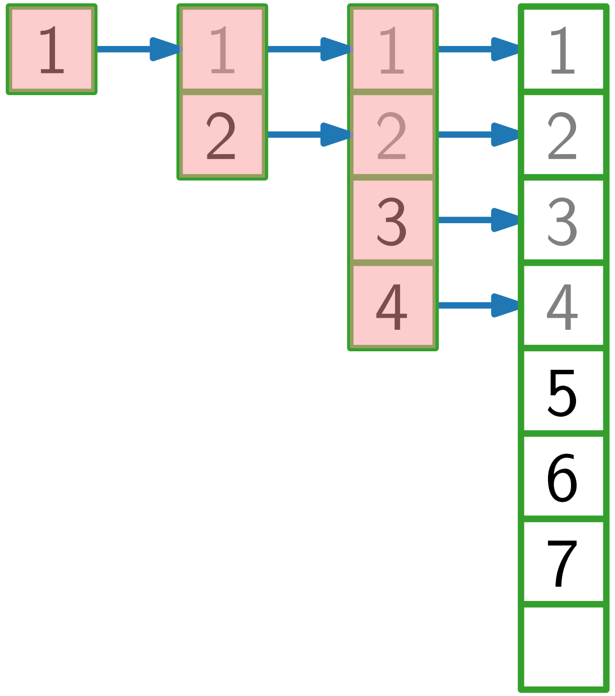
$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \le$$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren



Also betragen die Kosten für n Einfügeoperationen

$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq$$

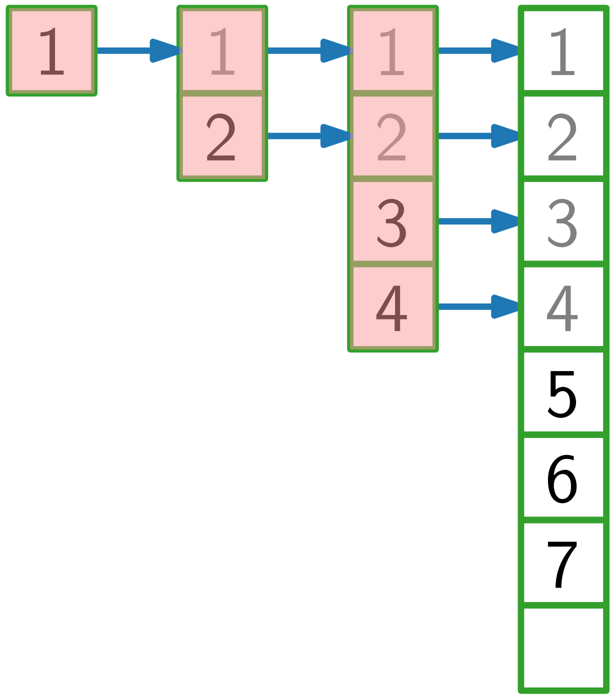
2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

c_i = Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren



Also betragen die Kosten für n Einfügeoperationen

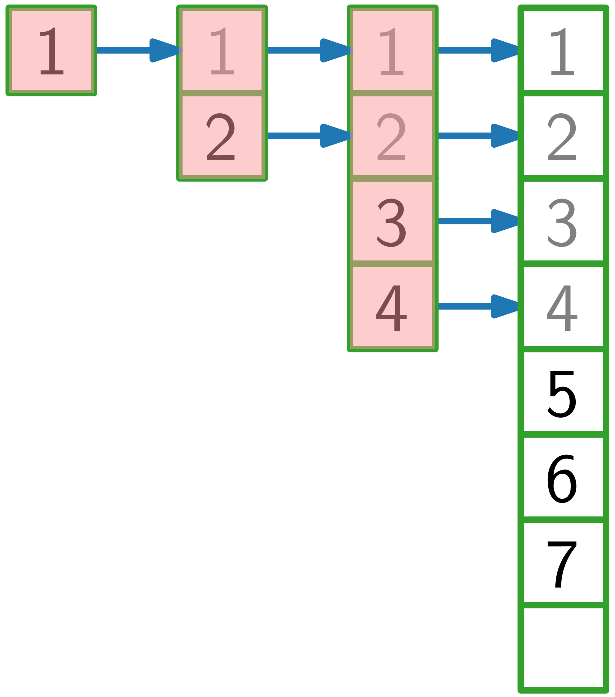
$$\sum_{1}^{n} c_i = n + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq n + \frac{1 - 2^{\log_2(n-1)+1}}{1-2}$$

2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei
c_i = Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren



Also betragen die Kosten für n Einfügeoperationen

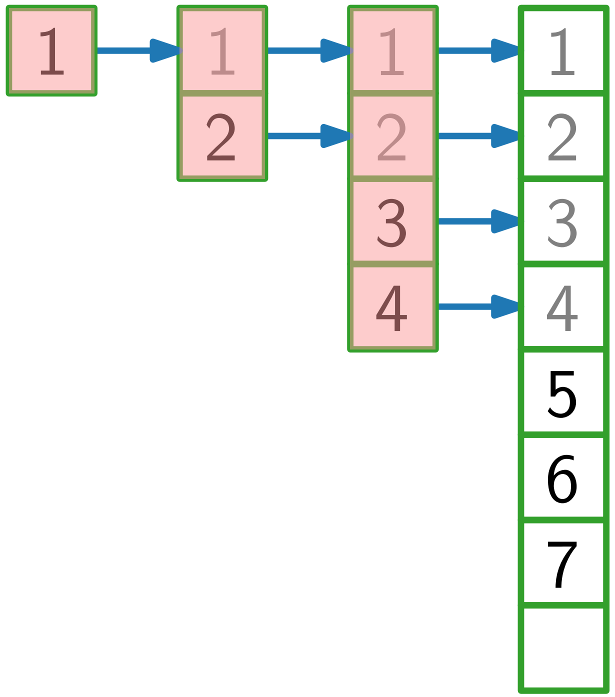
$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq n + \frac{1 - 2^{\log_2(n-1)+1}}{1-2} = n + 2^{\log_2(n-1)+1} - 1$$

2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

c_i = Kosten fürs i-te Einfügen.



i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	$\leftarrow$ Einfügen
		1	2		4				8	$\leftarrow$ Kopieren

Also betragen die Kosten für n Einfügeoperationen

$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq n + \frac{1 - 2^{\log_2(n-1)+1}}{1-2} = n + 2^{\log_2(n-1)+1} - 1$$

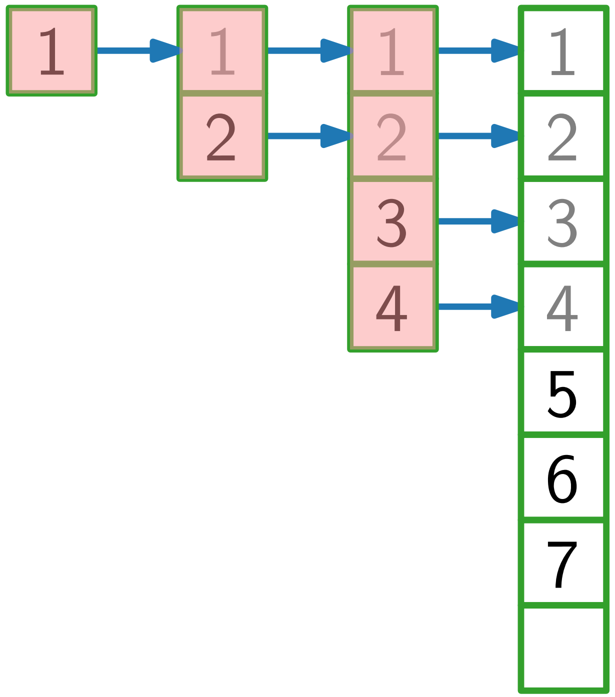
$$= n + 2 \cdot 2^{\log_2(n-1)} - 1$$

2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.



i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren

Also betragen die Kosten für n Einfügeoperationen

$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq n + \frac{1 - 2^{\log_2(n-1)+1}}{1-2} = n + 2^{\log_2(n-1)+1} - 1$$

2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

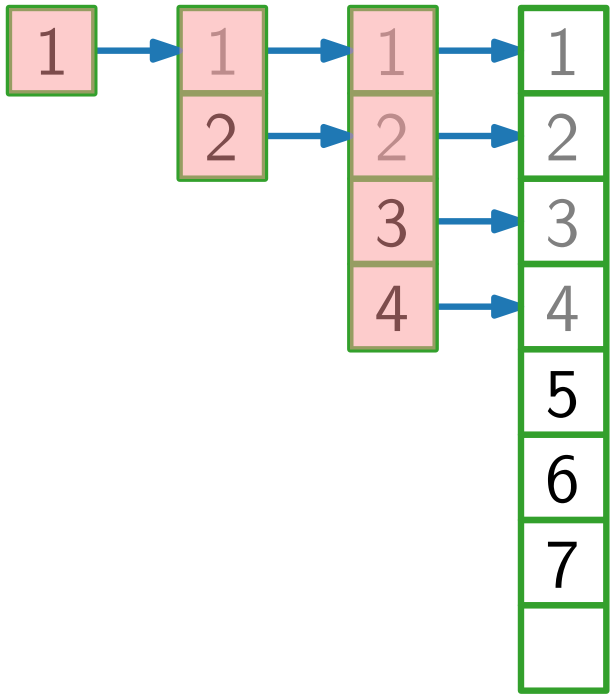
$$= n + 2 \cdot 2^{\log_2(n-1)} - 1$$

$$= n + 2(n-1) - 1$$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei
c_i = Kosten fürs i-te Einfügen.

i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren



Also betragen die Kosten für n Einfügeoperationen

$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq n + \frac{1 - 2^{\log_2(n-1)+1}}{1-2} = n + 2^{\log_2(n-1)+1} - 1$$

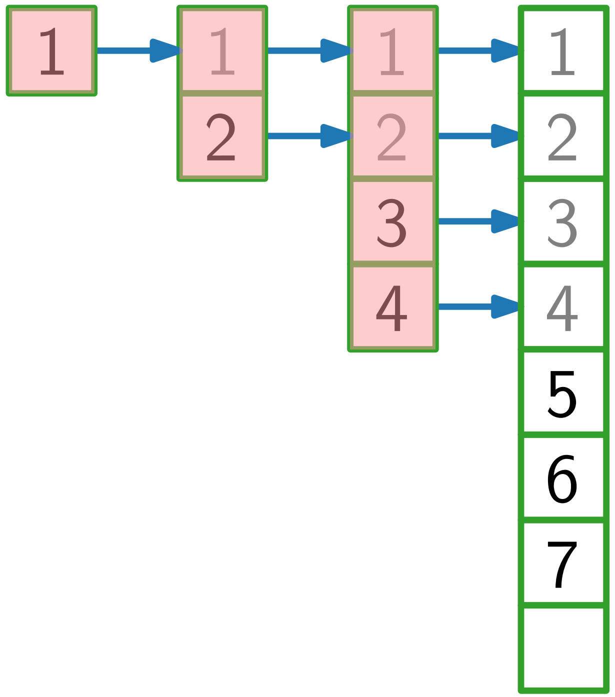
$$= n + 2 \cdot 2^{\log_2(n-1)} - 1$$

$$= n + 2(n-1) - 1 = 3n - 3$$

2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei
c_i = Kosten fürs i-te Einfügen.



i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren

Also betragen die Kosten für n Einfügeoperationen

$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq n + \frac{1 - 2^{\log_2(n-1)+1}}{1-2} = n + 2^{\log_2(n-1)+1} - 1$$

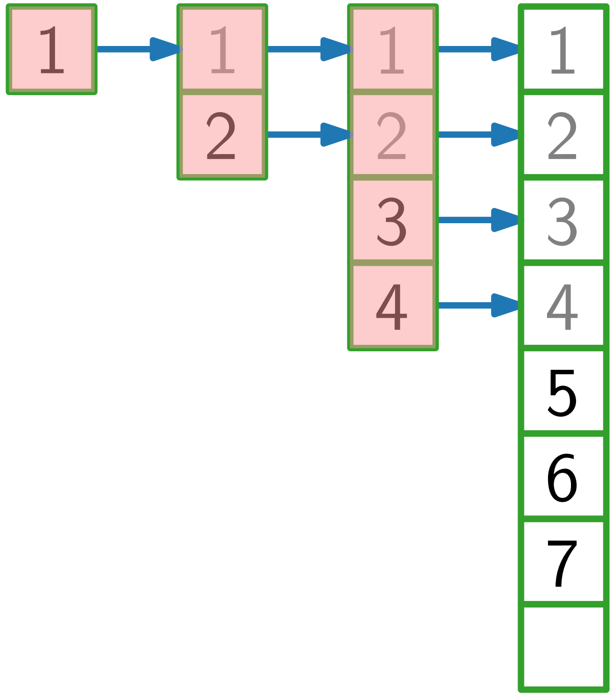
2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

$$= n + 2 \cdot 2^{\log_2(n-1)} - 1$$

$$= n + 2(n-1) - 1 = 3n - 3 \in \Theta(n)$$

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei
c_i = Kosten fürs i-te Einfügen.



i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren

Also betragen die Kosten für n Einfügeoperationen

$$\sum_{1}^{n} c_i = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq n + \frac{1 - 2^{\log_2(n-1)+1}}{1-2} = n + 2^{\log_2(n-1)+1} - 1$$

2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

$$= n + 2 \cdot 2^{\log_2(n-1)} - 1$$

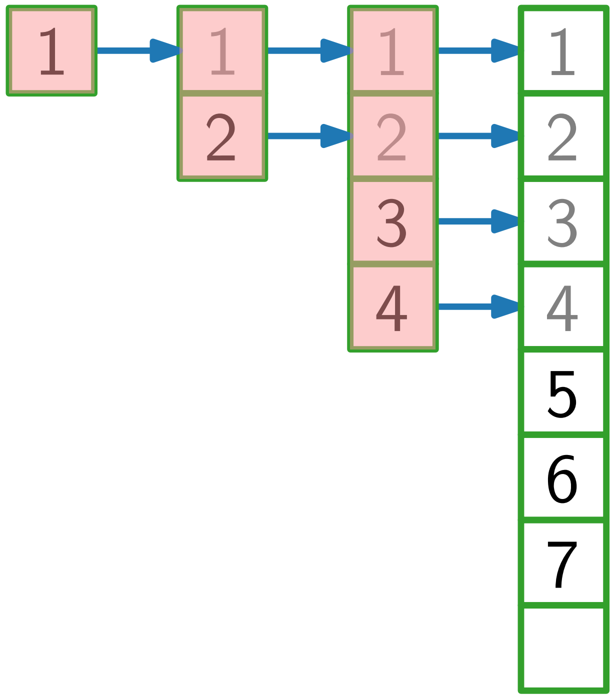
$$= n + 2(n-1) - 1 = 3n - 3 \in \Theta(n)$$

D.h. die durchschnittlichen (**amortisierten**) Kosten sind $\Theta(1)$.

Genauere Abschätzung: Aggregationsmethode

Für $i = 1, \ldots, n$ sei

$c_i =$ Kosten fürs i-te Einfügen.



i	1	2	3	4	5	6	7	8	9	
$size_i$	1	2	4	4	8	8	8	8	16	
c_i	1	1	1	1	1	1	1	1	1	← Einfügen
		1	2		4				8	← Kopieren

Also betragen die Kosten für n Einfügeoperationen

$$\boxed{\sum_{1}^{n} c_i} = \boxed{n} + \sum_{j=0}^{\lfloor \log_2(n-1) \rfloor} 2^j \leq n + \frac{1 - 2^{\log_2(n-1)+1}}{1-2} = n + 2^{\log_2(n-1)+1} - 1$$

2) $\sum_{j=0}^{n} q^j = \frac{1-q^{n+1}}{1-q}$ **geometrische Reihe**

$$= n + 2 \cdot 2^{\log_2(n-1)} - 1$$

$$= n + 2(n-1) - 1 = 3n - 3 \in \Theta(n)$$

D.h. die durchschnittlichen (**amortisierten**) Kosten sind $\Theta(1)$.

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben –

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Bei amortisierter Analyse geht es jedoch um die durchschnittliche Laufzeit *im schlechtesten Fall* – nicht im Erwartungswert!

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Bei amortisierter Analyse geht es jedoch um die durchschnittliche Laufzeit *im schlechtesten Fall* – nicht im Erwartungswert!

Die **amortisierte** Laufzeit einer Methode ist f, wenn jede Sequenz von k Aufrufen der Methode Laufzeit $\leq k \cdot f$ hat (wenn k *groß genug* ist).

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Bei amortisierter Analyse geht es jedoch um die durchschnittliche Laufzeit *im schlechtesten Fall* – nicht im Erwartungswert!

Die **amortisierte** Laufzeit einer Methode ist f, wenn jede Sequenz von k Aufrufen der Methode Laufzeit $\leq k \cdot f$ hat (wenn k *groß genug* ist).

Wir betrachten 3 verschiedene Typen von amortisierter Analyse:

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Bei amortisierter Analyse geht es jedoch um die durchschnittliche Laufzeit *im schlechtesten Fall* – nicht im Erwartungswert!

Die **amortisierte** Laufzeit einer Methode ist f, wenn jede Sequenz von k Aufrufen der Methode Laufzeit $\leq k \cdot f$ hat (wenn k *groß genug* ist).

Wir betrachten 3 verschiedene Typen von amortisierter Analyse:

- Aggregationsmethode
- Buchhaltermethode
- Potentialmethode

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Bei amortisierter Analyse geht es jedoch um die durchschnittliche Laufzeit *im schlechtesten Fall* – nicht im Erwartungswert!

Die **amortisierte** Laufzeit einer Methode ist f, wenn jede Sequenz von k Aufrufen der Methode Laufzeit $\leq k \cdot f$ hat (wenn k *groß genug* ist).

Wir betrachten 3 verschiedene Typen von amortisierter Analyse:

- Aggregationsmethode ✓
- Buchhaltermethode
- Potentialmethode

Amortisierte Analyse...

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Bei amortisierter Analyse geht es jedoch um die durchschnittliche Laufzeit *im schlechtesten Fall* – nicht im Erwartungswert!

Die **amortisierte** Laufzeit einer Methode ist f, wenn jede Sequenz von k Aufrufen der Methode Laufzeit $\leq k \cdot f$ hat (wenn k *groß genug* ist).

Wir betrachten 3 verschiedene Typen von amortisierter Analyse:

- Aggregationsmethode ✓
- Buchhaltermethode
- Potentialmethode

Buchhaltermethode

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$,

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$, die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$, die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

  $\hat{c}_i > c_i \;\Rightarrow$

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$, die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

  $\hat{c}_i > c_i \;\Rightarrow$ Wir legen etwas beiseite. 

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$, die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

  $\hat{c}_i > c_i \;\Rightarrow$ Wir legen etwas beiseite. 

  $\hat{c}_i < c_i \;\Rightarrow$

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$, die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

  $\hat{c}_i > c_i \;\Rightarrow$ Wir legen etwas beiseite. 

  $\hat{c}_i < c_i \;\Rightarrow$ Wir bezahlen teure Operationen mit vorher Beiseitegelegtem. 

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$,
  die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

  $\hat{c}_i > c_i \quad \Rightarrow$ Wir legen etwas beiseite. 

  $\hat{c}_i < c_i \quad \Rightarrow$ Wir bezahlen teure Operationen mit vorher Beiseitegelegtem. 

- Damit's klappt: wir dürfen nie in die Miesen kommen –

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$,
  die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

  $\hat{c}_i > c_i \Rightarrow$ Wir legen etwas beiseite. 

  $\hat{c}_i < c_i \Rightarrow$ Wir bezahlen teure Operationen mit vorher Beiseitegelegtem. 

- Damit's klappt: wir dürfen nie in die Miesen kommen –

  Guthaben $\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i$ darf nicht negativ werden!

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$,
  die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

  $\hat{c}_i > c_i \;\Rightarrow$ Wir legen etwas beiseite. 

  $\hat{c}_i < c_i \;\Rightarrow$ Wir bezahlen teure Operationen mit vorher Beiseitegelegtem. 

- Damit's klappt: wir dürfen nie in die Miesen kommen –

  Guthaben $\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i$ darf nicht negativ werden!

  Dann gilt $\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$.

Buchhaltermethode

- Verbindet mit jeder Operation op_i **amortisierte** Kosten $\hat{c}_i$, die oft nicht mit den **tatsächlichen** Kosten c_i übereinstimmen.

  $\hat{c}_i > c_i \;\Rightarrow\;$ Wir legen etwas beiseite. 

  $\hat{c}_i < c_i \;\Rightarrow\;$ Wir bezahlen teure Operationen mit vorher Beiseitegelegtem. 

- Damit's klappt: wir dürfen nie in die Miesen kommen –

  Guthaben $\displaystyle\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i$ darf nicht negativ werden!

  Dann gilt $\displaystyle\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i.$

  D.h. **amortisierte** Kosten sind obere Schranke für **tatsächliche** Kosten!

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$:

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$  :

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$  :

-  fürs tatsächliche Einfügen und

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$  :

-  fürs tatsächliche Einfügen und
-  fürs Kopieren in die nächstgrößere Tabelle.

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.

Buchhaltermethode für dynamische Tabellen

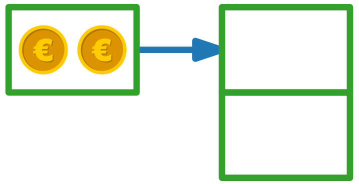
Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

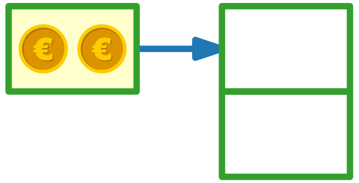
- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

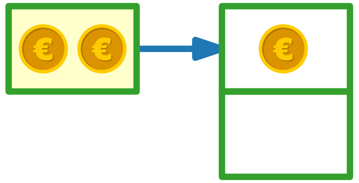
- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

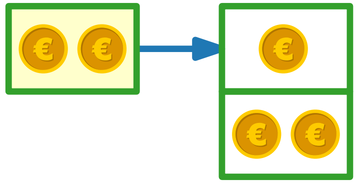
- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

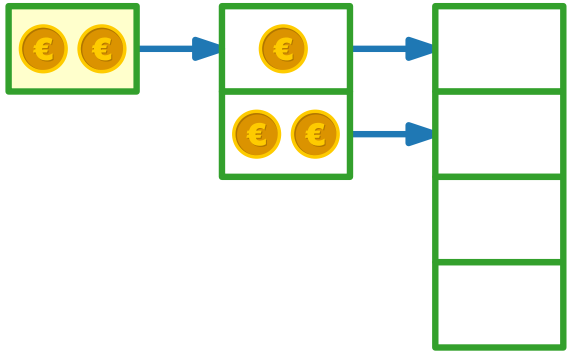
- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$  :

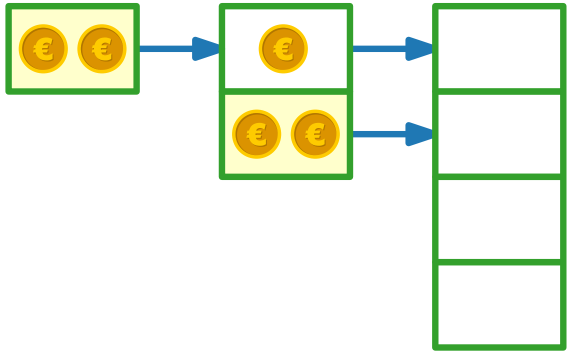
-  fürs tatsächliche Einfügen und
-  fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ € € € :

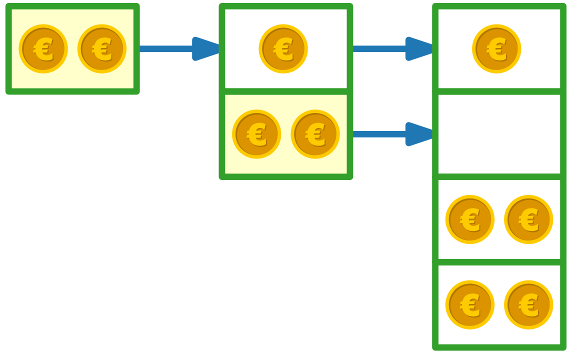
- € fürs tatsächliche Einfügen und
- € € fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

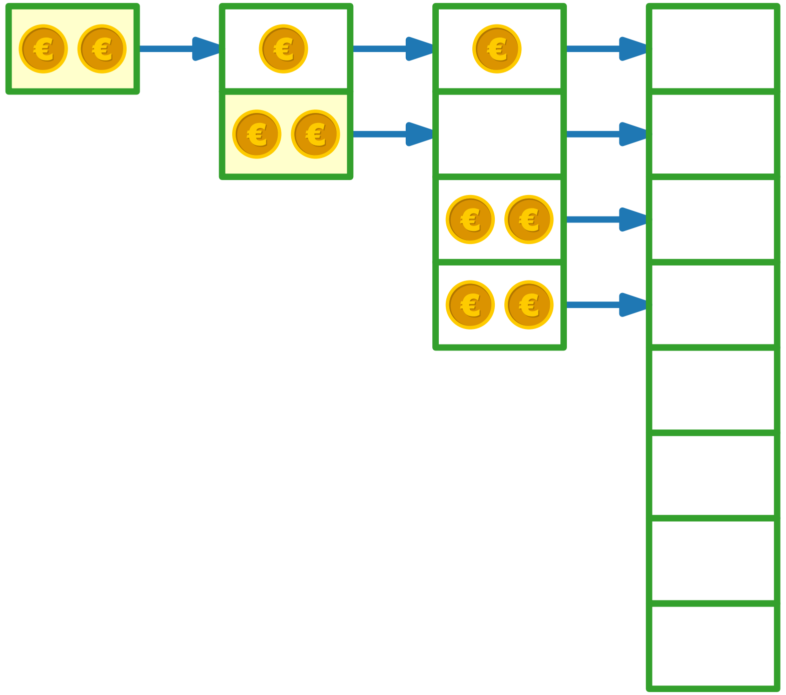
- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ € € € :

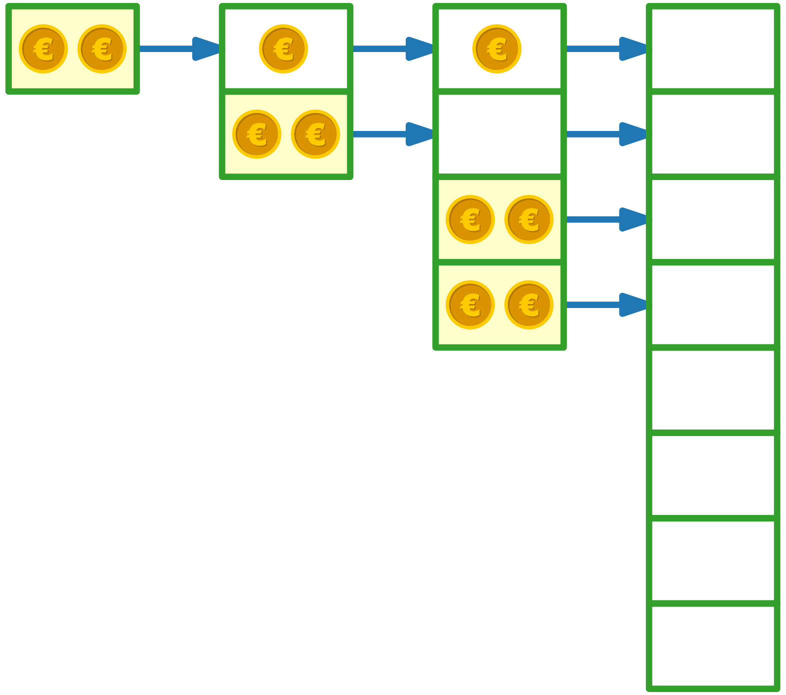
- € fürs tatsächliche Einfügen und
- € € fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

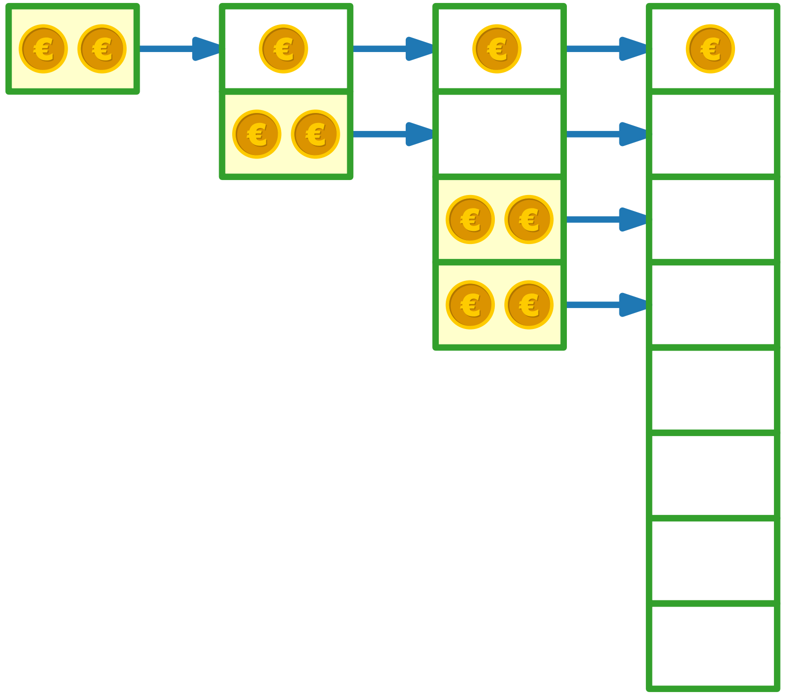
- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

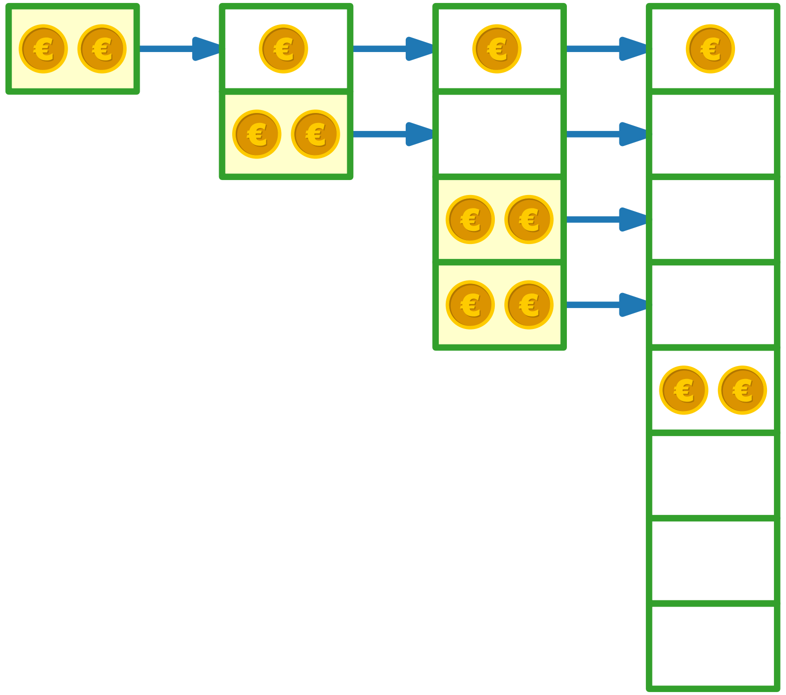
- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ € € € :

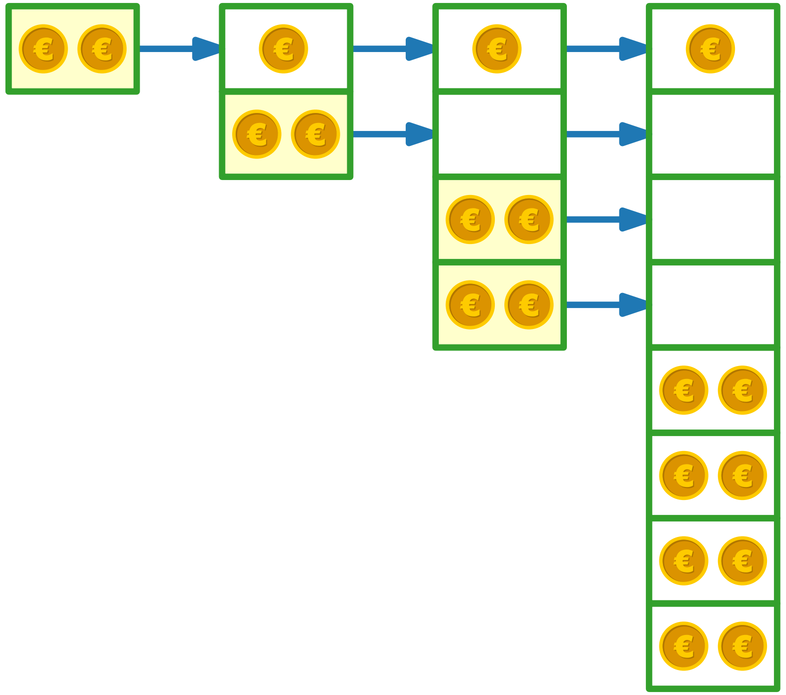
- € fürs tatsächliche Einfügen und
- € € fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

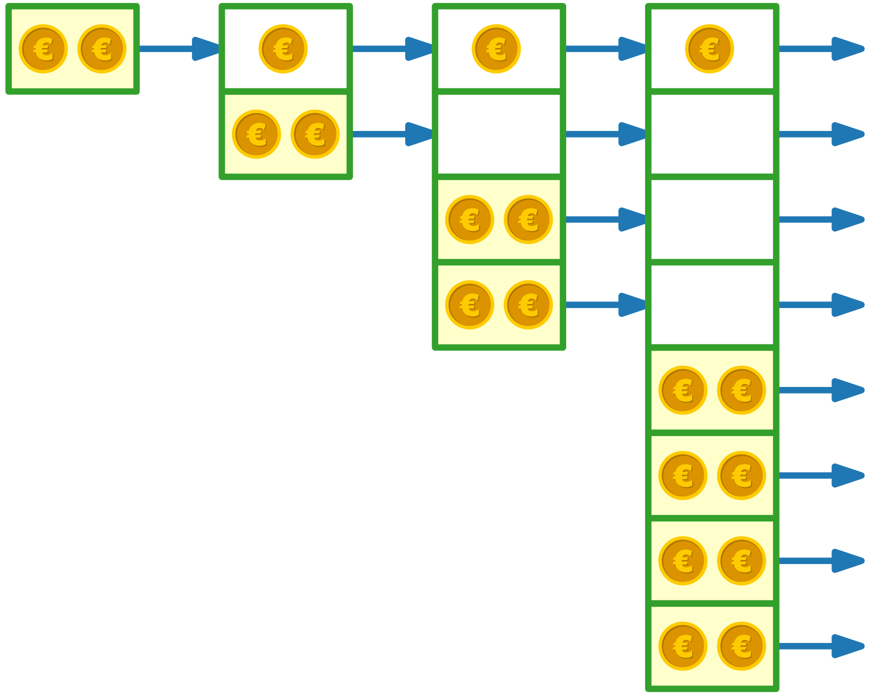
- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.

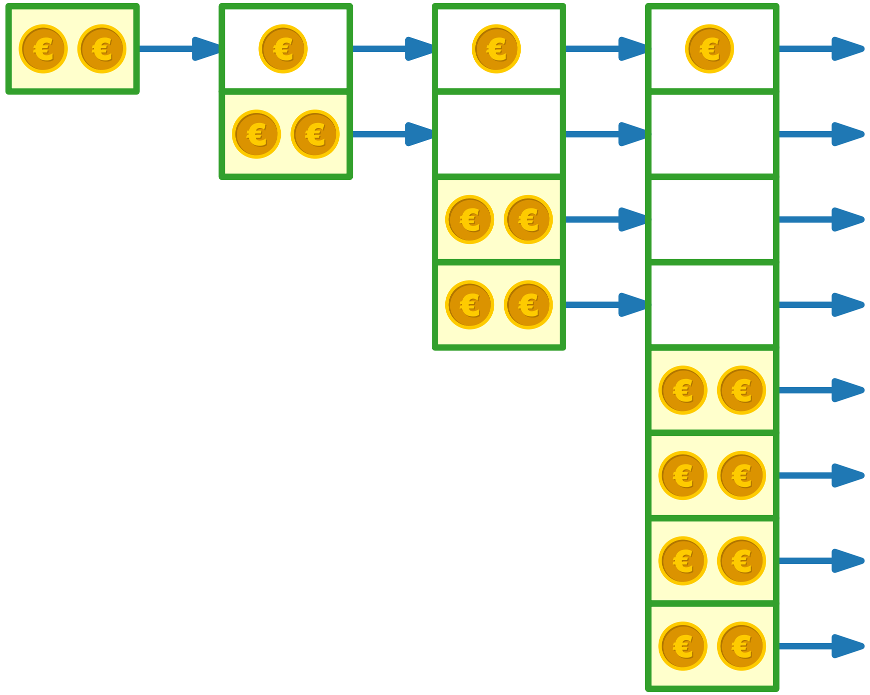


Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ € € € :

- € fürs tatsächliche Einfügen und
- € € fürs Kopieren in die nächstgrößere Tabelle.

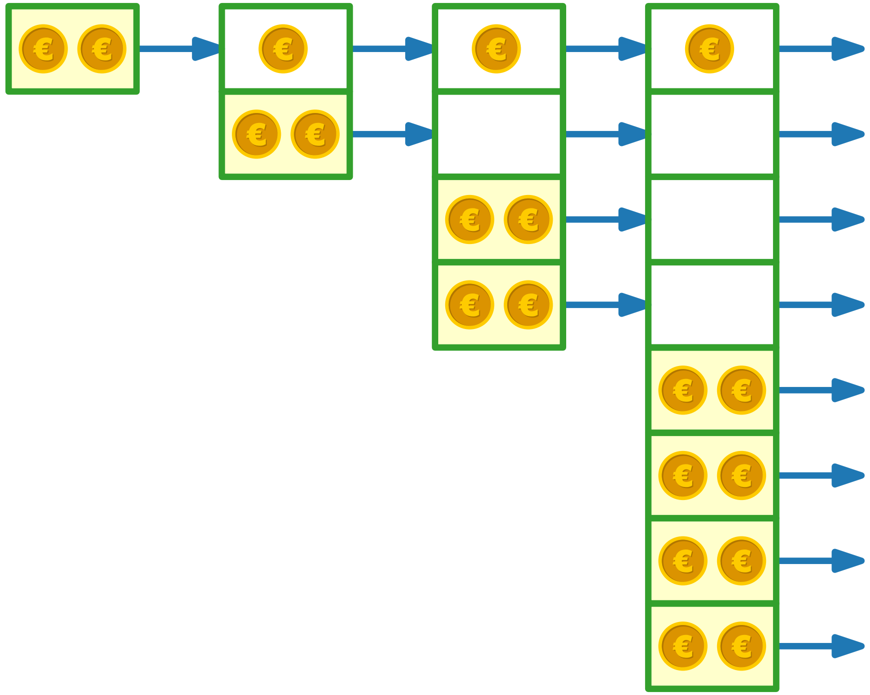
Wir verknüpfen die Teilguthaben mit konkreten Objekten der Datenstruktur.



Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



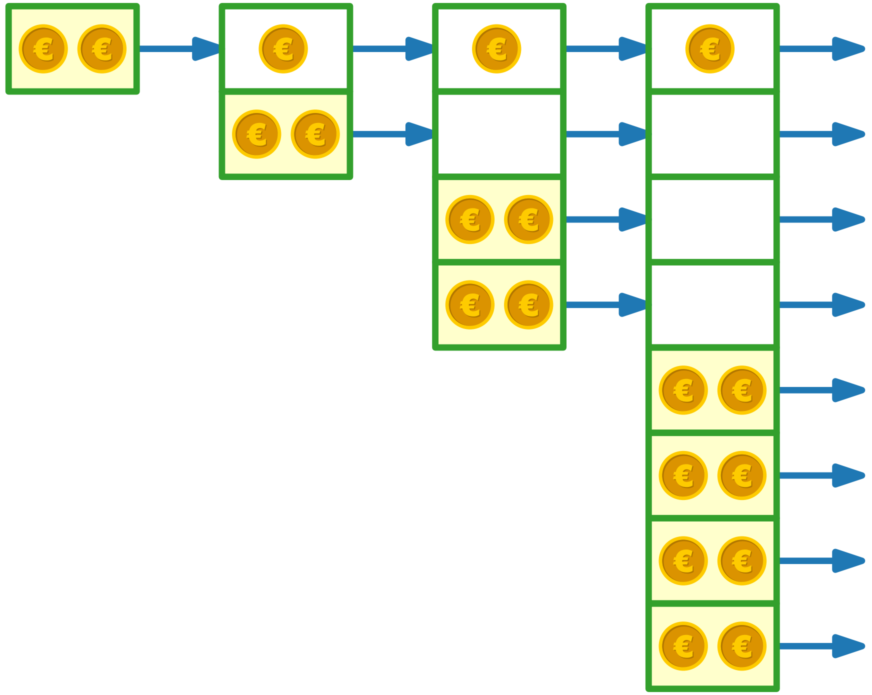
Wir verknüpfen die Teilguthaben mit konkreten Objekten der Datenstruktur.

Damit wird deutlich, dass die Datenstruktur nie Miese macht.

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Wir verknüpfen die Teilguthaben mit konkreten Objekten der Datenstruktur.

Damit wird deutlich, dass die Datenstruktur nie Miese macht.

$$\Downarrow$$

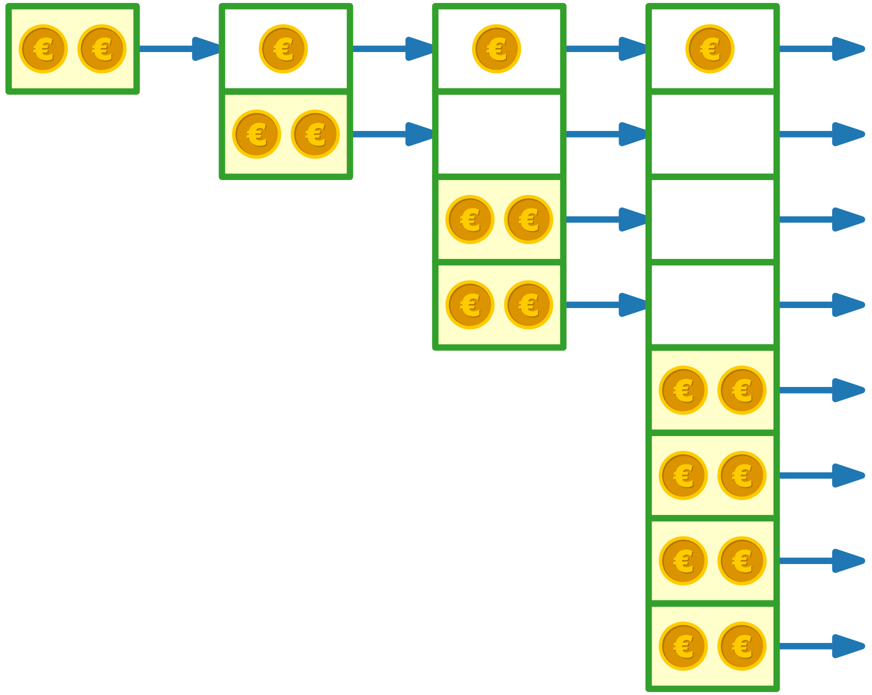
D.h. **amortisierte** Kosten sind obere Schranke für **tatsächliche** Kosten!

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i$$

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ € € € :

- € fürs tatsächliche Einfügen und
- € € fürs Kopieren in die nächstgrößere Tabelle.



Wir verknüpfen die Teilguthaben mit konkreten Objekten der Datenstruktur.

Damit wird deutlich, dass die Datenstruktur nie Miese macht.

$$\Downarrow$$

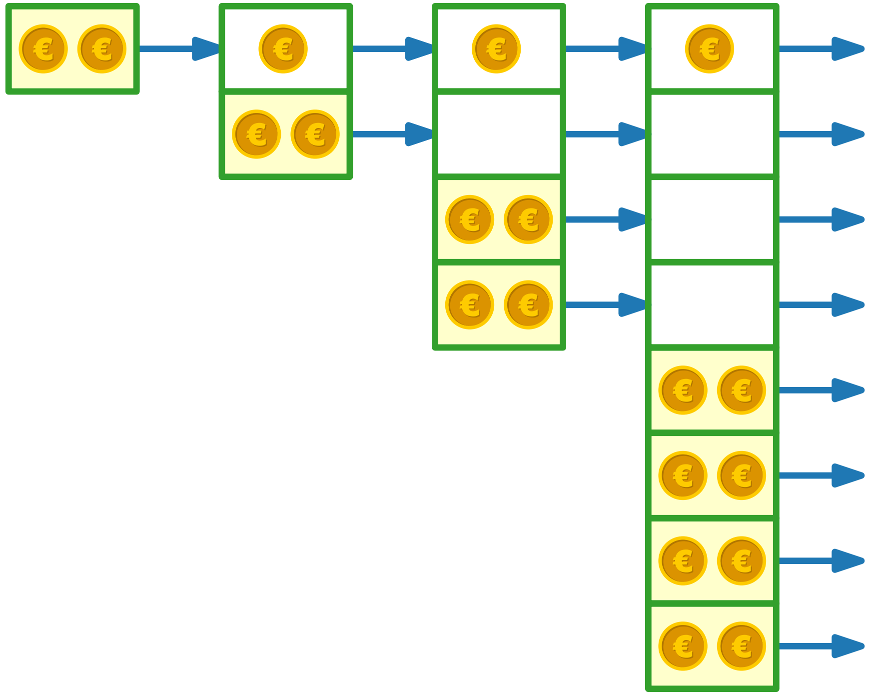
D.h. **amortisierte** Kosten sind obere Schranke für **tatsächliche** Kosten!

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i = 3n$$

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ € € € :

- € fürs tatsächliche Einfügen und
- € € fürs Kopieren in die nächstgrößere Tabelle.



Wir verknüpfen die Teilguthaben mit konkreten Objekten der Datenstruktur.

Damit wird deutlich, dass die Datenstruktur nie Miese macht.

$$\Downarrow$$

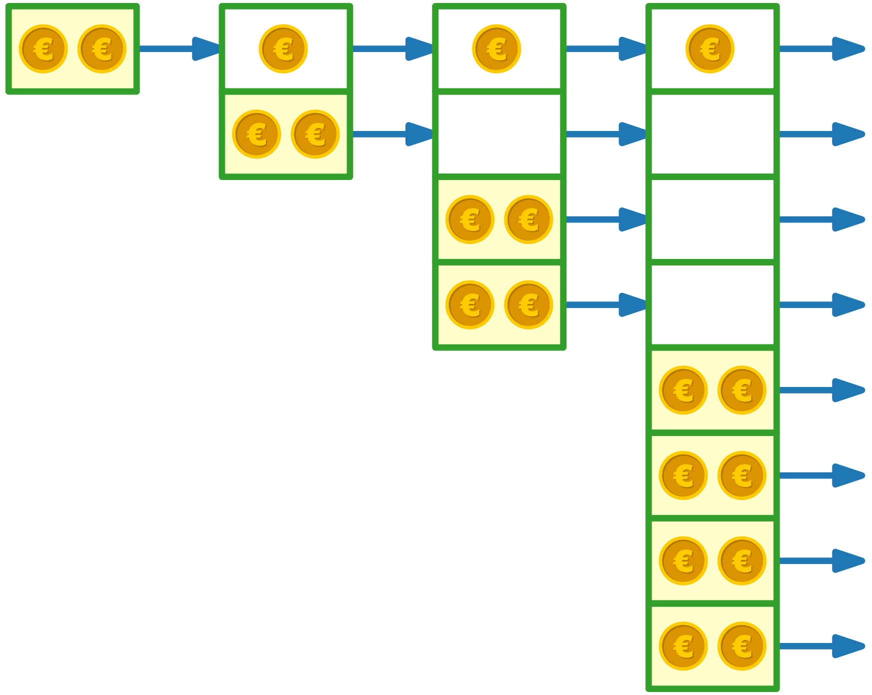
D.h. **amortisierte** Kosten sind obere Schranke für **tatsächliche** Kosten!

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i = 3n = \Theta(n)$$

Buchhaltermethode für dynamische Tabellen

Jede Einfügeoperation op_i bezahlt $\hat{c}_i =$ €€€ :

- € fürs tatsächliche Einfügen und
- €€ fürs Kopieren in die nächstgrößere Tabelle.



Wir verknüpfen die Teilguthaben mit konkreten Objekten der Datenstruktur.

Damit wird deutlich, dass die Datenstruktur nie Miese macht.

$$\Downarrow$$

D.h. **amortisierte** Kosten sind obere Schranke für **tatsächliche** Kosten!

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} \hat{c}_i = 3n = \Theta(n)$$

D.h. die **tatsächlichen** Kosten für n Einfügeoperationen betragen $\Theta(n)$.

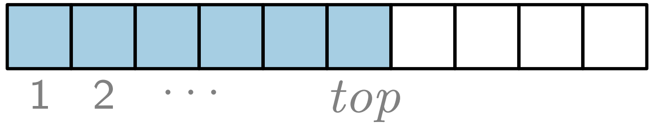
Buchhaltermethode für Stapel mit MULTIPOP

verwaltet sich ändernde Menge nach *LIFO-Prinzip*

Operation	Implementierung
Stack(int n)	
boolean EMPTY()	
PUSH(key k)	
key POP()	
key TOP()	

Buchhaltermethode für Stapel mit MULTIPOP

verwaltet sich ändernde Menge nach *LIFO-Prinzip*



key[ ] A

Operation	**Implementierung**
Stack(int n)	
boolean EMPTY()	
PUSH(key k)	
key POP()	
key TOP()	
key[ ] MULTIPOP(int k) neu!	

Buchhaltermethode für Stapel mit MULTIPOP

verwaltet sich ändernde Menge nach *LIFO-Prinzip*

Operation	Implementierung
Stack(int n)	
boolean EMPTY()	
PUSH(key k)	
key POP()	
key TOP()	
key[ ] MULTIPOP(int k) neu!	B = **new** key[k] **return** B

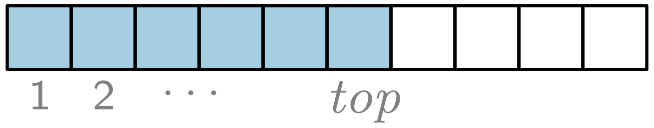
Buchhaltermethode für Stapel mit MULTIPOP

verwaltet sich ändernde Menge nach *LIFO-Prinzip*

Operation	**Implementierung**
Stack(int n)	
boolean EMPTY()	
PUSH(key k)	
key POP()	
key TOP()	
key[ ] MULTIPOP(int k) neu!	B = **new** key[k] **Aufgabe.** Vervollständigen Sie den Pseudocode. **return** B

Buchhaltermethode für Stapel mit $\textsc{Multipop}$

verwaltet sich ändernde Menge nach *LIFO-Prinzip*

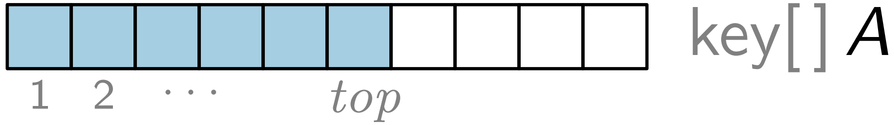


key[ ] A

Operation	Implementierung
Stack(int n) boolean $\textsc{Empty}()$ $\textsc{Push}$(key k) key $\textsc{Pop}()$ key $\textsc{Top}()$	
key[ ] $\textsc{Multipop}$(int k) neu!	B = **new** key[k] **while** **do** $B[k] = \textsc{Pop}()$ **return** B

Buchhaltermethode für Stapel mit MULTIPOP

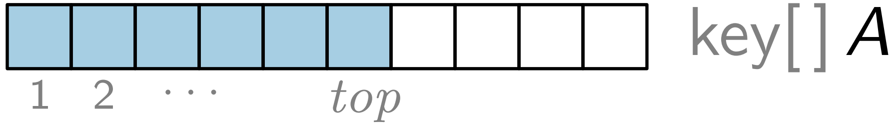
verwaltet sich ändernde Menge nach *LIFO-Prinzip*



Operation	**Implementierung**
Stack(int n)	
boolean EMPTY()	
PUSH(key k)	
key POP()	
key TOP()	
key[] MULTIPOP(int k) neu!	B = **new** key[k] **while** **do** $B[k]$ = POP() $k = k - 1$ **return** B

Buchhaltermethode für Stapel mit MULTIPOP

verwaltet sich ändernde Menge nach *LIFO-Prinzip*

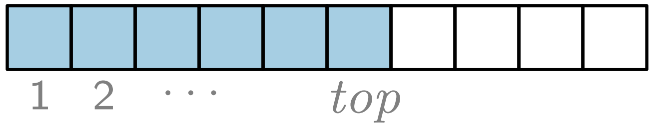


Operation	Implementierung
Stack(int n)	
boolean EMPTY()	
PUSH(key k)	
key POP()	
key TOP()	

key[ ] MULTIPOP(int k) neu!	B = **new** key[k]
	while $k > 0$ **do**
	    $B[k] = $ POP()
	    $k = k - 1$
	return B

Buchhaltermethode für Stapel mit Multipop

verwaltet sich ändernde Menge nach *LIFO-Prinzip*



key[ ] A

Operation	Implementierung
Stack(int n)	
boolean Empty()	
Push(key k)	
key Pop()	
key Top()	

key[ ] Multipop(int k) neu!	B = **new** key[k]
	while $k > 0$ **and not** Empty() **do**
	    $B[k]$ = Pop()
	    $k = k - 1$
	return B

Buchhaltermethode für Stapel mit Multipop

Betrachte Folge von Push-, Pop- und Multipop-Operationen.

Buchhaltermethode für Stapel mit MULTIPOP

Betrachte Folge von PUSH-, POP- und MULTIPOP-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
PUSH		
POP		
MULTIPOP(k_i)		

Buchhaltermethode für Stapel mit Multipop

Betrachte Folge von Push-, Pop- und Multipop-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
Push		
Pop		
Multipop(k_i)		

Buchhaltermethode für Stapel mit MULTIPOP

Betrachte Folge von PUSH-, POP- und MULTIPOP-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
PUSH		
POP		
MULTIPOP(k_i)		

Buchhaltermethode für Stapel mit Multipop

Betrachte Folge von Push-, Pop- und Multipop-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
Push		
Pop		
Multipop(k_i)	k_i	

Buchhaltermethode für Stapel mit MULTIPOP

Betrachte Folge von PUSH-, POP- und MULTIPOP-Operationen.

Operation i	tatsächliche Kosten c_i	amortisierte Kosten $\hat{c}_i$
PUSH		
POP		
MULTIPOP(k_i)	$\min\{k_i, size_i\}$	

Buchhaltermethode für Stapel mit Multipop

Betrachte Folge von Push-, Pop- und Multipop-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
Push	€	€ €
Pop	€	0
Multipop(k_i)	$\min\{k_i, size_i\}$	0

Buchhaltermethode für Stapel mit Multipop

Betrachte Folge von Push-, Pop- und Multipop-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
Push		
Pop		0
Multipop(k_i)	$\min\{k_i, size_i\}$	0

Geht das gut?

Buchhaltermethode für Stapel mit Multipop

Betrachte Folge von Push-, Pop- und Multipop-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
Push	€	€ €
Pop	€	0
Multipop(k_i)	$\min\{k_i, size_i\}$	0

Geht das gut?

Zeige: **Amortisierte** Kosten „bezahlen“ immer für die **tatsächlichen**!

Buchhaltermethode für Stapel mit MULTIPOP

Betrachte Folge von PUSH-, POP- und MULTIPOP-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
PUSH	€	€ €
POP	€	0
MULTIPOP(k_i)	$\min\{k_i, size_i\}$	0

Geht das gut?

Zeige: **Amortisierte** Kosten „bezahlen“ immer für die **tatsächlichen**!

- Jede PUSH-Operation legt ein Buch auf den Stapel.

Buchhaltermethode für Stapel mit MULTIPOP

Betrachte Folge von PUSH-, POP- und MULTIPOP-Operationen.

Operation i	tatsächliche Kosten c_i	amortisierte Kosten $\hat{c}_i$
PUSH	€	€ €
POP	€	0
MULTIPOP(k_i)	$\min\{k_i, size_i\}$	0

Geht das gut?

Zeige: **Amortisierte** Kosten „bezahlen“ immer für die **tatsächlichen**!

- Jede PUSH-Operation legt ein Buch auf den Stapel.
  Dafür bezahlt sie € und legt noch € in das Buch.

Buchhaltermethode für Stapel mit Multipop

Betrachte Folge von Push-, Pop- und Multipop-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
Push	€	€ €
Pop	€	0
Multipop(k_i)	$\min\{k_i, size_i\}$	0

Geht das gut?

Zeige: **Amortisierte** Kosten „bezahlen“ immer für die **tatsächlichen**!

- Jede Push-Operation legt ein Buch auf den Stapel.
  Dafür bezahlt sie € und legt noch € in das Buch.
- Jede (Multi-)Pop-Operation wird mit den Euros in den Büchern, die sie wegnimmt, komplett bezahlt.

Buchhaltermethode für Stapel mit MULTIPOP

Betrachte Folge von PUSH-, POP- und MULTIPOP-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
PUSH	€	€ €
POP	€	0
MULTIPOP(k_i)	$\min\{k_i, size_i\}$	0

Geht das gut? – Ja!

Zeige: **Amortisierte** Kosten „bezahlen“ immer für die **tatsächlichen**!

- Jede PUSH-Operation legt ein Buch auf den Stapel.

  Dafür bezahlt sie € und legt noch € in das Buch.

- Jede (MULTI-)POP-Operation wird mit den Euros in den Büchern, die sie wegnimmt, komplett bezahlt.

Buchhaltermethode für Stapel mit MULTIPOP

Betrachte Folge von PUSH-, POP- und MULTIPOP-Operationen.

Operation i	**tatsächliche Kosten** c_i	**amortisierte Kosten** $\hat{c}_i$
PUSH	€	€ €
POP	€	0
MULTIPOP(k_i)	$\min\{k_i, size_i\}$	0

Geht das gut? – Ja! D.h. Folge von n Operationen dauert $\Theta(n)$ Zeit.

Zeige: **Amortisierte** Kosten „bezahlen“ immer für die **tatsächlichen**!

- Jede PUSH-Operation legt ein Buch auf den Stapel.
  Dafür bezahlt sie € und legt noch € in das Buch.
- Jede (MULTI-)POP-Operation wird mit den Euros in den Büchern, die sie wegnimmt, komplett bezahlt.

Amortisierte Analyse

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben –
auch wenn einzelne Operationen in der Folge teuer sind!

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Bei amortisierter Analyse geht es jedoch um die durchschnittliche Laufzeit *im schlechtesten Fall* – nicht im Erwartungswert!

Die **amortisierte** Laufzeit einer Methode ist f, wenn jede Sequenz von k Aufrufen der Methode Laufzeit $\leq k \cdot f$ hat (wenn k *groß genug* ist).

Wir betrachten 3 verschiedene Typen von amortisierter Analyse:

- Aggregationsmethode ✓
- Buchhaltermethode ✓
- Potentialmethode

Amortisierte Analyse

...bedeutet zu zeigen, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben –
auch wenn einzelne Operationen in der Folge teuer sind!

Auch *randomisierte Analyse* kann man als Durchschnittsbildung (über alle mögl. Ergebnisse, gewichtet nach Wahrscheinlichkeit) betrachten.

Bei amortisierter Analyse geht es jedoch um die durchschnittliche Laufzeit *im schlechtesten Fall* – nicht im Erwartungswert!

Die **amortisierte** Laufzeit einer Methode ist f, wenn jede Sequenz von k Aufrufen der Methode Laufzeit $\leq k \cdot f$ hat (wenn k *groß genug* ist).

Wir betrachten 3 verschiedene Typen von amortisierter Analyse:

- Aggregationsmethode ✓
- Buchhaltermethode ✓
- Potentialmethode

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \longrightarrow D_1 \longrightarrow \cdots \longrightarrow D_n$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi \colon D_i \to \mathbb{R}$.

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)
als physikalische Größe,
die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi \colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

Def. $\hat{c}_i = c_i + \Delta\Phi(D_i)$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

Def. $\underbrace{\hat{c}_i}_{\text{amortisierte Kosten}} = \underbrace{c_i}_{\text{echte Kosten}} + \Delta\Phi(D_i)$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \dots, n$.

amortisierte Kosten

echte Kosten

Def. $\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \dots, n$.

Def. $\underbrace{\hat{c}_i}_{\text{amortisierte Kosten}} = \underbrace{c_i}_{\text{echte Kosten}} + \Delta\Phi(D_i)$, wobei $\underbrace{\Delta\Phi(D_i)}_{\text{Potentialdifferenz}} = \Phi(D_i) - \Phi(D_{i-1})$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)
als physikalische Größe,
die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

Def. $\underbrace{\hat{c}_i}_{\text{amortisierte Kosten}} = \underbrace{c_i}_{\text{echte Kosten}} + \underbrace{\Delta\Phi(D_i)}_{\text{Potentialdifferenz}}$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^{n} \hat{c}_i =$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

Def. $\underbrace{\hat{c}_i}_{\text{amortisierte Kosten}} = \underbrace{c_i}_{\text{echte Kosten}} + \underbrace{\Delta\Phi(D_i)}_{\text{Potentialdifferenz}}$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n}$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

amortisierte Kosten

echte Kosten

Potentialdifferenz

Def. $\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} \left(c_i + \Phi(D_i) - \Phi(D_{i-1})\right)$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)

als physikalische Größe,

die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

Def. $\underbrace{\hat{c}_i}_{\text{amortisierte Kosten}} = \underbrace{c_i}_{\text{echte Kosten}} + \underbrace{\Delta\Phi(D_i)}_{\text{Potentialdifferenz}}$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} \left(c_i + \Phi(D_i) - \Phi(D_{i-1})\right)$ Teleskopsumme

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)
als physikalische Größe,
die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

Def. $\overset{\text{amortisierte Kosten}}{\hat{c}_i} = \overset{\text{echte Kosten}}{c_i} + \Delta\Phi(D_i)$, wobei $\overset{\text{Potentialdifferenz}}{\Delta\Phi(D_i)} = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} \left(c_i + \Phi(D_i) - \Phi(D_{i-1})\right)$ Teleskopsumme

$= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)
als physikalische Größe,
die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \ldots, n$.

amortisierte Kosten echte Kosten Potentialdifferenz

Def. $\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} \big(c_i + \Phi(D_i) - \Phi(D_{i-1})\big)$ Teleskopsumme

$$= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0)$$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)
als physikalische Größe,
die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi \colon D_i \to \mathbb{R}$. O.B.d.A. $\Phi(D_0) = 0$

Ziel. Bank macht keine Miesen.

Also fordern wir $\Phi(D_i) \geq 0$ für $i = 1, \dots, n$.

amortisierte Kosten

echte Kosten

Potentialdifferenz

Def. $\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n \left(c_i + \Phi(D_i) - \Phi(D_{i-1})\right)$ Teleskopsumme

$$= \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0)$$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)
als physikalische Größe,
die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\boxed{\Phi(D_0) = 0}$

Ziel. Bank macht keine Miesen.

Also fordern wir $\boxed{\Phi(D_i) \geq 0}$ für $i = 1, \ldots, n$.

amortisierte Kosten

echte Kosten

Potentialdifferenz

Def. $\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n \left(c_i + \Phi(D_i) - \Phi(D_{i-1})\right)$ Teleskopsumme

$= \sum_{i=1}^n c_i + \boxed{\Phi(D_n)} - \boxed{\Phi(D_0)} \geq \sum_{i=1}^n c_i$

Potentialmethode

Idee. Betrachte Bankguthaben (siehe Buchhaltermethode)
als physikalische Größe,
die den augenblicklichen Zustand der DS beschreibt.

Datenstruktur $D_0 \xrightarrow{op_1} D_1 \xrightarrow{op_2} \cdots \xrightarrow{op_n} D_n$

Wähle Potential $\Phi\colon D_i \to \mathbb{R}$. O.B.d.A. $\boxed{\Phi(D_0) = 0}$

Ziel. Bank macht keine Miesen.

Also fordern wir $\boxed{\Phi(D_i) \geq 0}$ für $i = 1, \dots, n$.

amortisierte Kosten

echte Kosten

Potentialdifferenz

Def. $\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

Folge: $\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} \big(c_i + \Phi(D_i) - \Phi(D_{i-1})\big)$ Teleskopsumme

$$= \sum_{i=1}^{n} c_i + \boxed{\Phi(D_n)} - \boxed{\Phi(D_0)} \geq \sum_{i=1}^{n} c_i$$

D.h. **amortisierte** Kosten „bezahlen“ für **tatsächliche** Kosten.

Potentialmethode für Stapel mit Multipop

To do: Definiere Potentialfunktion –

Potentialmethode für Stapel mit Multipop

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) =$

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \ldots, \Phi(D_n) \geq 0$.

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = \mathit{size}_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe:

Was sind die amort. Kosten?

Potentialmethode für Stapel mit $\textsc{Multipop}$

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine $\textsc{Push}$-Operation ist:

Was sind die amort. Kosten?

Potentialmethode für Stapel mit $\textsc{Multipop}$

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \ldots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine $\textsc{Push}$-Operation ist:
$\Rightarrow \Delta\Phi(D_i) = 1$

Was sind die amort. Kosten?

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine PUSH-Operation ist:
$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i$

Was sind die amort. Kosten?

Potentialmethode für Stapel mit $\textsc{Multipop}$

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine $\textsc{Push}$-Operation ist:
$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Was sind die amort. Kosten?

Potentialmethode für Stapel mit $\textsc{Multipop}$

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \ldots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine $\textsc{Push}$-Operation ist:
$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Was sind die amort. Kosten?

Falls die i-te Operation eine $(\textsc{Multi-})\textsc{Pop}$-Operation ist:

Potentialmethode für Stapel mit $\textsc{Multipop}$

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die *i*-te Operation eine $\textsc{Push}$-Operation ist:
$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Falls die *i*-te Operation eine $(\textsc{Multi-})\textsc{Pop}$-Operation ist:
$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, size_i\}$

Was sind die amort. Kosten?

Potentialmethode für Stapel mit $\textsc{Multipop}$

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = \mathit{size}_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine $\textsc{Push}$-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Falls die i-te Operation eine $(\textsc{Multi-})\textsc{Pop}$-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, \mathit{size}_i\}$

$c_i = \min\{k_i, \mathit{size}_i\}$

Was sind die amort. Kosten?

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine PUSH-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Falls die i-te Operation eine (MULTI-)POP-Operation ist:

$$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, size_i\}$$

$$c_i = \min\{k_i, size_i\}$$

Was sind die amort. Kosten?

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion – in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = \mathit{size}_i$, also aktuelle Stapelgröße.

$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine PUSH-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Was sind die amort. Kosten?

Falls die i-te Operation eine (MULTI-)POP-Operation ist:

$$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, \mathit{size}_i\}$$

$$c_i = \min\{k_i, \mathit{size}_i\}$$

$$\hat{c}_i = c_i + \Delta\Phi D_i$$

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine PUSH-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Was sind die amort. Kosten?

Falls die i-te Operation eine (MULTI-)POP-Operation ist:

$$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, size_i\}$$

$$c_i = \min\{k_i, size_i\}$$

$$\hat{c}_i = c_i + \Delta\Phi D_i = 0$$

Potentialmethode für Stapel mit Multipop

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = \mathit{size}_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine Push-Operation ist:

Was sind die amort. Kosten?

$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Falls die i-te Operation eine (Multi-)Pop-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, \mathit{size}_i\}$

$$c_i = \min\{k_i, \mathit{size}_i\}$$

$$\hat{c}_i = c_i + \Delta\Phi D_i = 0 \quad \text{(bei Pop } k_i = 1\text{)}$$

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = size_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine PUSH-Operation ist:

Was sind die amort. Kosten?

$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Falls die i-te Operation eine (MULTI-)POP-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, size_i\}$

$$c_i = \min\{k_i, size_i\}$$

$$\hat{c}_i = c_i + \Delta\Phi D_i = 0$$ (bei POP $k_i = 1$)

Also:

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = \mathit{size}_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine PUSH-Operation ist:

Was sind die amort. Kosten?

$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = 2$

Falls die i-te Operation eine (MULTI-)POP-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, \mathit{size}_i\}$

$c_i = \min\{k_i, \mathit{size}_i\}$

$\hat{c}_i = c_i + \Delta\Phi D_i = 0$ (bei POP $k_i = 1$)

Also: **Amortisierte** Kosten pro Operation $\Theta(1)$.

Potentialmethode für Stapel mit MULTIPOP

To do: Definiere Potentialfunktion –
in Abhängigkeit vom aktuellen Zustand des Stapels!

Idee: Nimm $\Phi(D_i) = \mathit{size}_i$, also aktuelle Stapelgröße.
$\Rightarrow \Phi(D_0) = 0$ und $\Phi(D_1), \dots, \Phi(D_n) \geq 0$. ✓

Prüfe: Falls die i-te Operation eine PUSH-Operation ist:

Was sind die amort. Kosten?

$\Rightarrow \Delta\Phi(D_i) = 1$ und $\hat{c}_i = c_i + \Delta\Phi D_i = 1 + 1 = \boxed{2}$

Falls die i-te Operation eine (MULTI-)POP-Operation ist:

$\Rightarrow \Delta\Phi(D_i) = -\min\{k_i, \mathit{size}_i\}$

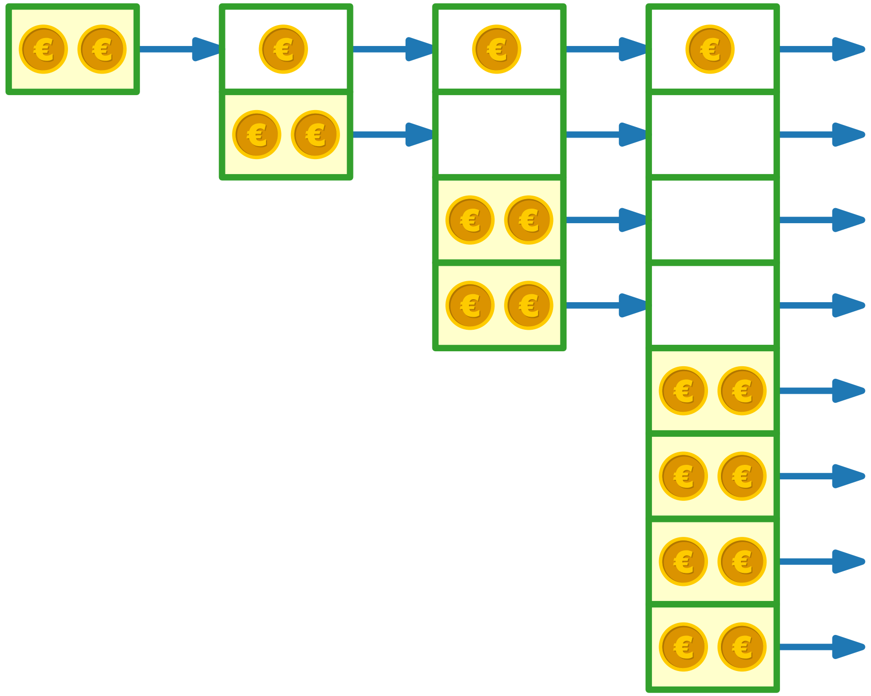
$c_i = \min\{k_i, \mathit{size}_i\}$

$\hat{c}_i = c_i + \Delta\Phi D_i = \boxed{0}$ (bei POP $k_i = 1$)

Also: **Amortisierte** Kosten pro Operation $\Theta(\boxed{1})$.
$\Rightarrow$ **Tatsächliche** Kosten für n Operationen im worst case $\Theta(n)$.

Potentialmethode für dynamische Tabellen

Idee.

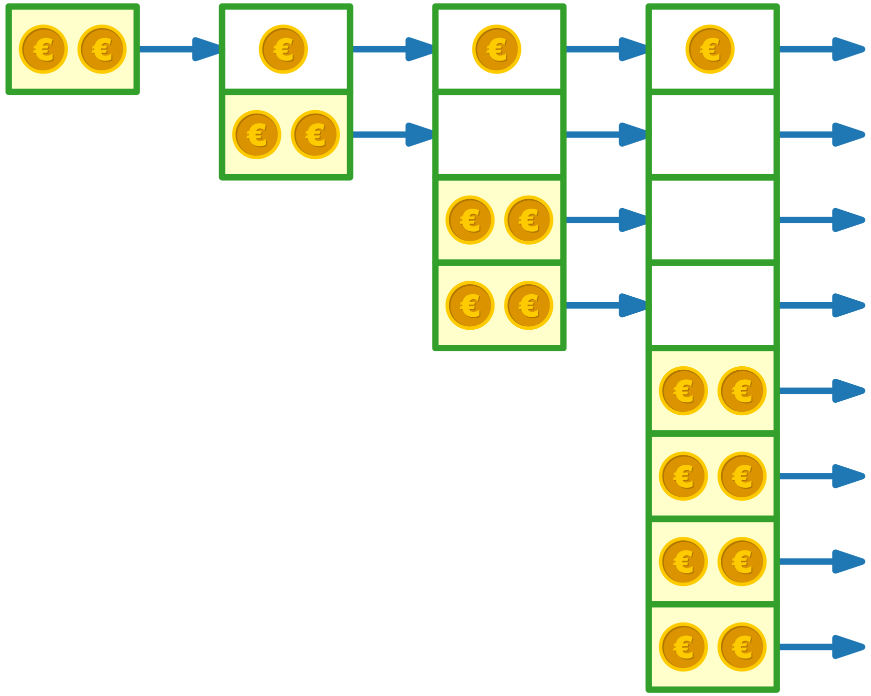


Potentialmethode für dynamische Tabellen

Idee.

$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



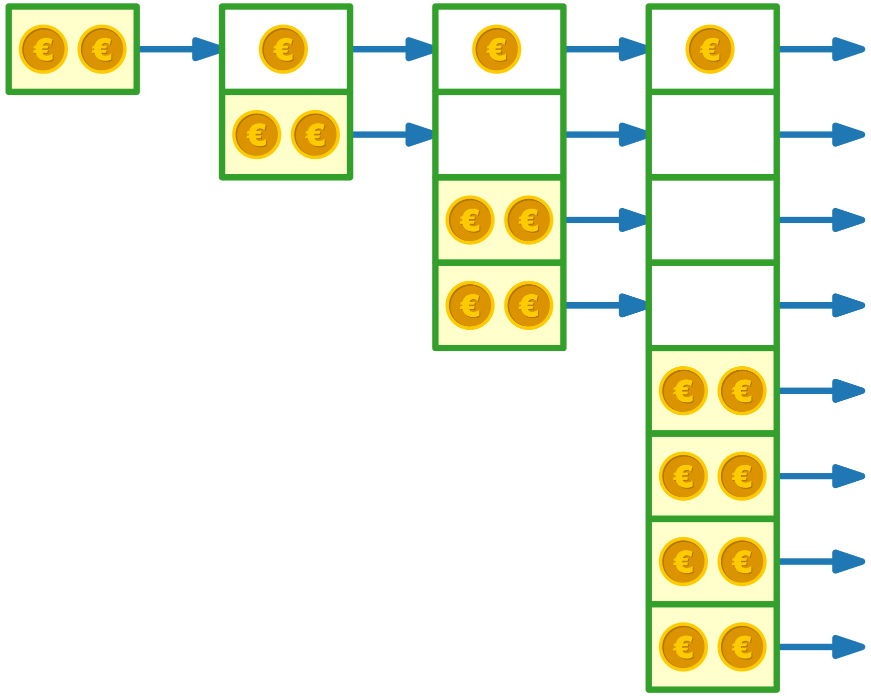
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$

$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$

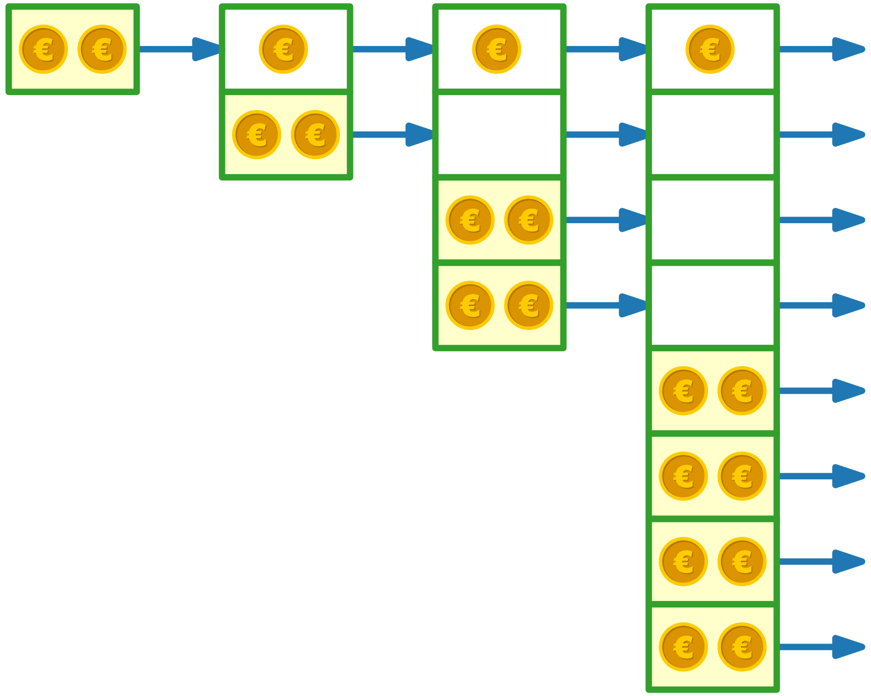


Potentialmethode für dynamische Tabellen

Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i - 1)$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



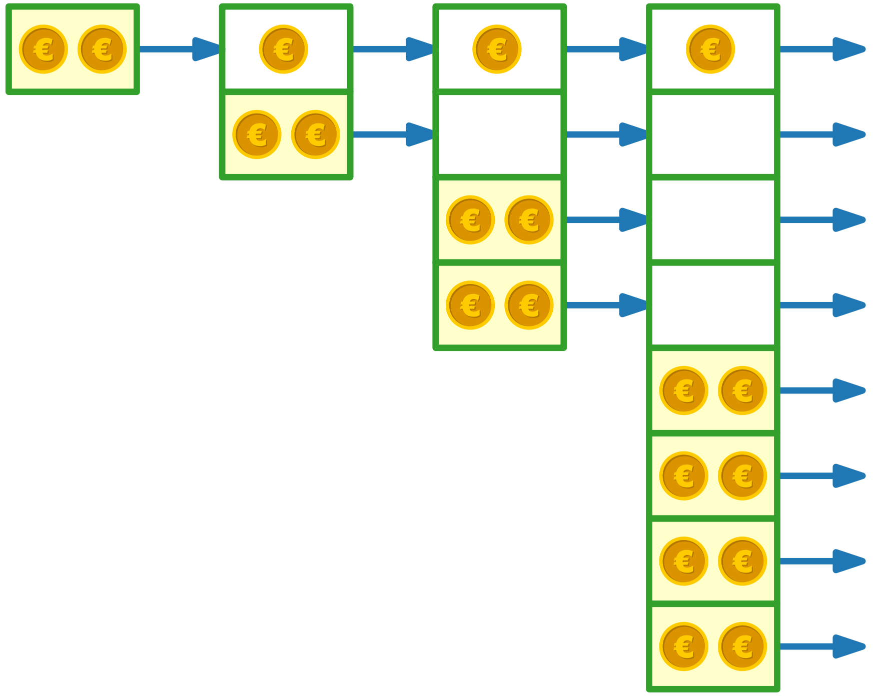
Potentialmethode für dynamische Tabellen

Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i - 1)$

$i - 1$ Elemente werden kopiert

$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



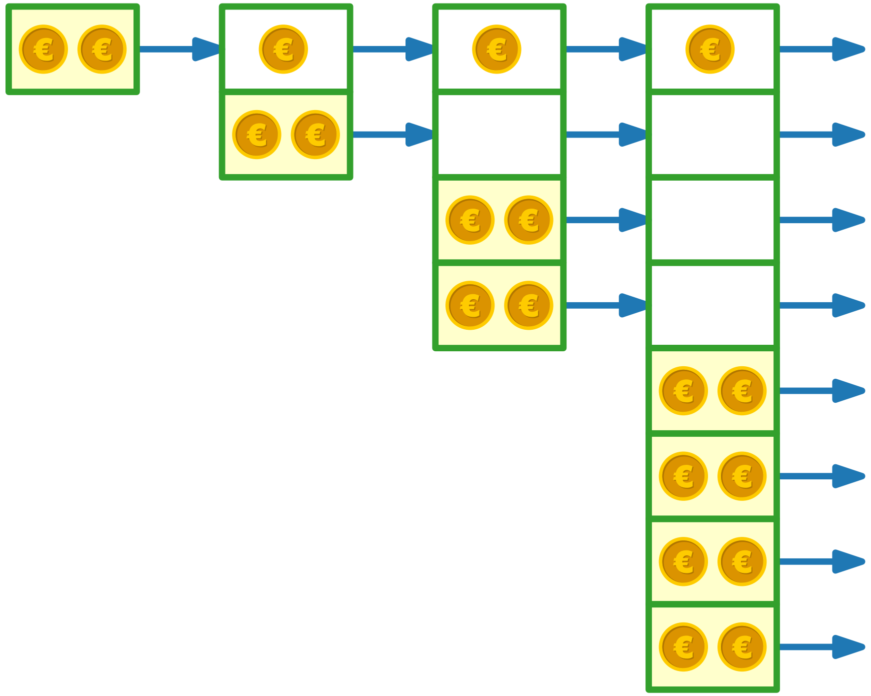
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$ — $i-1$ Elemente werden kopiert
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



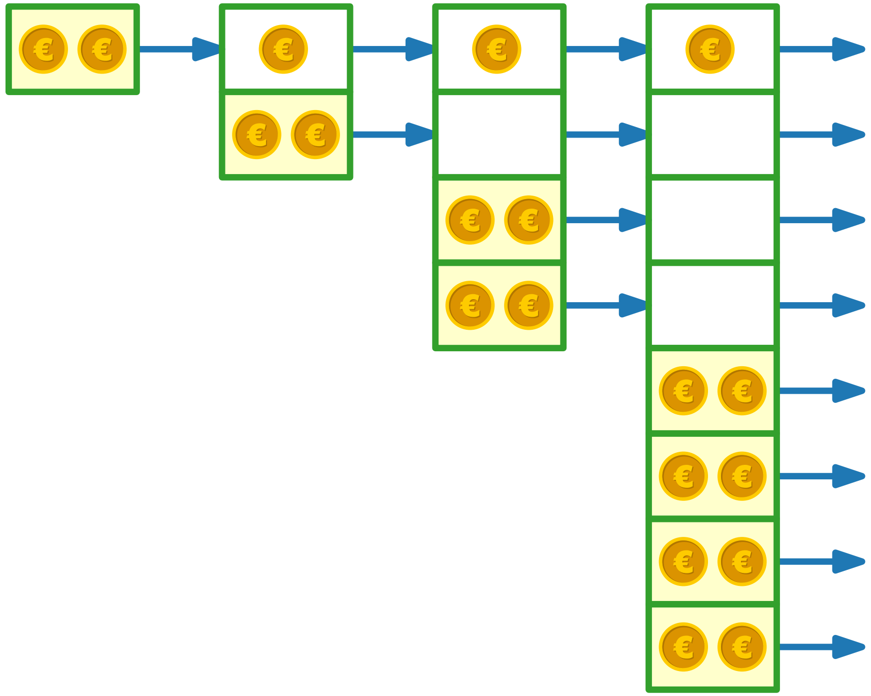
Potentialmethode für dynamische Tabellen

Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

  $i-1$ Elemente werden kopiert
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



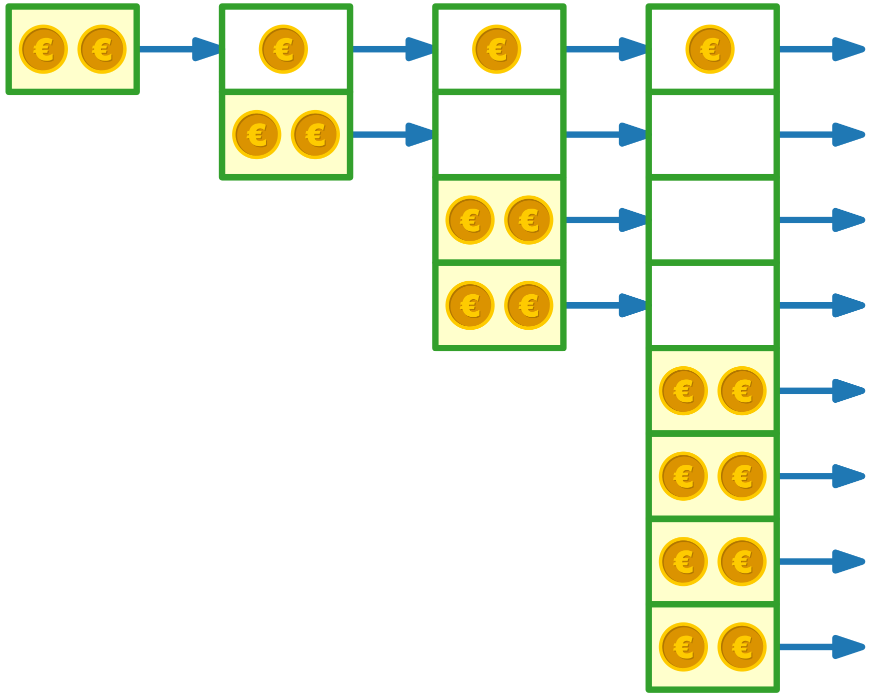
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$ — $i - 1$ Elemente werden kopiert
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i - 1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - table\text{-}size_i$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



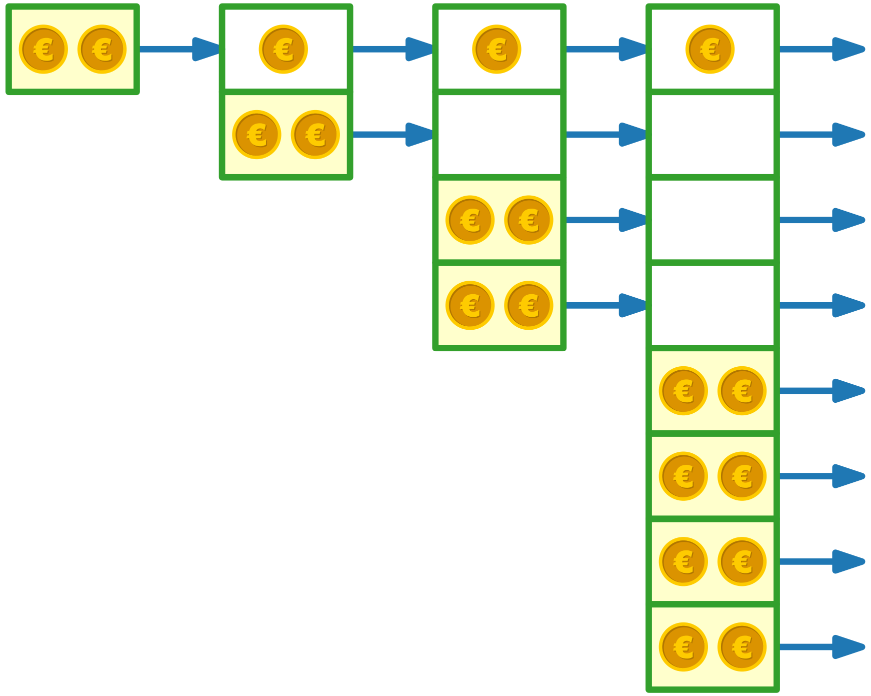
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$ — $i-1$ Elemente werden kopiert
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



Potentialmethode für dynamische Tabellen

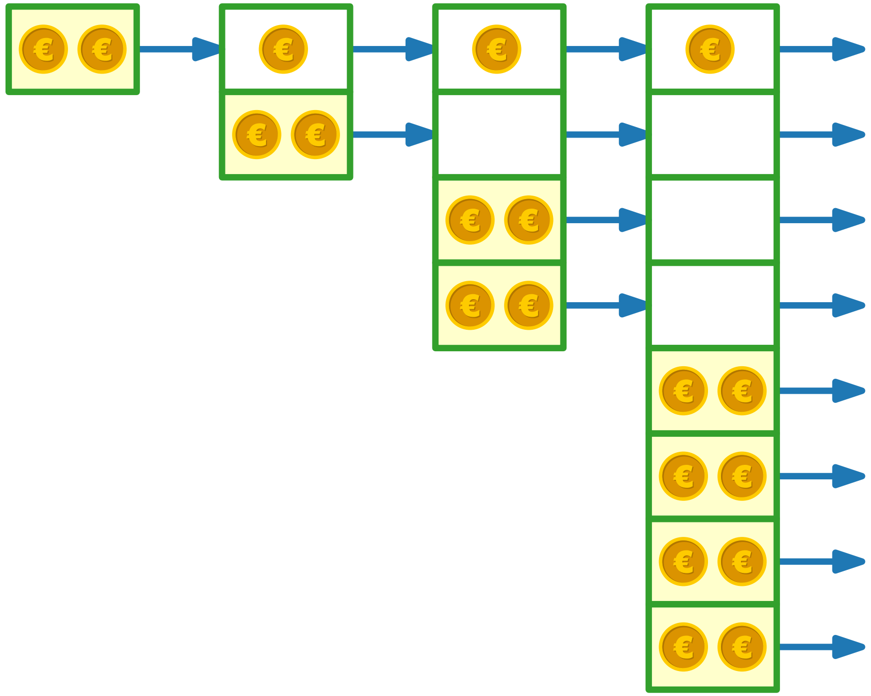
Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

  $i-1$ Elemente werden kopiert
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\textstyle\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^n c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$

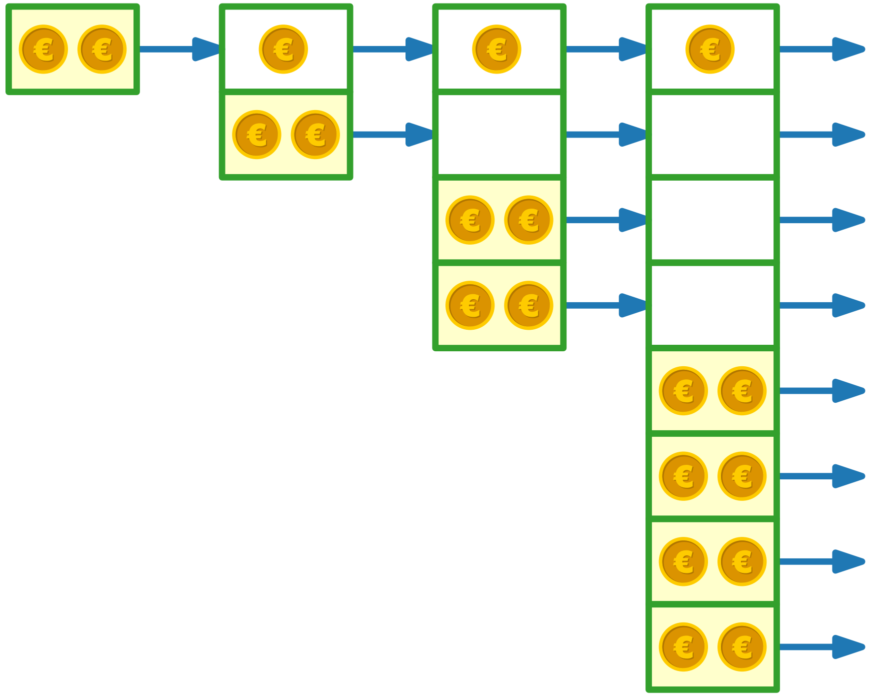
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$ — $i-1$ Elemente werden kopiert
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0

Potentialmethode für dynamische Tabellen

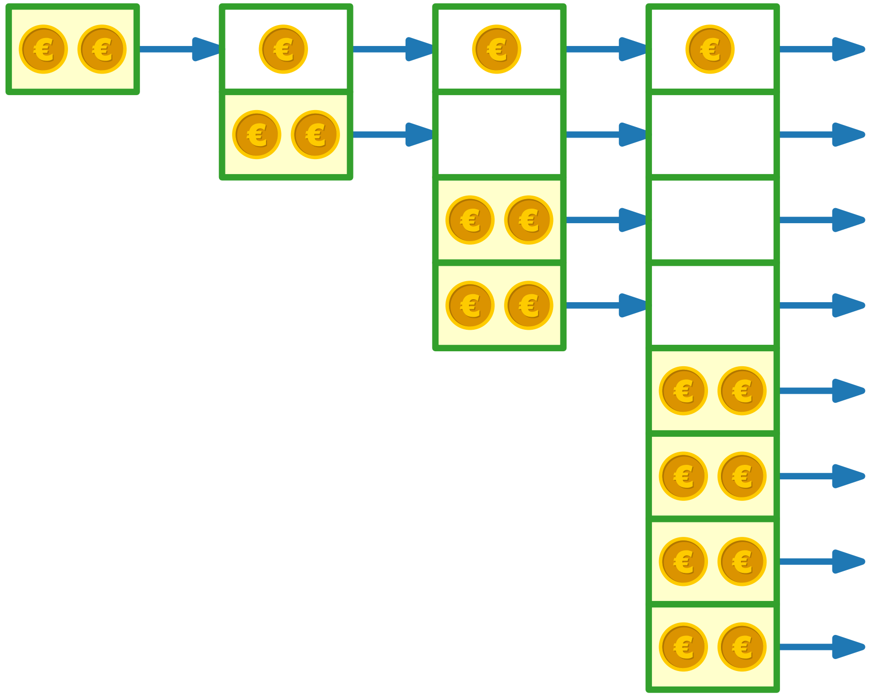
Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$i-1$ Elemente werden kopiert

INSERT(1)

$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1				

Potentialmethode für dynamische Tabellen

Idee.

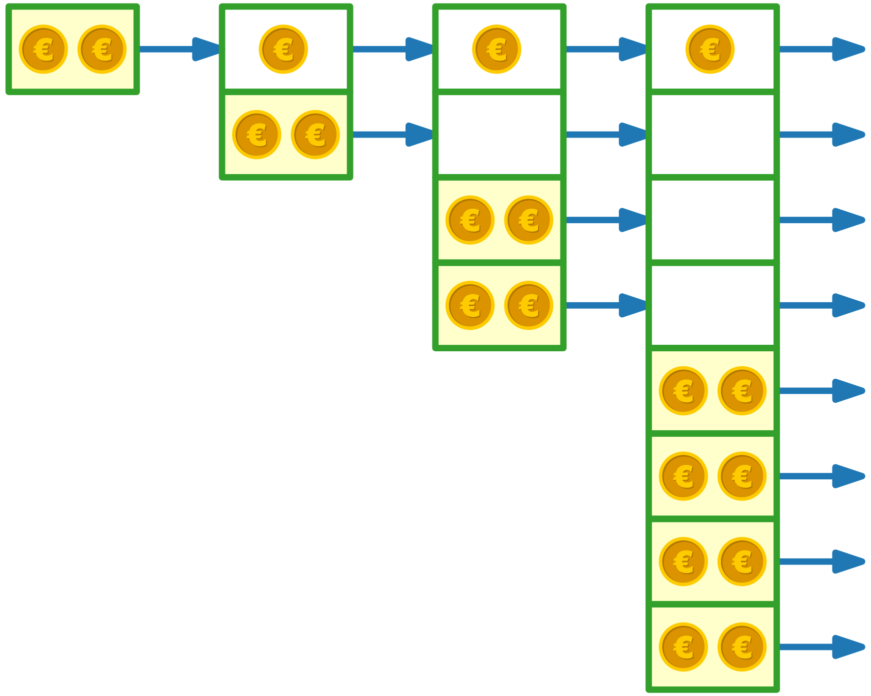
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

  $i-1$ Elemente werden kopiert

  INSERT(1) $\boxed{1}$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1		1		

Potentialmethode für dynamische Tabellen

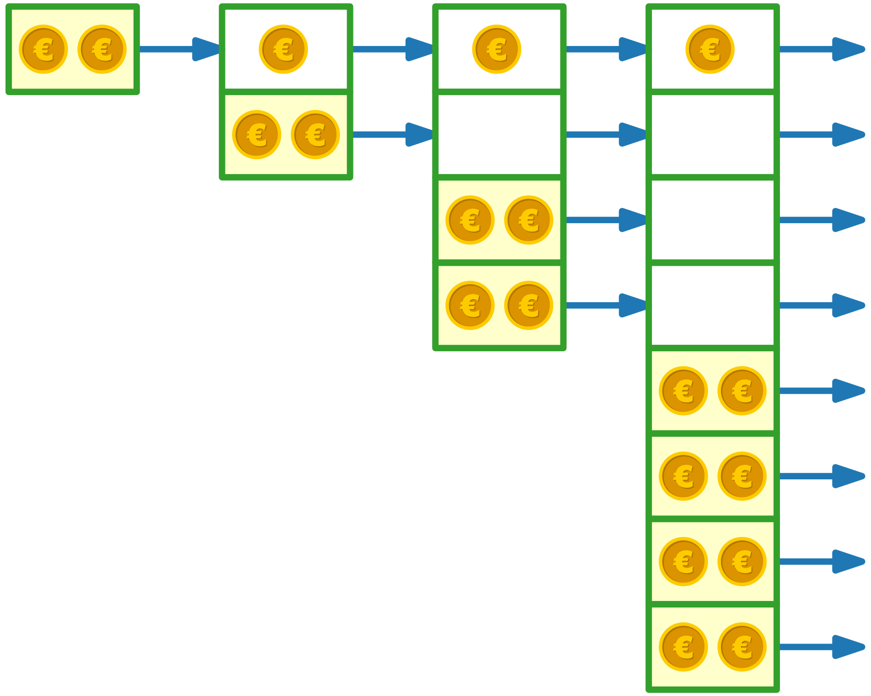
Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - \underbrace{(i-1)}_{i-1 \text{ Elemente werden kopiert}}$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \mathit{table\text{-}size}_i$

INSERT(1) $\boxed{1}$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1		1	2	

Potentialmethode für dynamische Tabellen

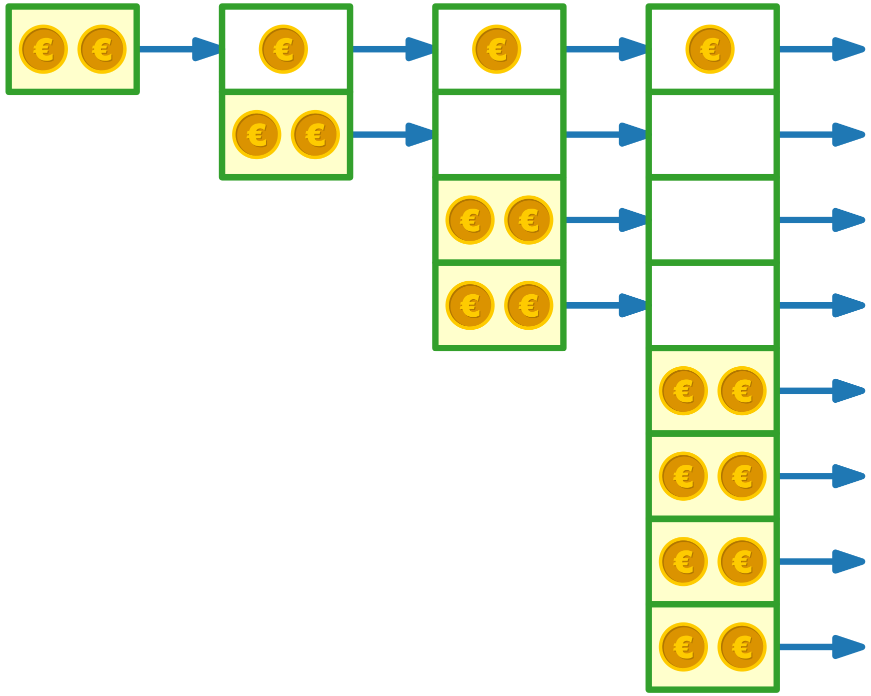
Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i - 1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - table\text{-}size_i$

$i - 1$ Elemente werden kopiert

$\textsc{Insert}(1)$ $\boxed{1}$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1		1	2	2

Potentialmethode für dynamische Tabellen

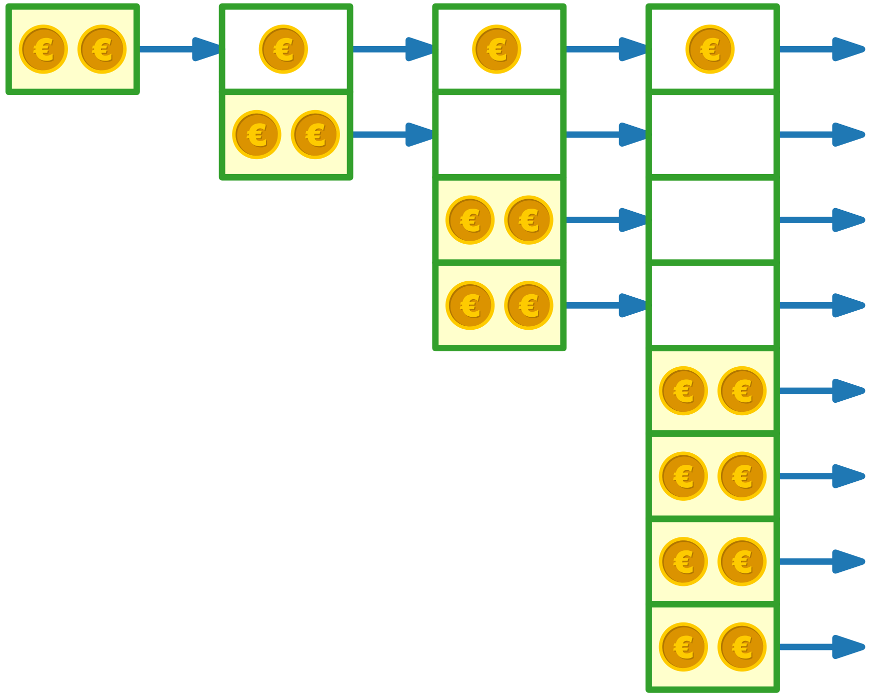
Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (\underbrace{i-1}_{i-1\text{ Elemente werden kopiert}})$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - table\text{-}size_i$

INSERT(1) $\boxed{1}$

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2

Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i - 1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

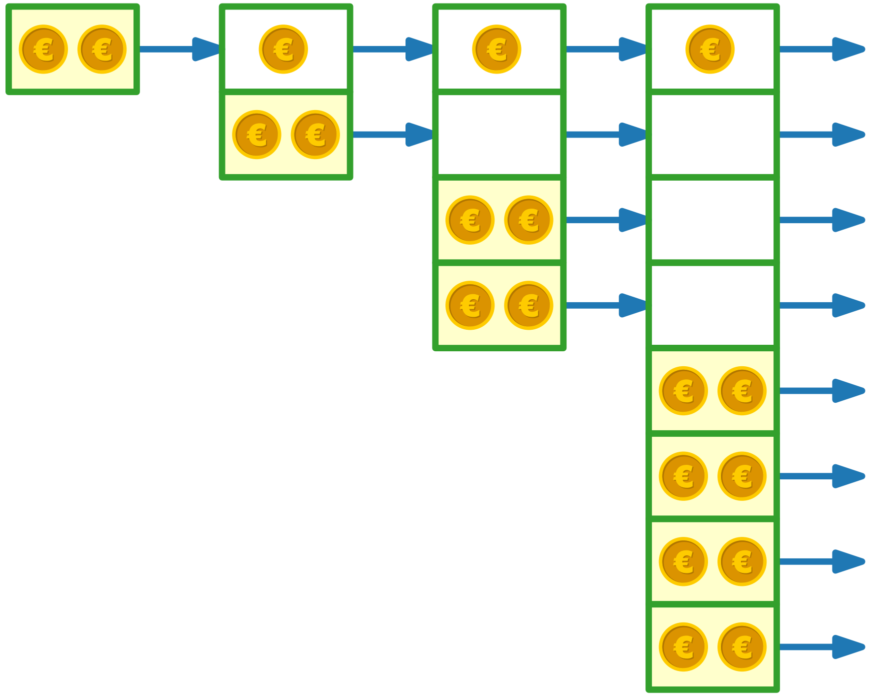
$i - 1$ Elemente werden kopiert

INSERT(1) $\boxed{1}$

INSERT(2)

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



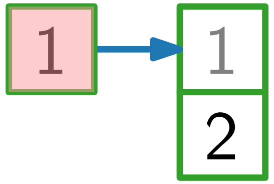
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2				

Potentialmethode für dynamische Tabellen

Idee.

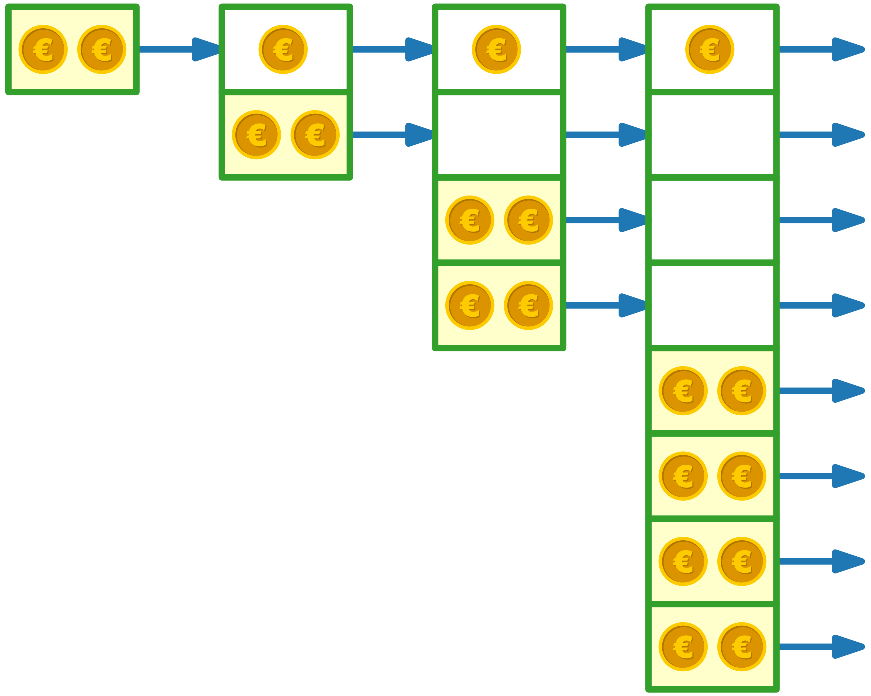
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - table\text{-}size_i$

$i-1$ Elemente werden kopiert



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$

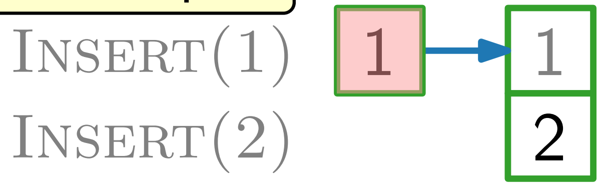


i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2				

Potentialmethode für dynamische Tabellen

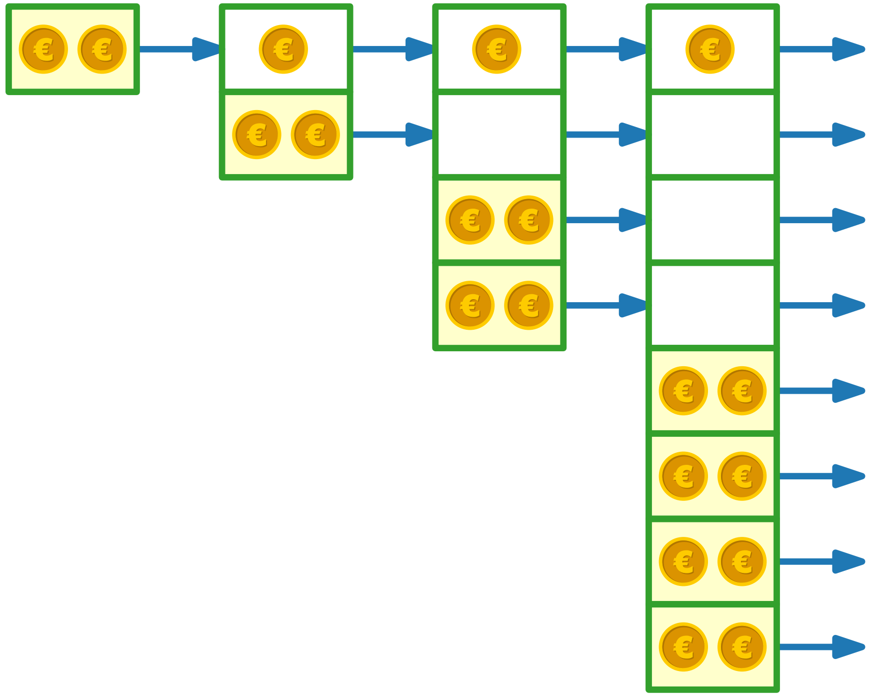
Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

$i-1$ Elemente werden kopiert



$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



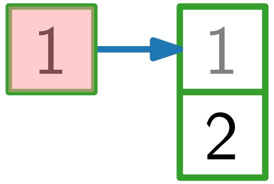
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2		2		

Potentialmethode für dynamische Tabellen

Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

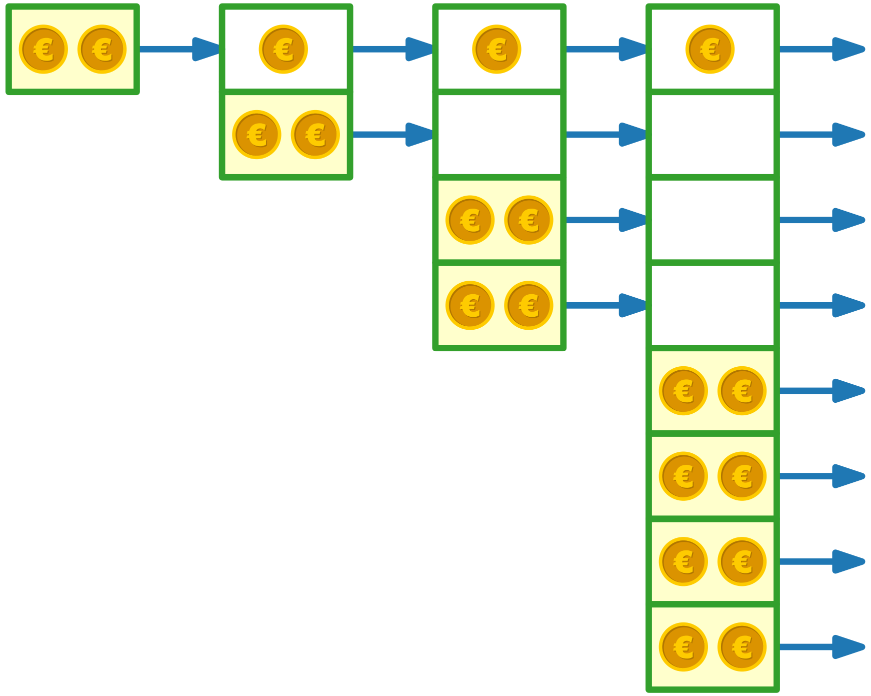
$i - 1$ Elemente werden kopiert

INSERT(1)
INSERT(2)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2		2	1	

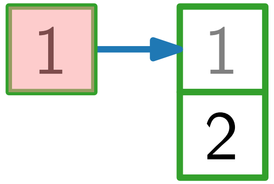
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i - 1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

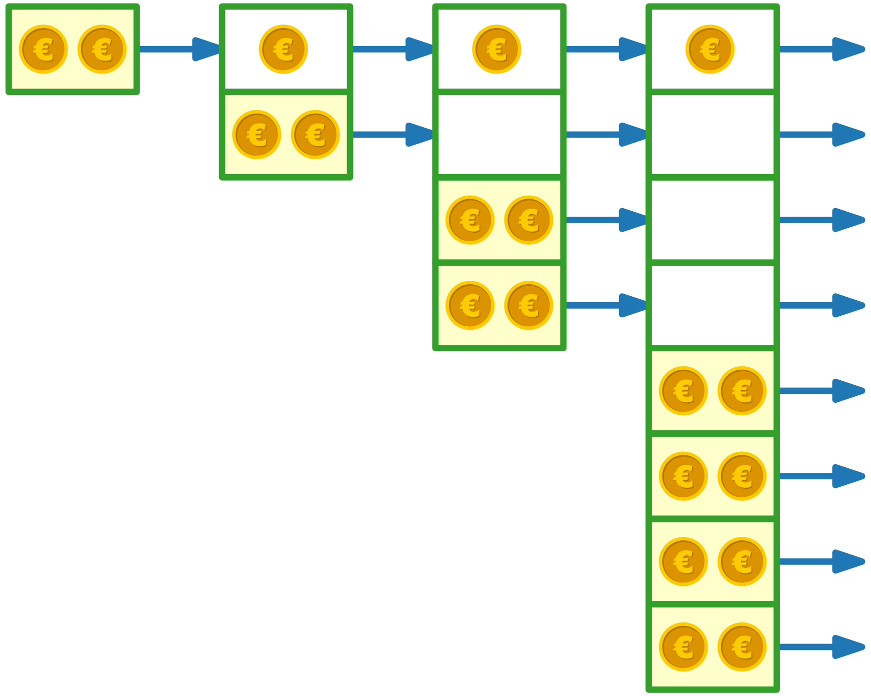
$i - 1$ Elemente werden kopiert

INSERT(1)
INSERT(2)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2		2	1	3

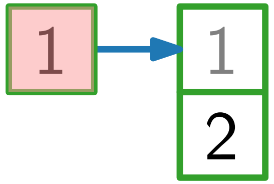
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

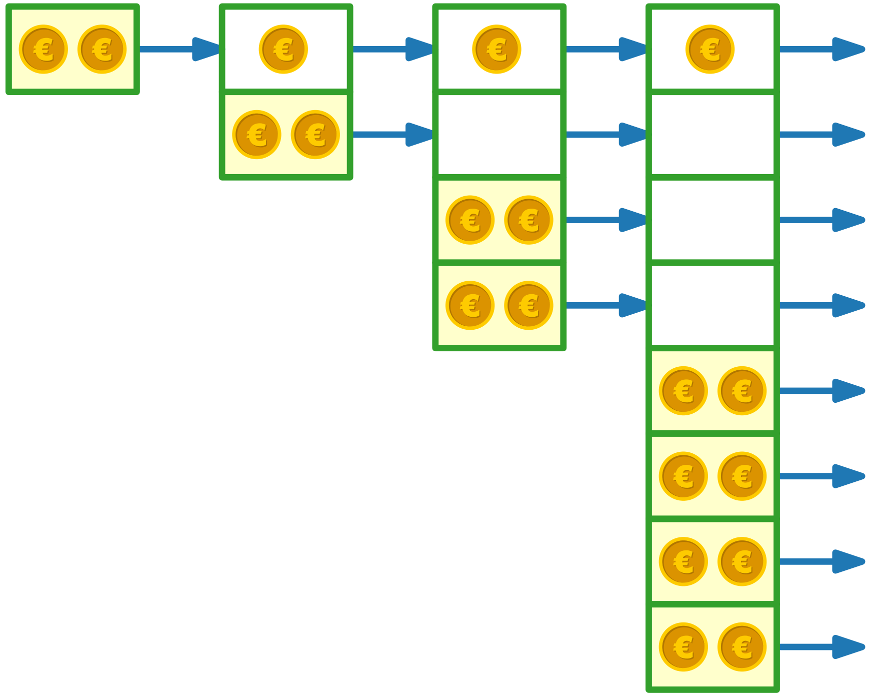
$i - 1$ Elemente werden kopiert

INSERT(1)
INSERT(2)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3

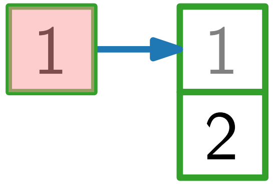
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

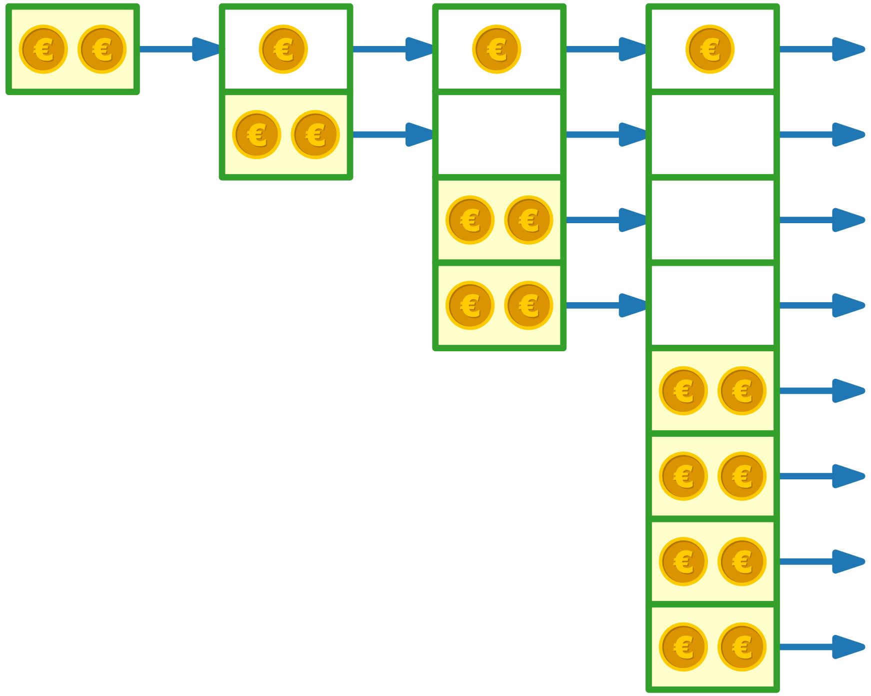
$i-1$ Elemente werden kopiert

INSERT(1)
INSERT(2)
INSERT(3)



$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



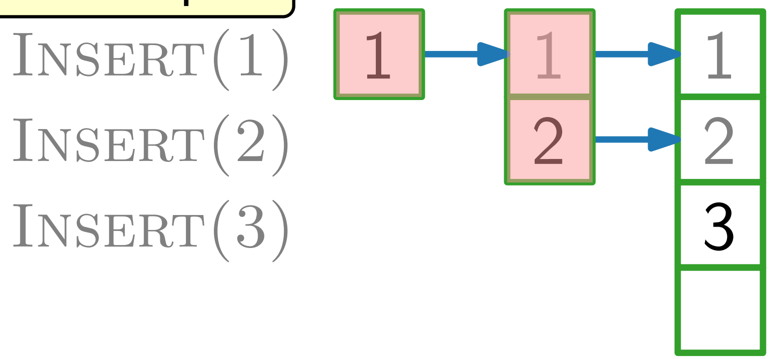
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3				

Potentialmethode für dynamische Tabellen

Idee.

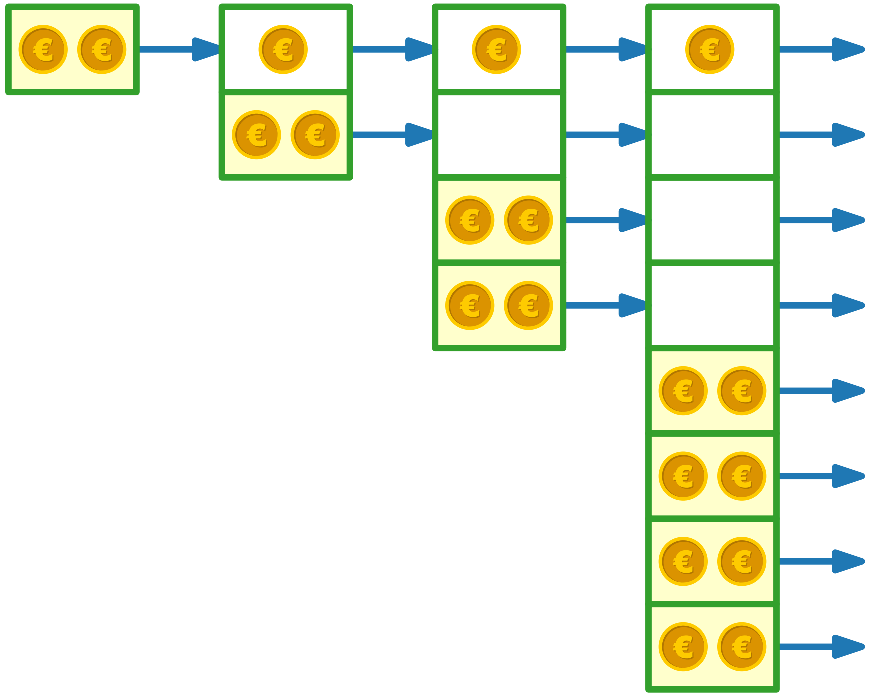
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$i - 1$ Elemente werden kopiert



$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



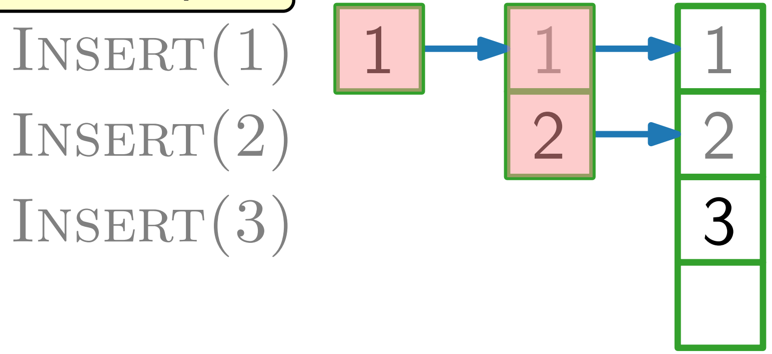
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3				

Potentialmethode für dynamische Tabellen

Idee.

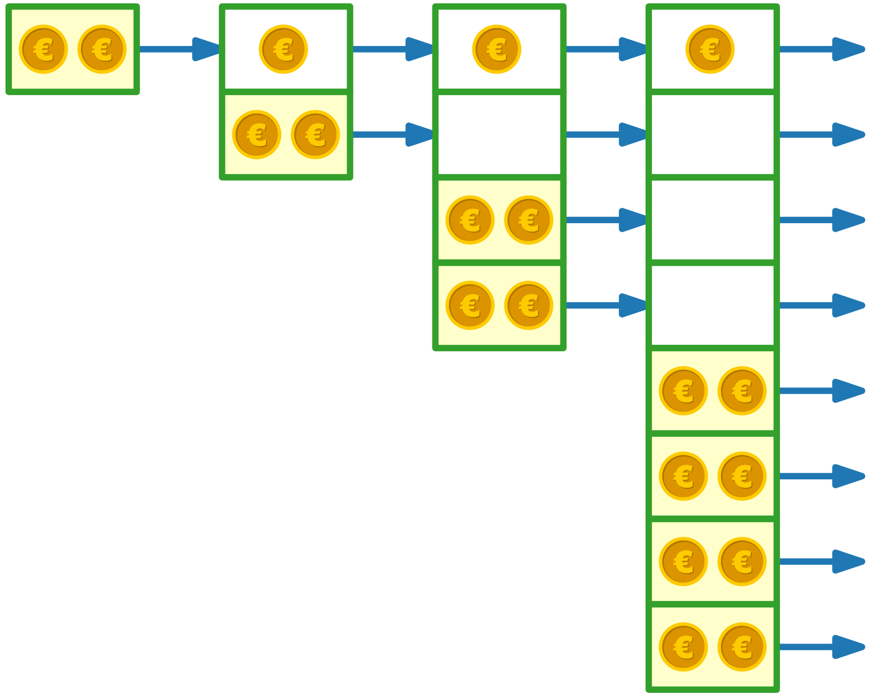
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

$i - 1$ Elemente werden kopiert



$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3		3		

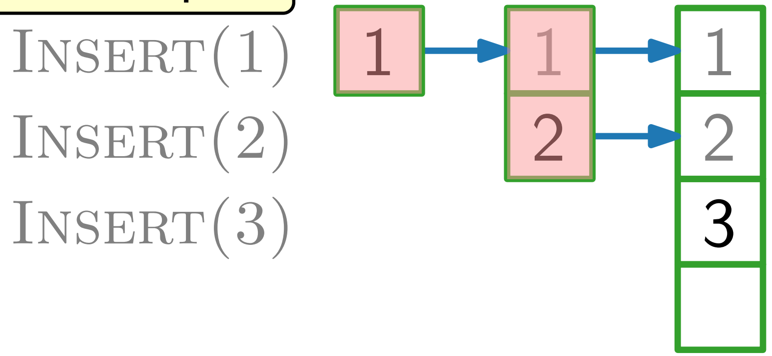
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

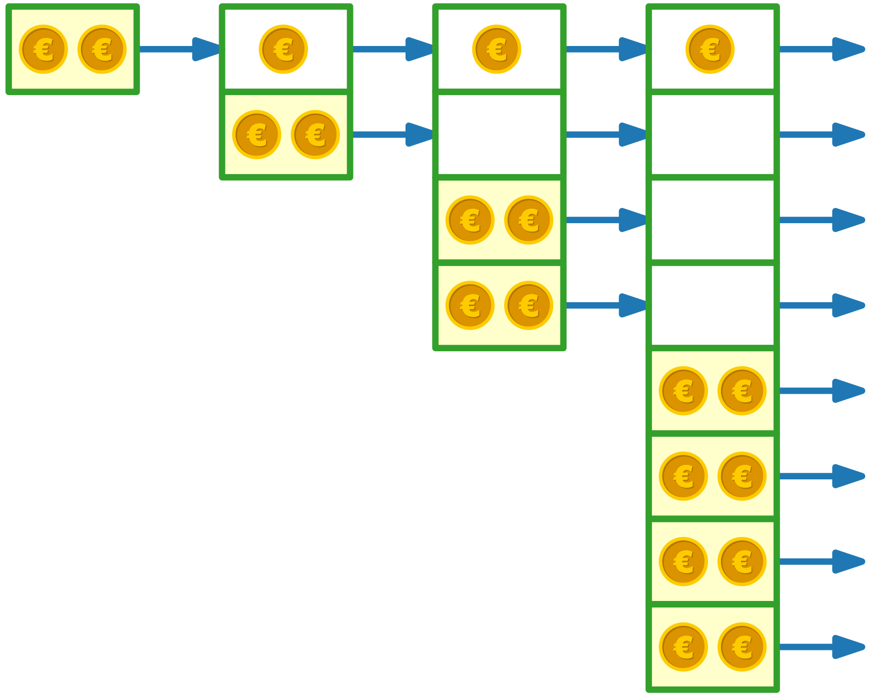
$i-1$ Elemente werden kopiert

- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



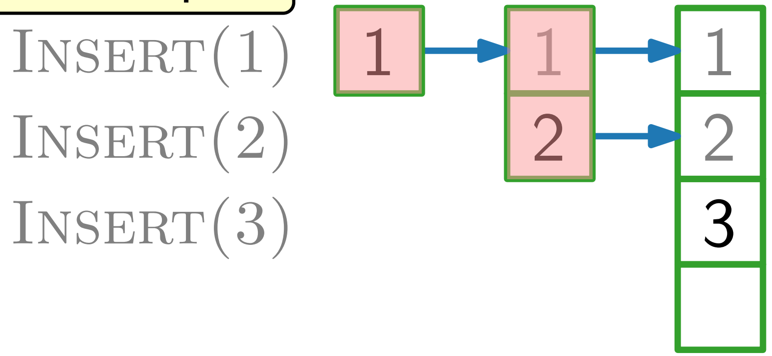
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3		3	0	

Potentialmethode für dynamische Tabellen

Idee.

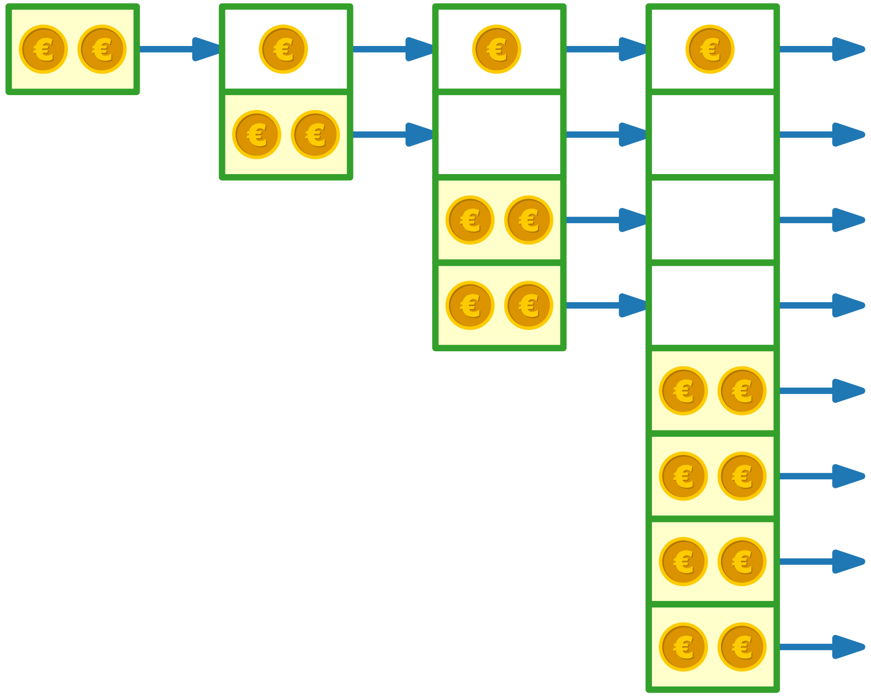
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$i-1$ Elemente werden kopiert



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^n c_i$$



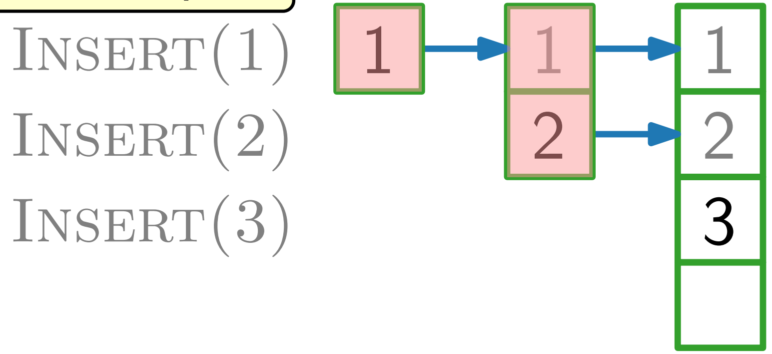
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3		3	0	3

Potentialmethode für dynamische Tabellen

Idee.

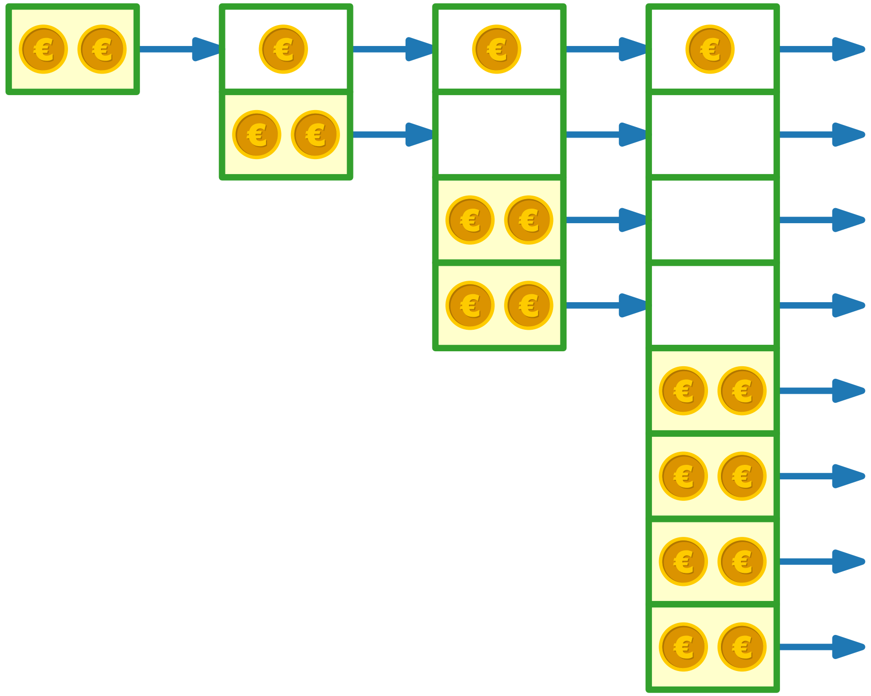
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$i-1$ Elemente werden kopiert



$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^n c_i$



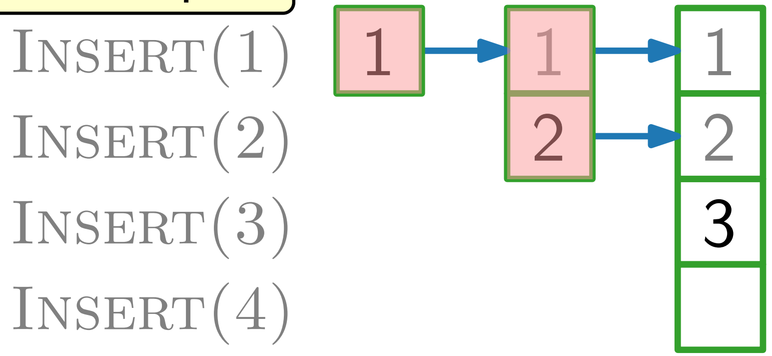
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3

Potentialmethode für dynamische Tabellen

Idee.

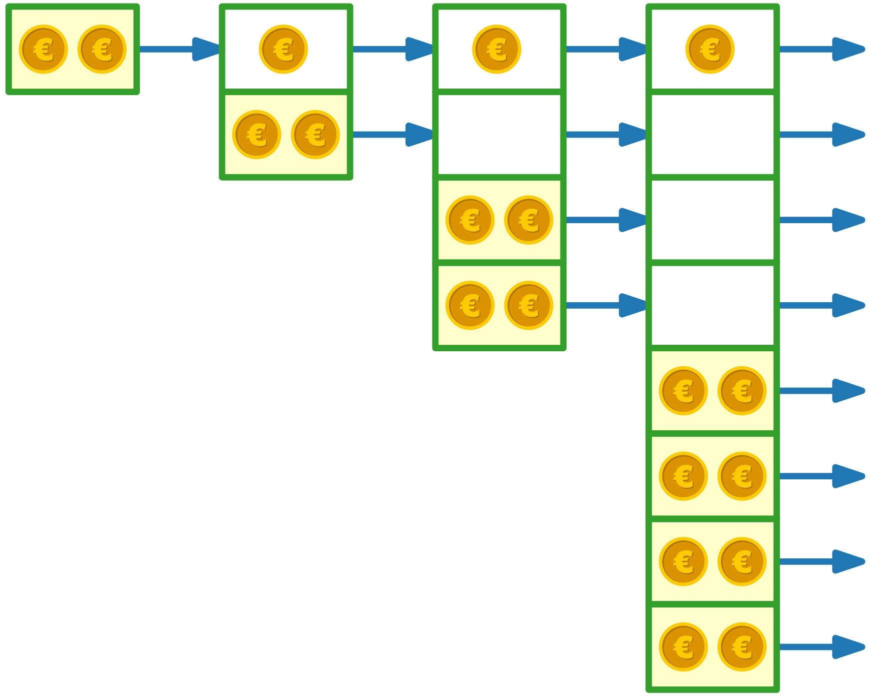
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$i-1$ Elemente werden kopiert



$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4				

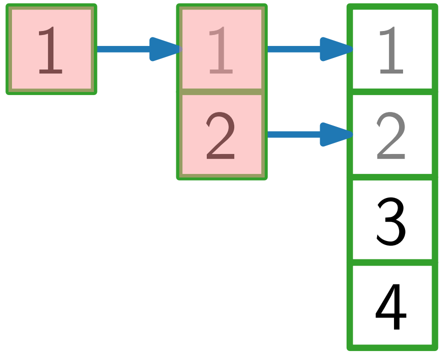
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

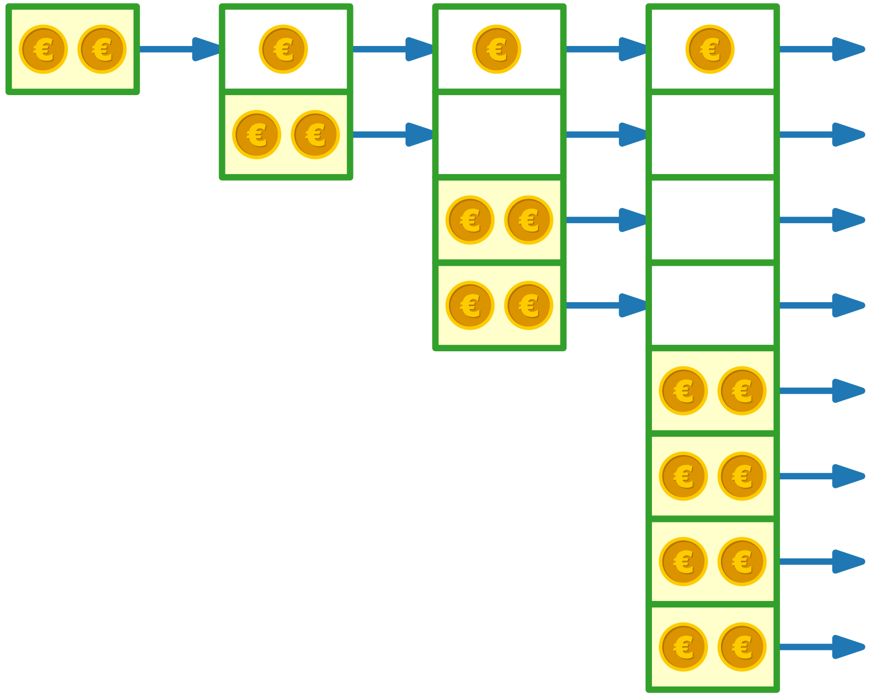
$i-1$ Elemente werden kopiert

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



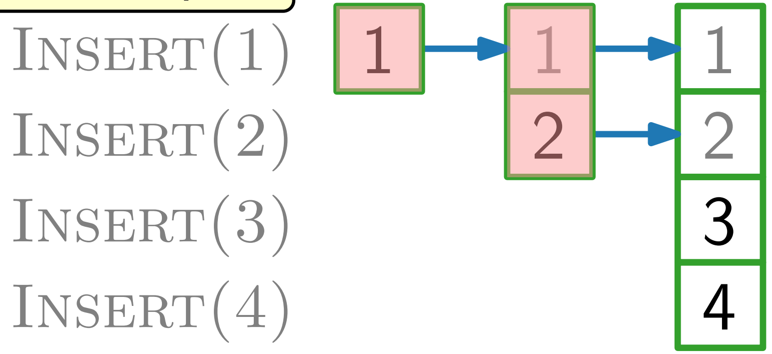
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4				

Potentialmethode für dynamische Tabellen

Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

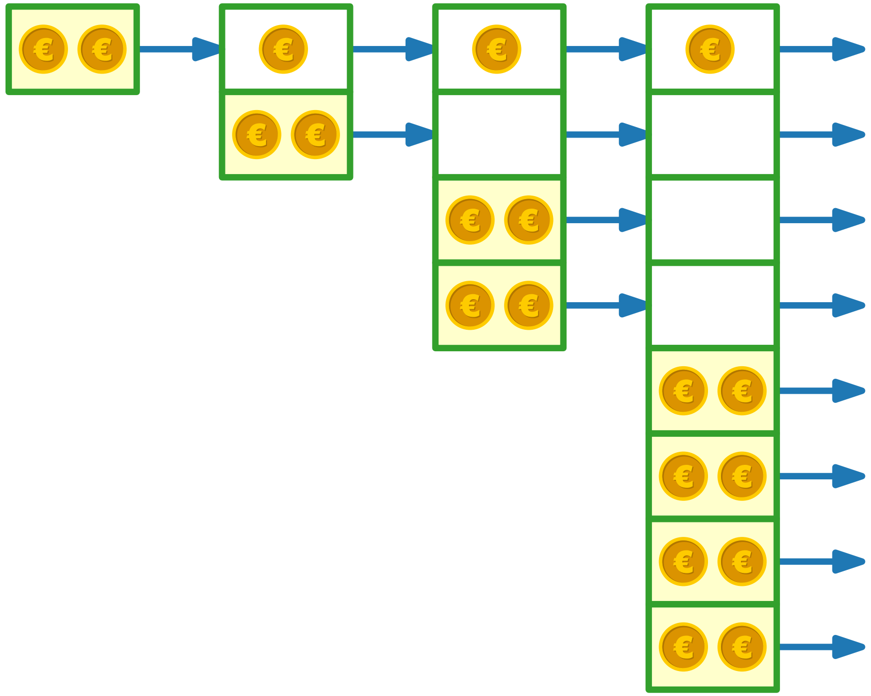
$i-1$ Elemente werden kopiert

- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4		1		

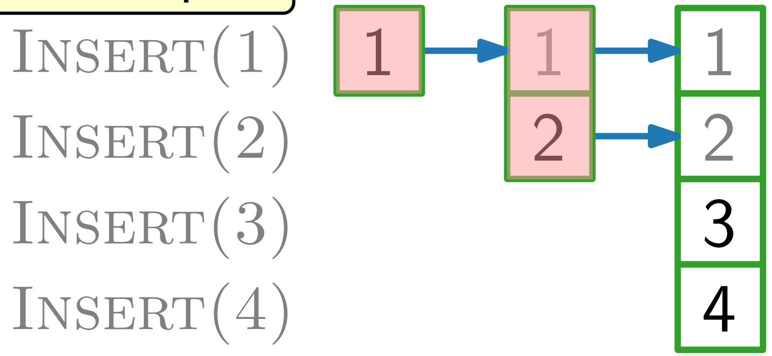
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

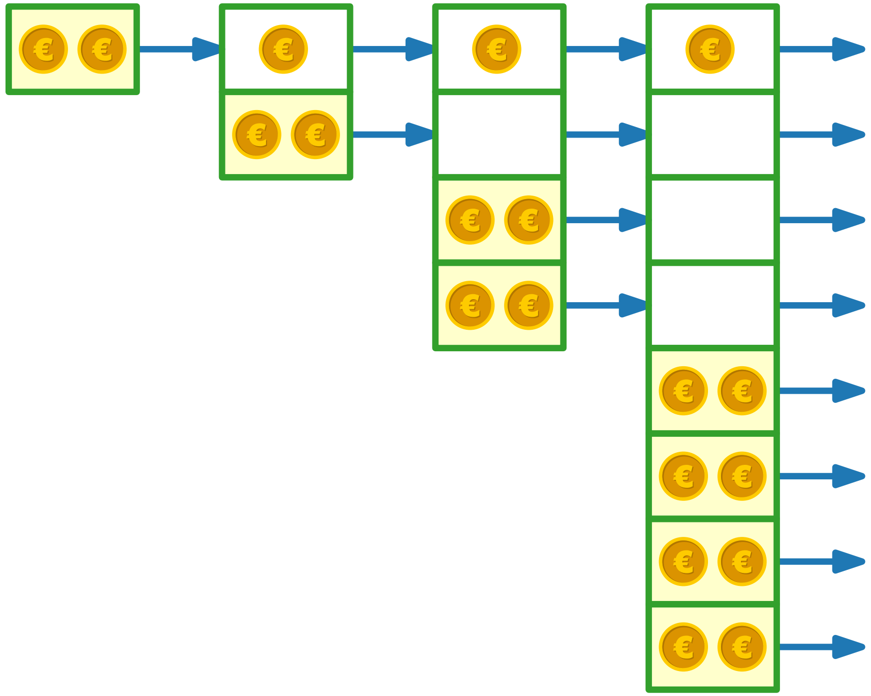
$i-1$ Elemente werden kopiert

- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$



$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



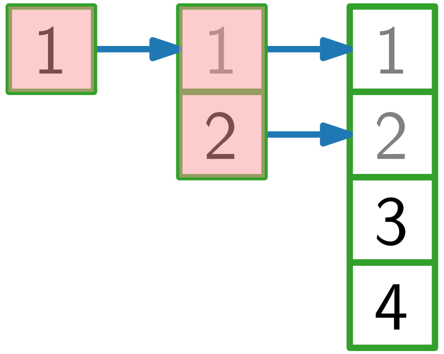
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4		1	2	

Potentialmethode für dynamische Tabellen

Idee.

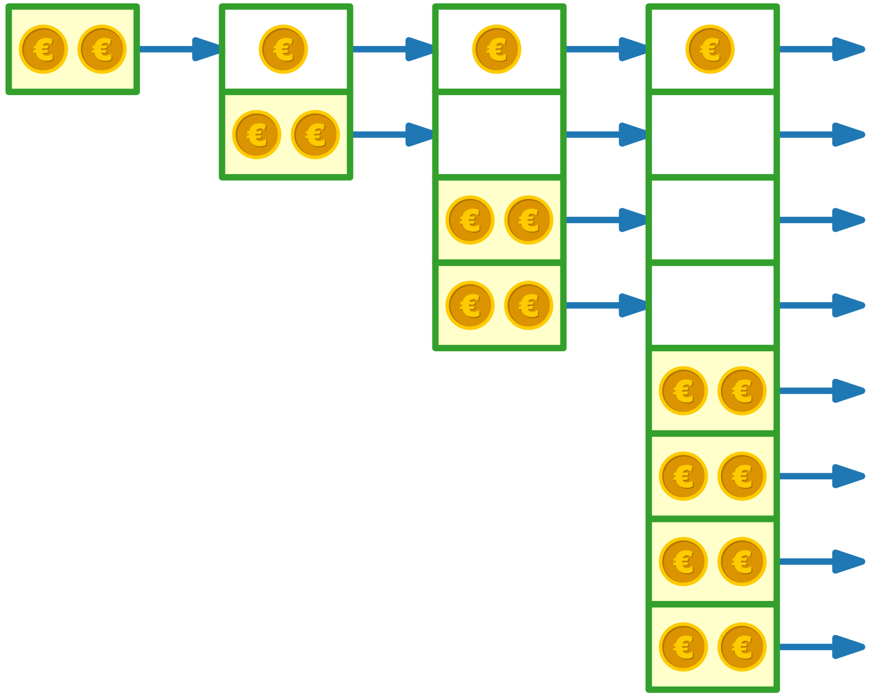
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$ — $i-1$ Elemente werden kopiert
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^n c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4		1	2	5

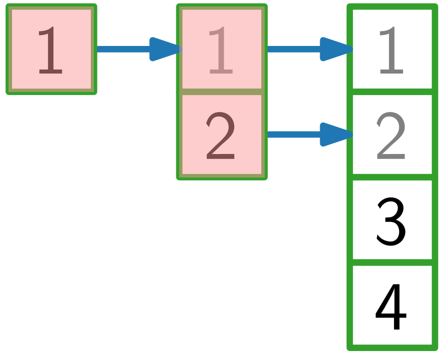
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

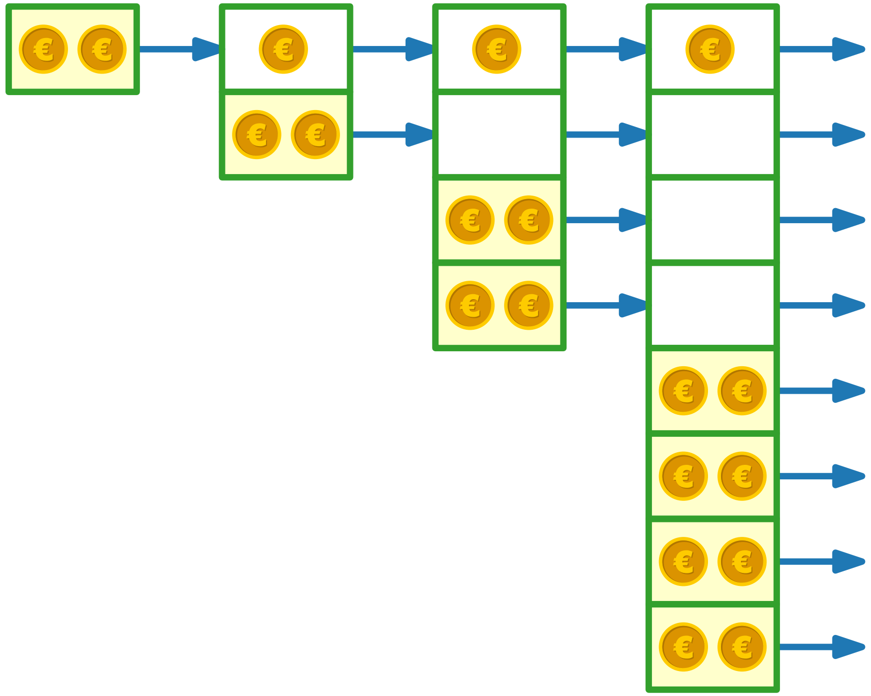
$i-1$ Elemente werden kopiert

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5

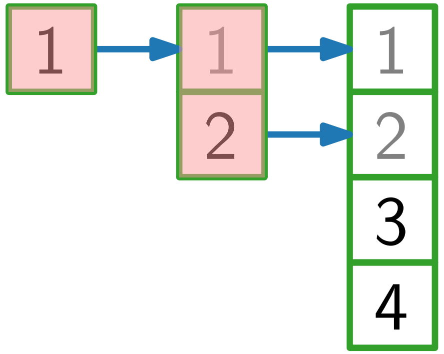
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \mathit{table\text{-}size}_i$

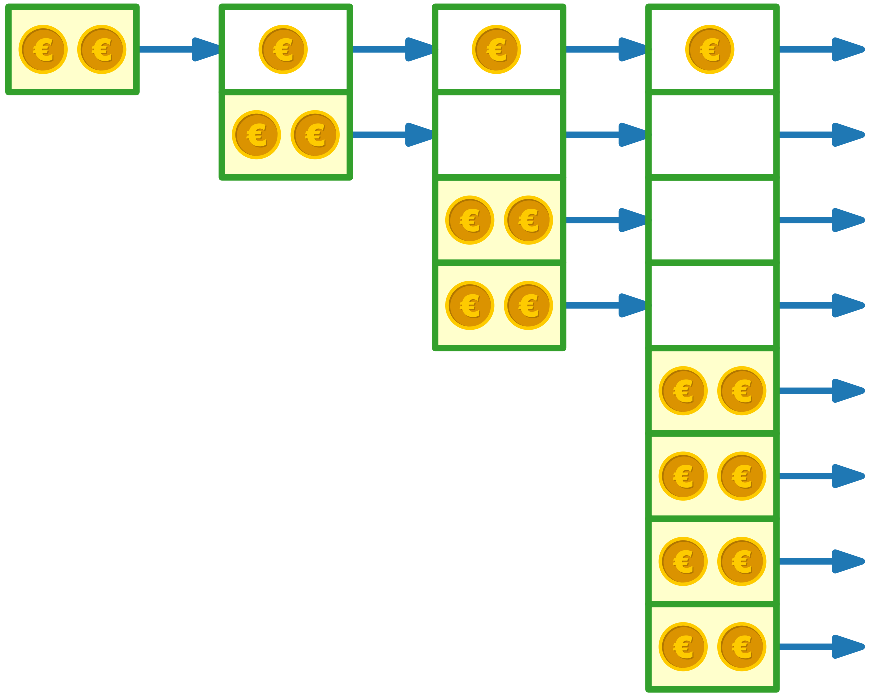
$i-1$ Elemente werden kopiert

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



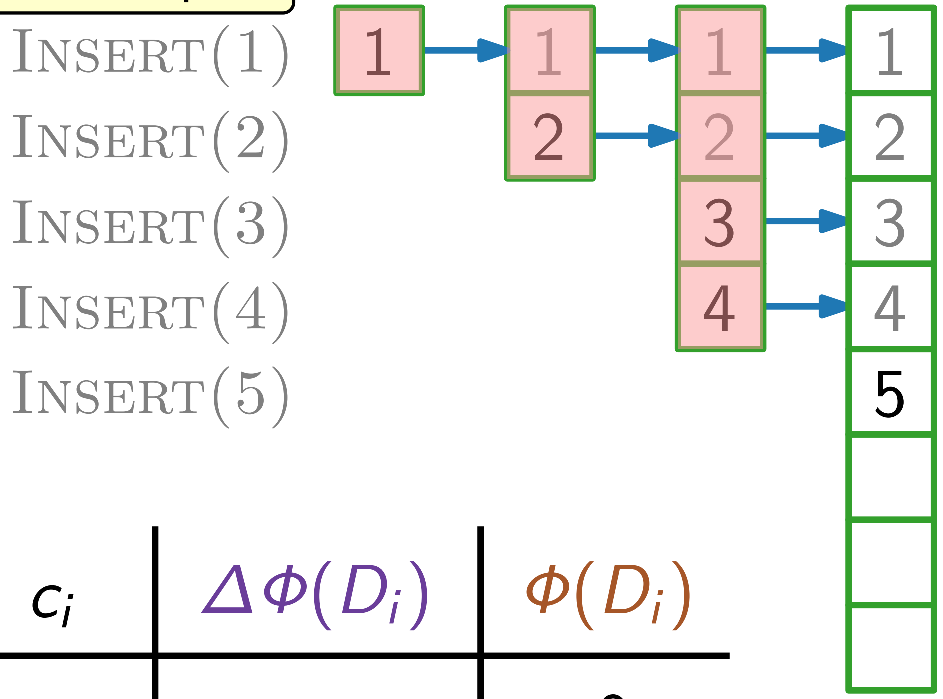
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5				

Potentialmethode für dynamische Tabellen

Idee.

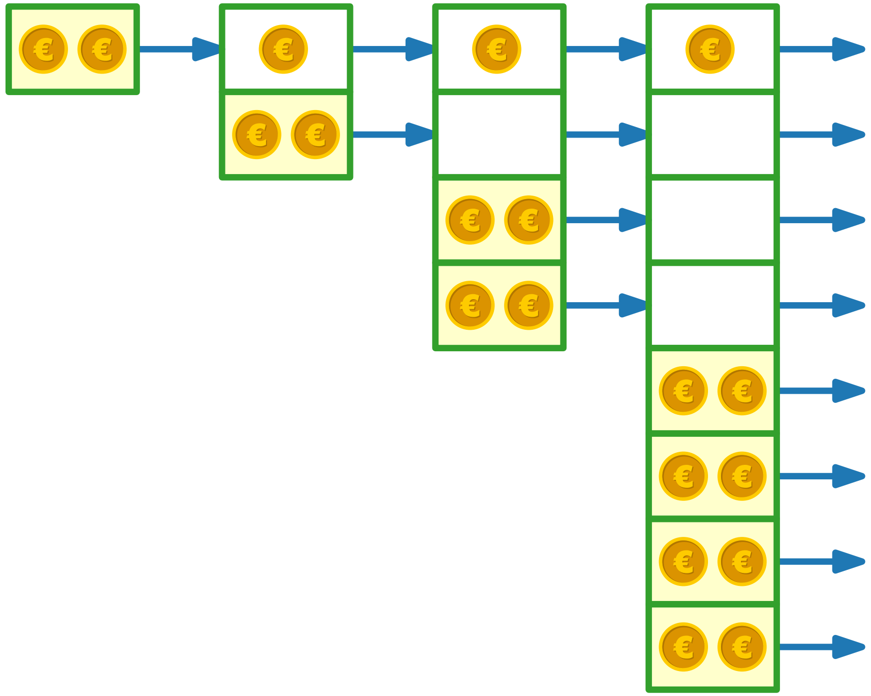
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i - 1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - table\text{-}size_i$

$i - 1$ Elemente werden kopiert



$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$



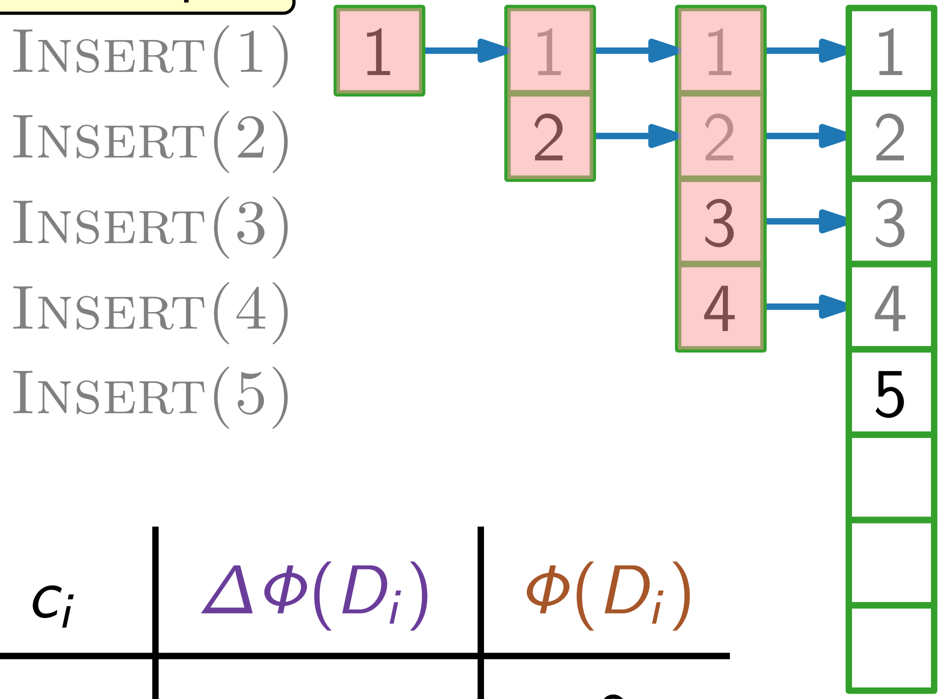
i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5				

Potentialmethode für dynamische Tabellen

Idee.

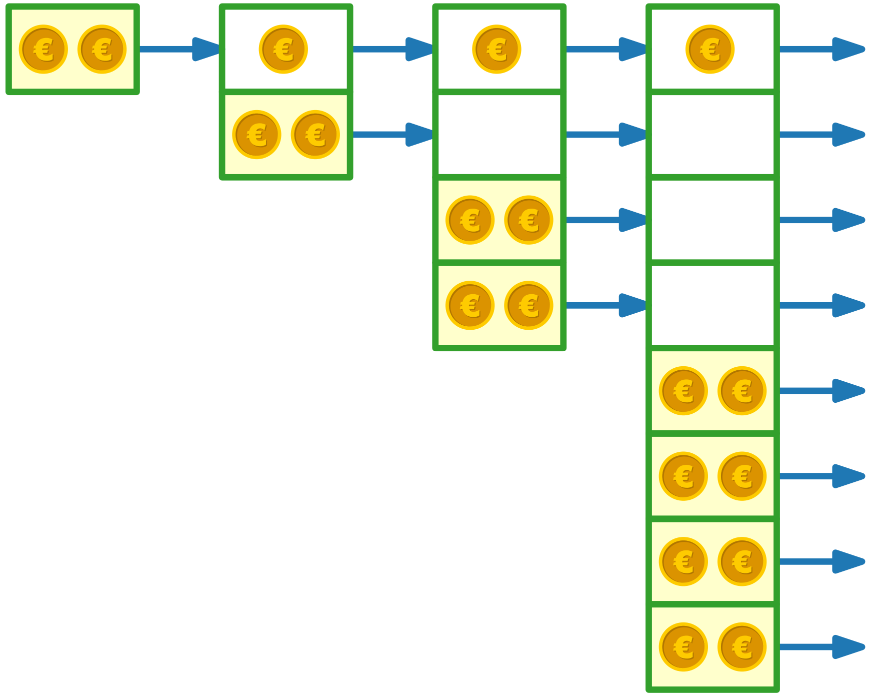
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

  $i - 1$ Elemente werden kopiert
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5		5		

Potentialmethode für dynamische Tabellen

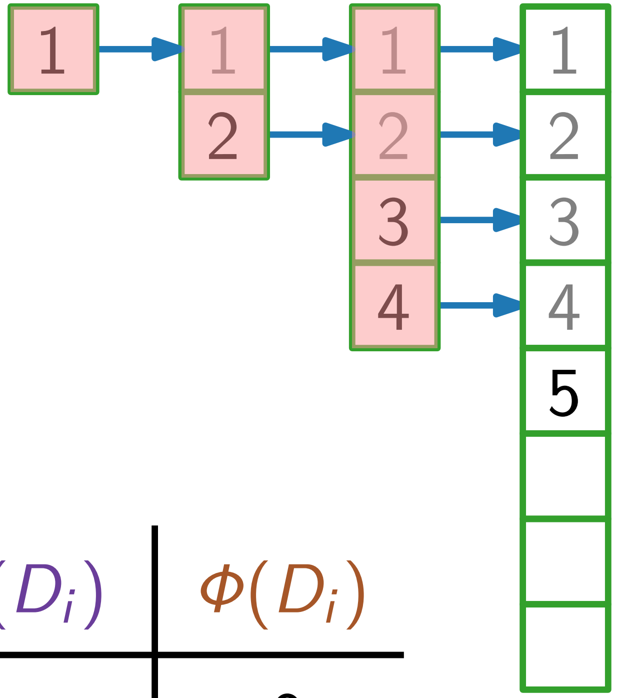
Idee.

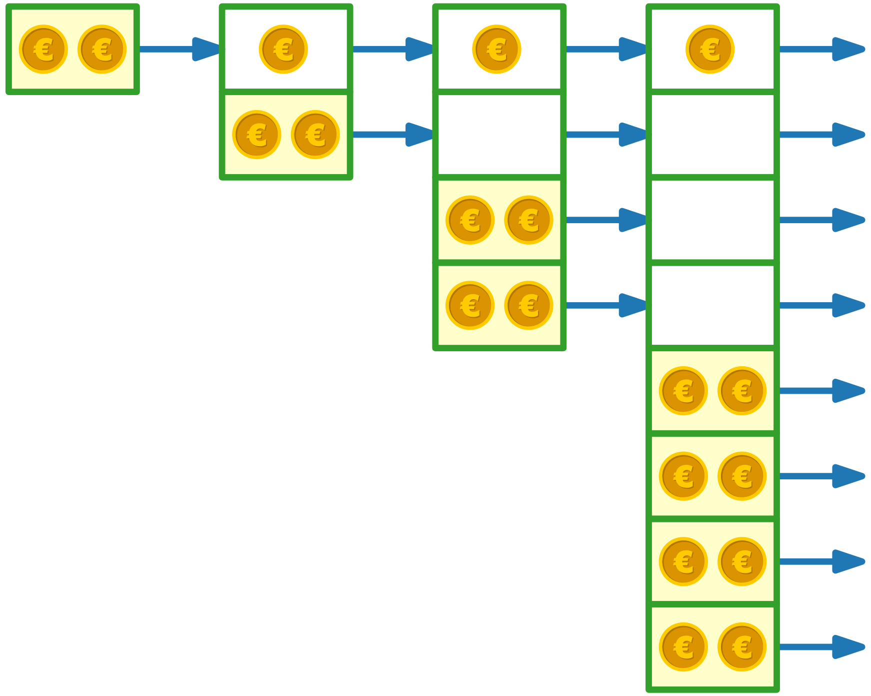
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i - 1)$ — $i - 1$ Elemente werden kopiert
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$$\hat{c}_i = c_i + \Delta\Phi(D_i),\text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)





i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5		5	-2	

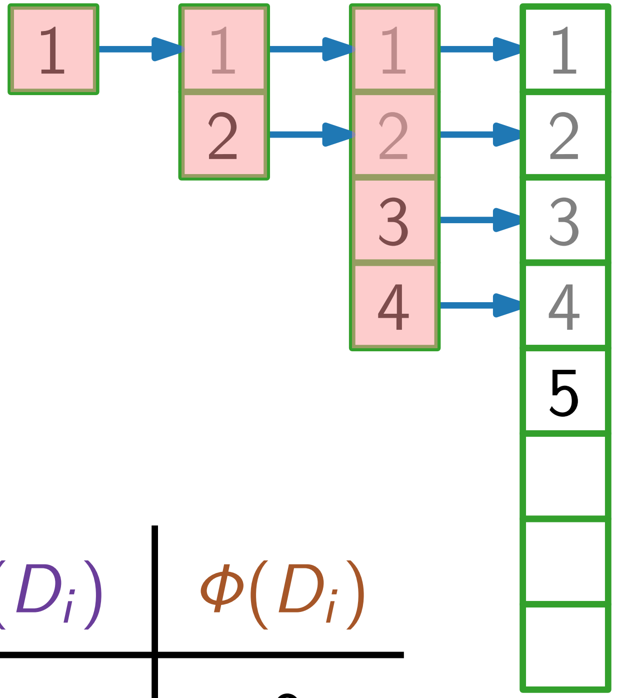
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

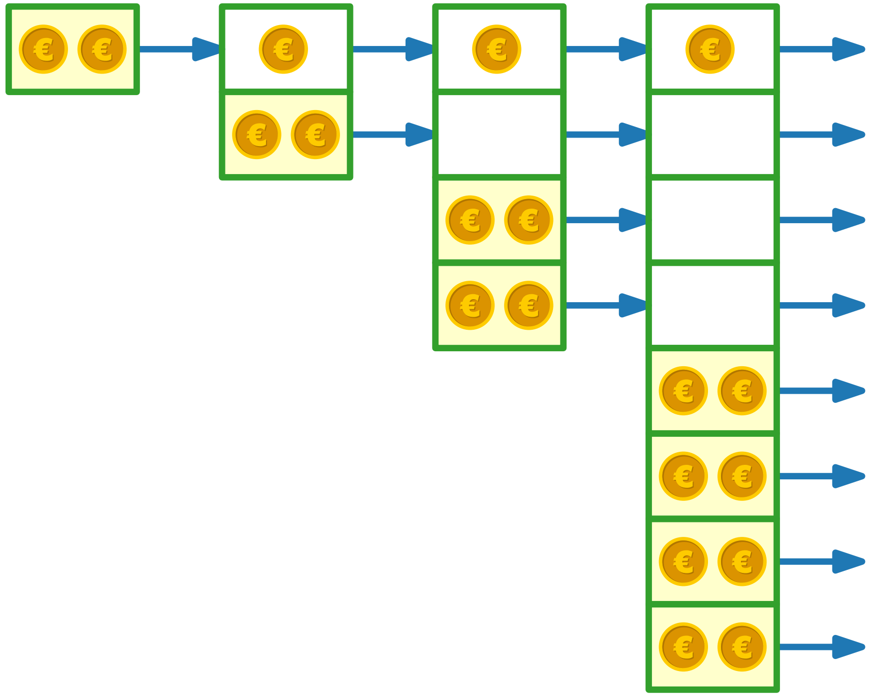
$i-1$ Elemente werden kopiert

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5		5	-2	3

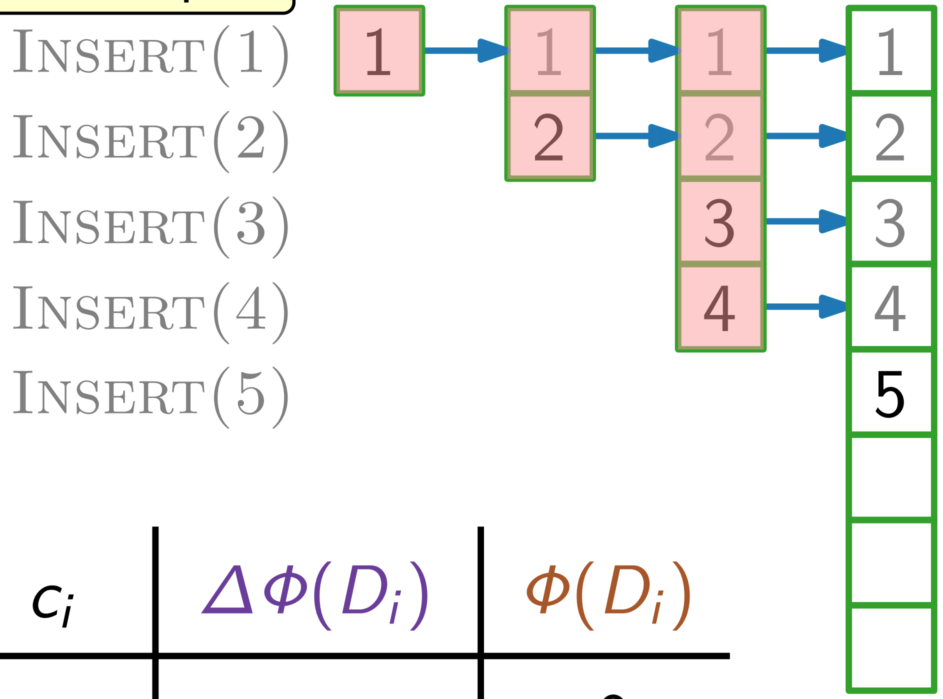
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

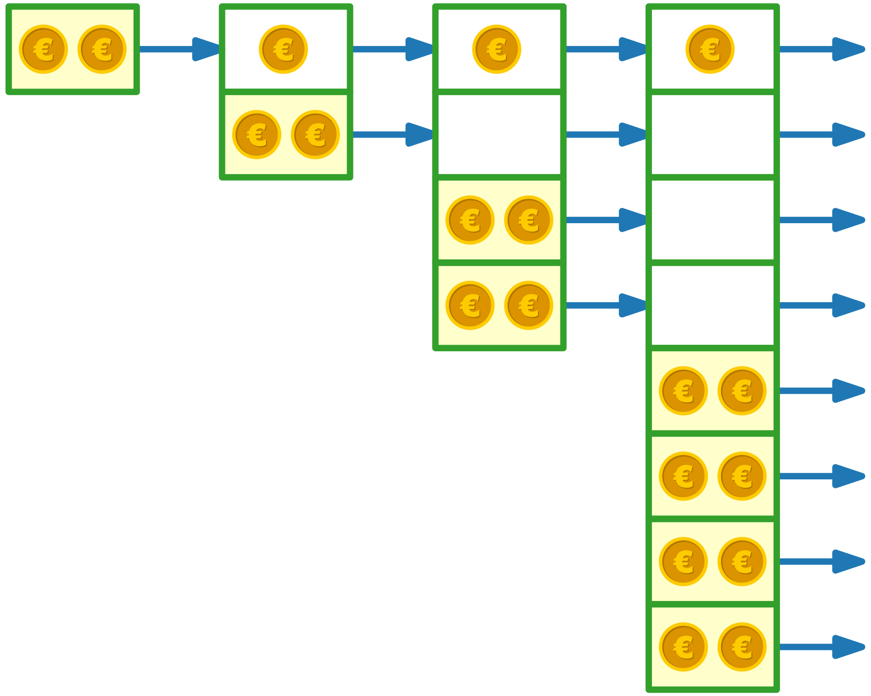
$i-1$ Elemente werden kopiert

- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - table\text{-}size_i$



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5	3	5	-2	3

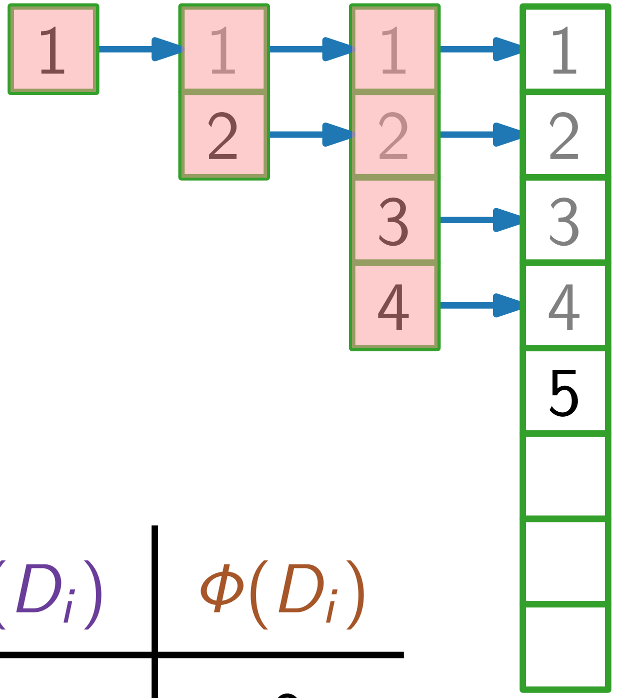
Potentialmethode für dynamische Tabellen

Idee.

- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

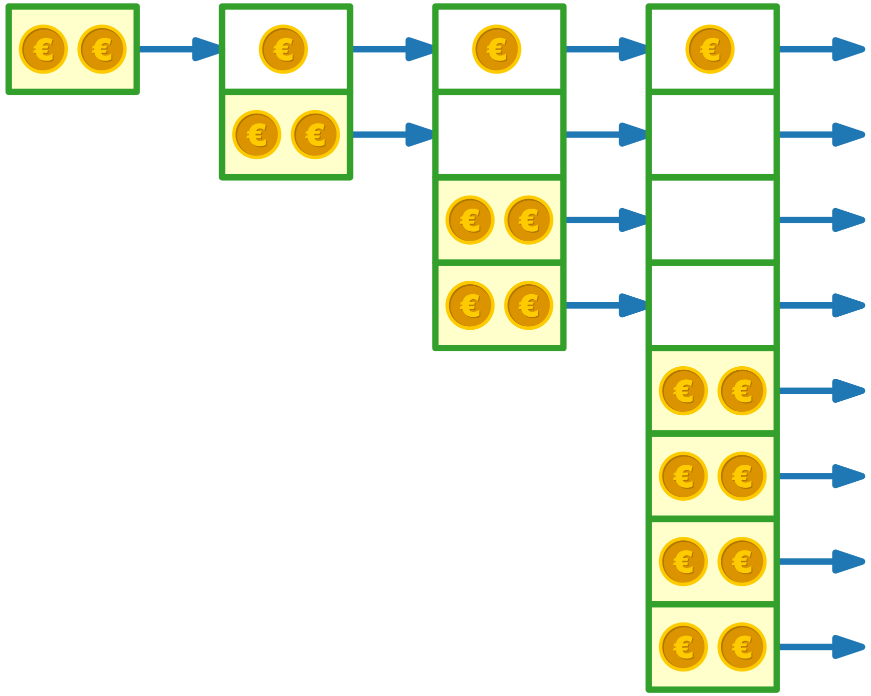
$i-1$ Elemente werden kopiert

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)
INSERT(6)



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5	3	5	-2	3
6				

Potentialmethode für dynamische Tabellen

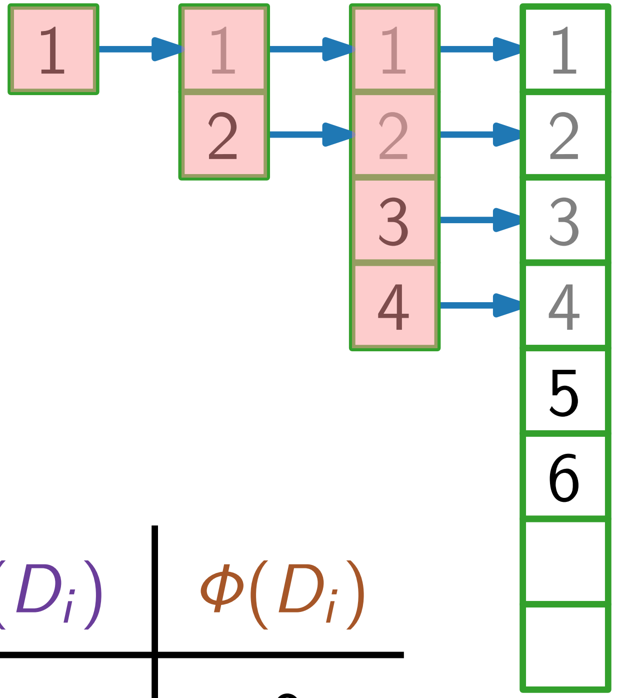
Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - table\text{-}size_i$

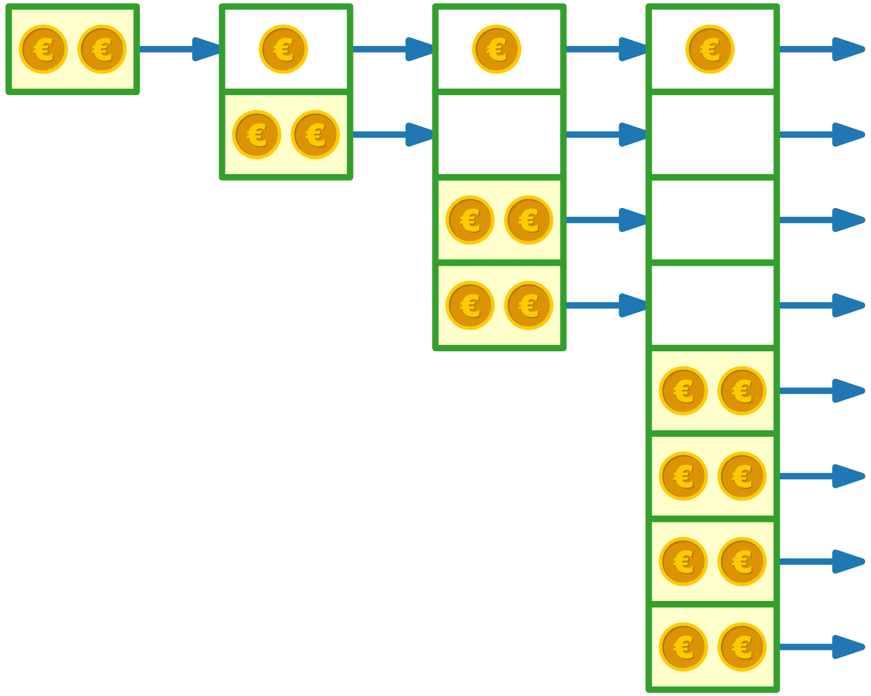
$i-1$ Elemente werden kopiert

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)
INSERT(6)



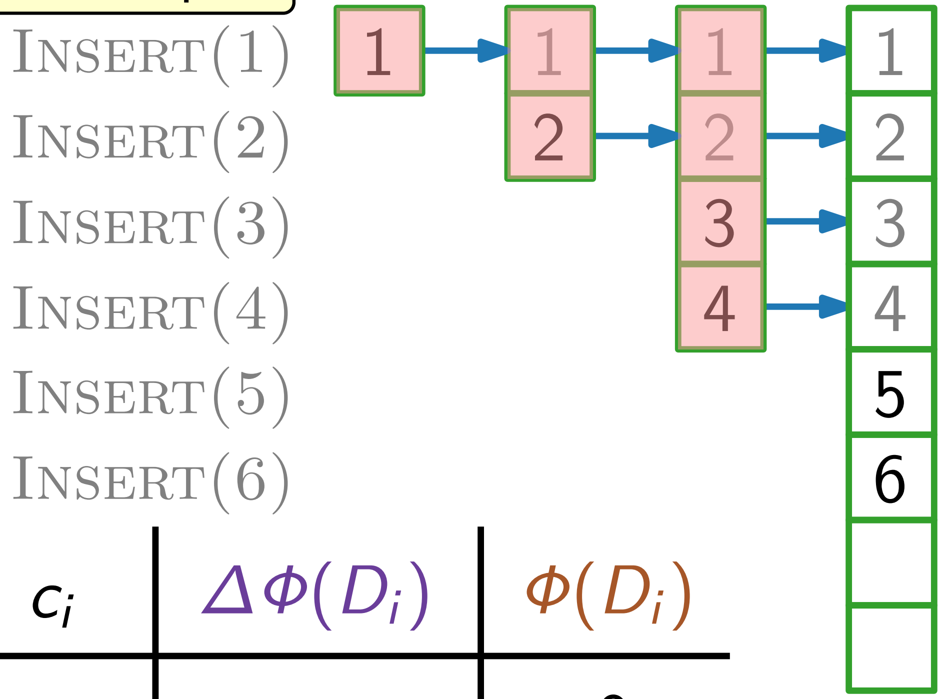


i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5	3	5	−2	3
6				

Potentialmethode für dynamische Tabellen

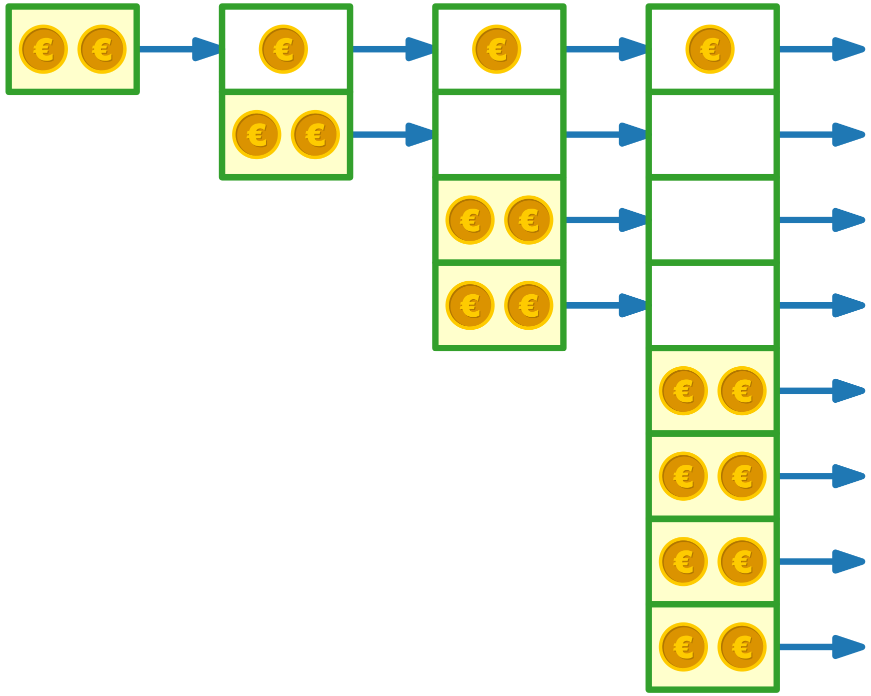
Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \mathit{table\text{-}size}_i$

$i-1$ Elemente werden kopiert



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^n c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5	3	5	−2	3
6	3	1	2	5

Potentialmethode für dynamische Tabellen

Idee.

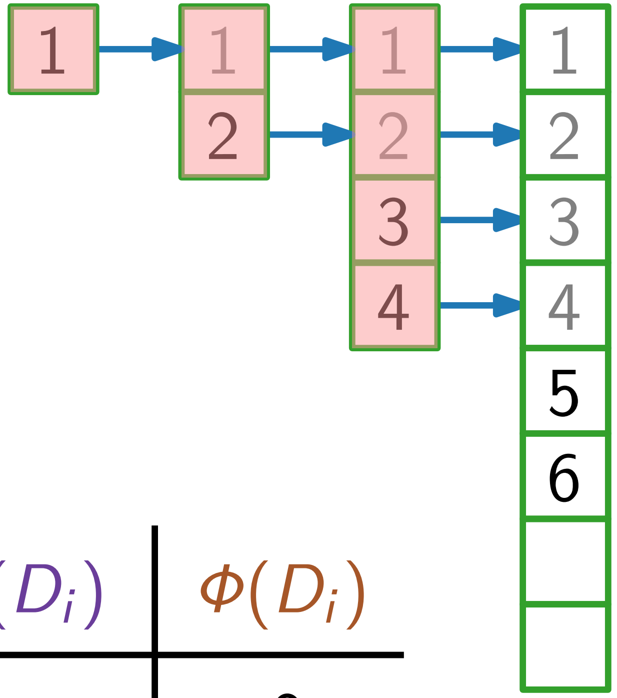
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

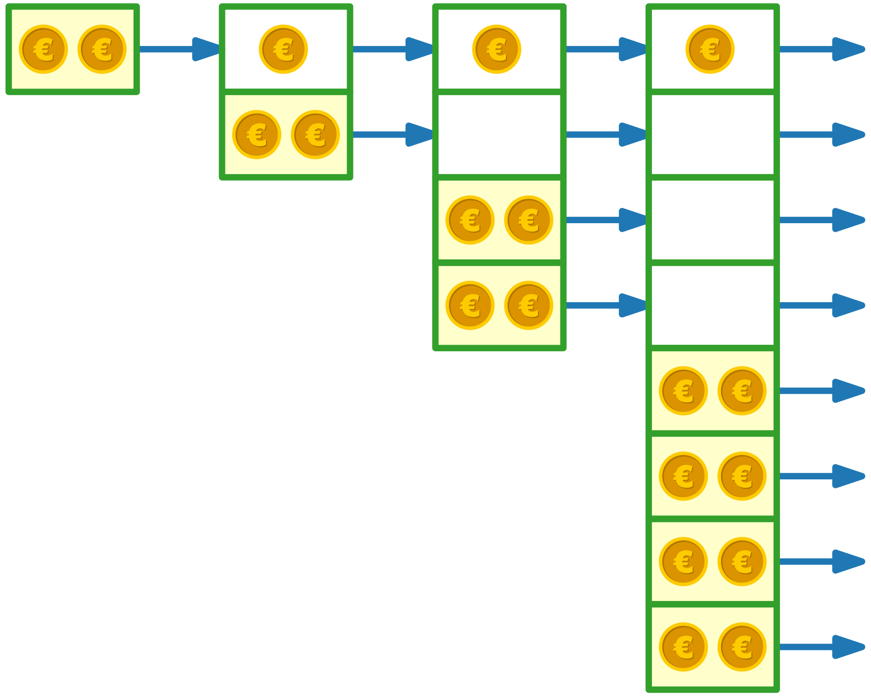
$i-1$ Elemente werden kopiert

$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$$

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)
INSERT(6)





i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5	3	5	−2	3
6	3	1	2	5

Potentialmethode für dynamische Tabellen

Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$

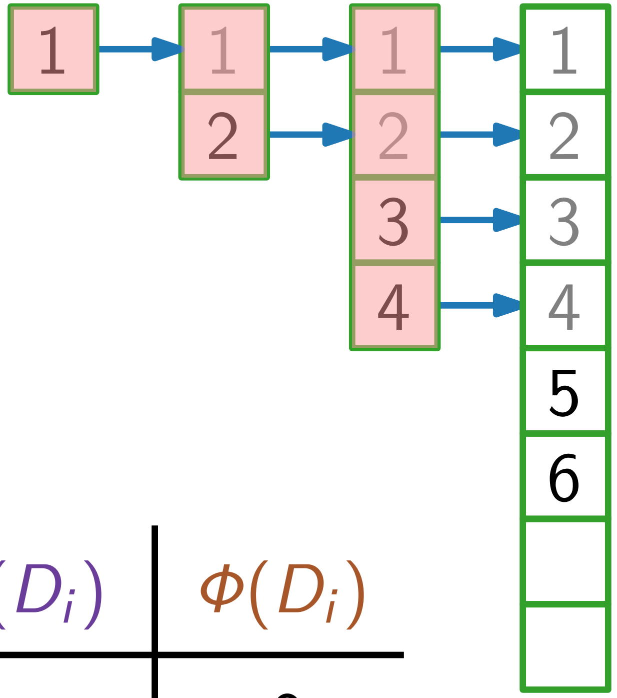
$i-1$ Elemente werden kopiert

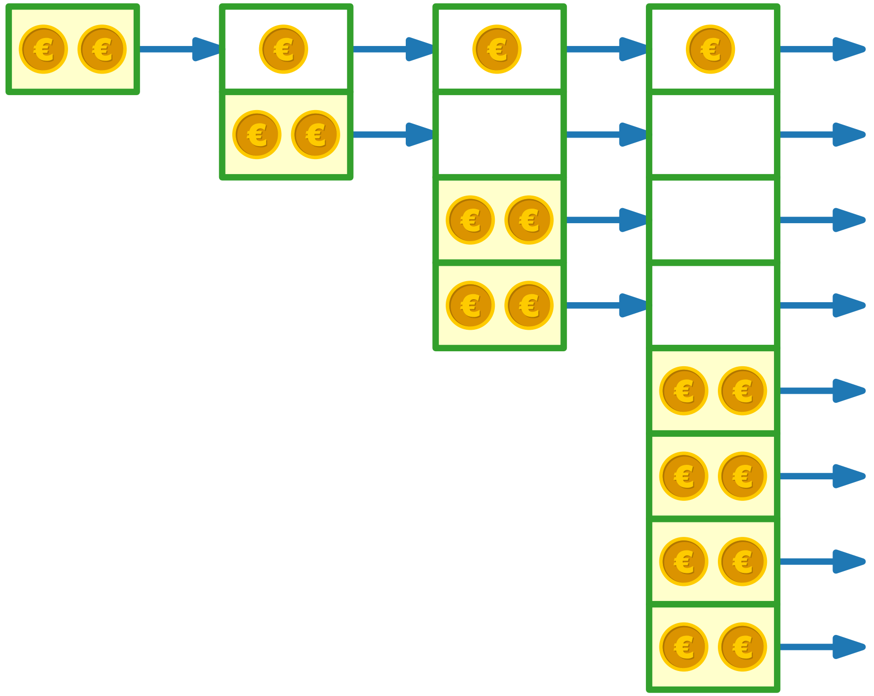
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot size_i - \textit{table-size}_i$

$\hat{c}_i = c_i + \Delta\Phi(D_i)$, wobei $\Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$

$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^{n} c_i$

INSERT(1)
INSERT(2)
INSERT(3)
INSERT(4)
INSERT(5)
INSERT(6)



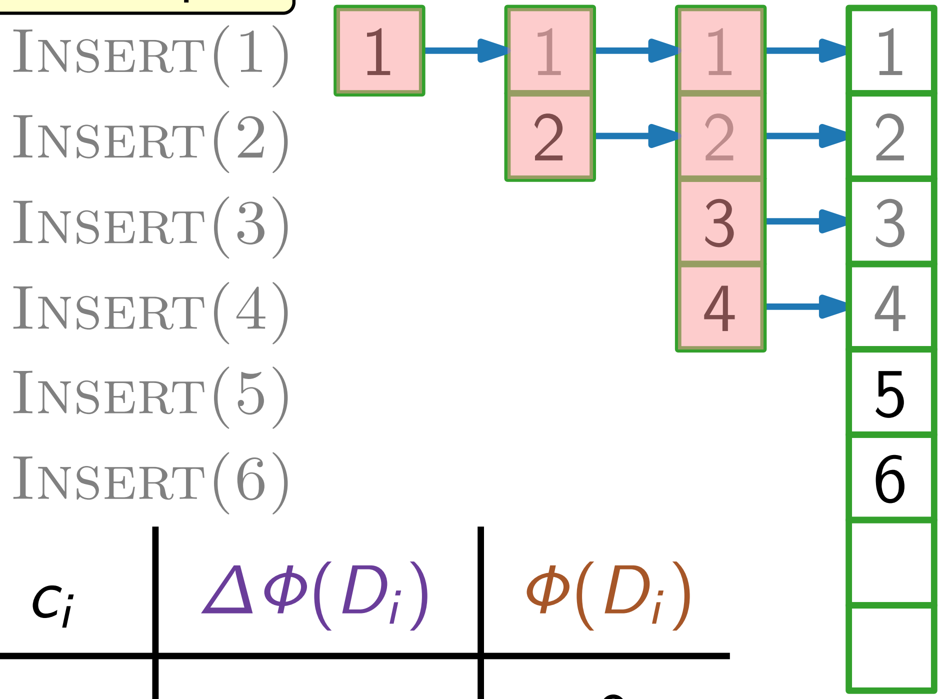


i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5	3	5	−2	3
6	3	1	2	5

Potentialmethode für dynamische Tabellen

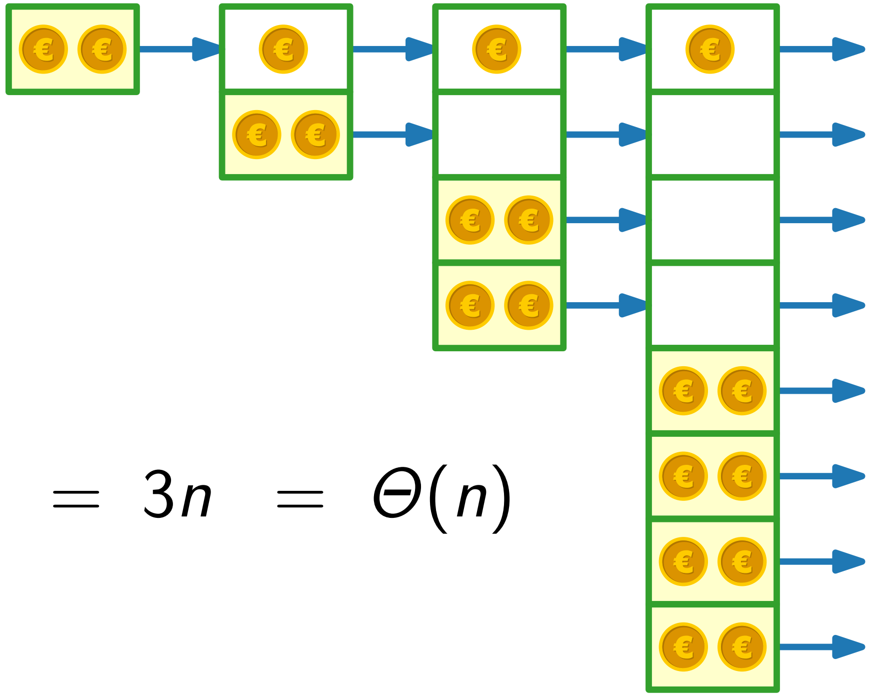
Idee.
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

$i-1$ Elemente werden kopiert



$$\hat{c}_i = c_i + \Delta\Phi(D_i), \text{ wobei } \Delta\Phi(D_i) = \Phi(D_i) - \Phi(D_{i-1})$$

$$\textstyle\sum_{i=1}^n \hat{c}_i = \sum_{i=1}^n c_i + \Phi(D_n) - \Phi(D_0) \geq \sum_{i=1}^n c_i$$



i	$\hat{c}_i$	c_i	$\Delta\Phi(D_i)$	$\Phi(D_i)$
0				0
1	3	1	2	2
2	3	2	1	3
3	3	3	0	3
4	3	1	2	5
5	3	5	-2	3
6	3	1	2	5

$$\sum_{i=1}^n c_i \leq \sum_{i=1}^n \hat{c}_i = 3n = \Theta(n)$$

Zusammenfassung

Zeige mit **amortisierter Analyse**, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Zusammenfassung

Zeige mit **amortisierter Analyse**, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Drei Typen von amortisierter Analyse:

- Aggregationsmethode
- Buchhaltermethode
- Potentialmethode

Zusammenfassung

Zeige mit **amortisierter Analyse**, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Drei Typen von amortisierter Analyse:

- Aggregationsmethode ✓

  Summiere tatsächliche Kosten (oder obere Schranken dafür) auf.

- Buchhaltermethode

- Potentialmethode

Zusammenfassung

Zeige mit **amortisierter Analyse**, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Drei Typen von amortisierter Analyse:

- Aggregationsmethode ✓

  Summiere tatsächliche Kosten (oder obere Schranken dafür) auf.

- Buchhaltermethode ✓

  Verbinde Extrakosten mit konkreten Objekten der Datenstruktur und bezahle damit teure Operationen.

- Potentialmethode

Zusammenfassung

Zeige mit **amortisierter Analyse**, dass die Operationen einer gegebenen Folge kleine durchschnittliche Kosten haben – *auch wenn einzelne Operationen in der Folge teuer sind!*

Drei Typen von amortisierter Analyse:

- Aggregationsmethode ✓

  Summiere tatsächliche Kosten (oder obere Schranken dafür) auf.

- Buchhaltermethode ✓

  Verbinde Extrakosten mit konkreten Objekten der Datenstruktur und bezahle damit teure Operationen.

- Potentialmethode ✓

  Definiere Potential der gesamten Datenstruktur, so dass mit der Potentialdifferenz teure Operationen bezahlt werden können.