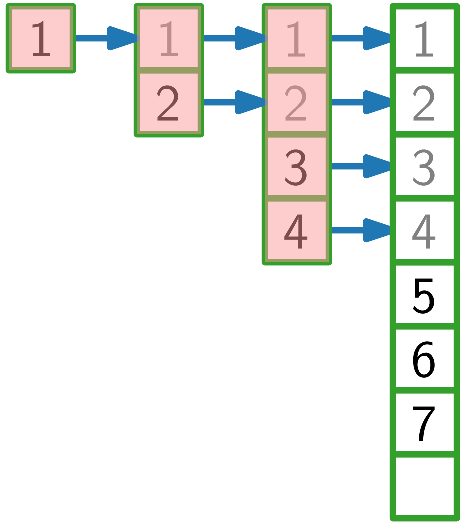
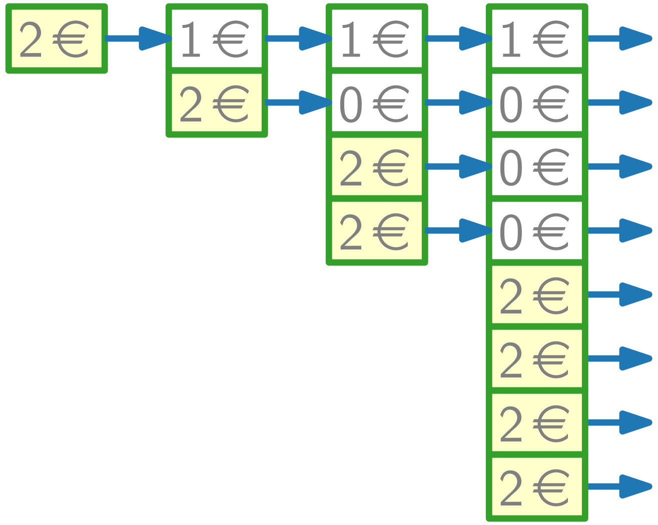
Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Algorithmen und Datenstrukturen



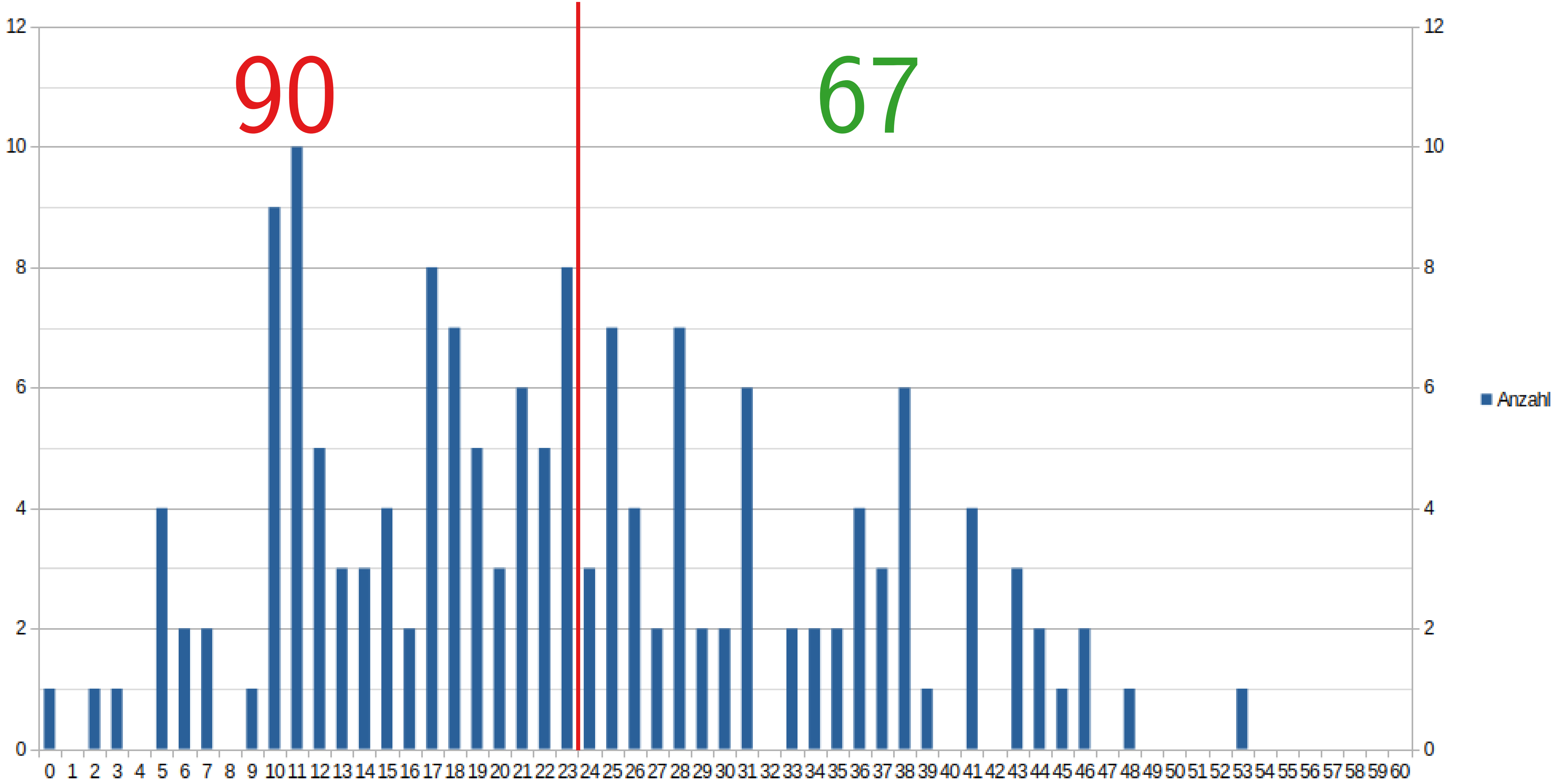
Vorlesung 16:
Amortisierte Analyse



Alexander Wolff

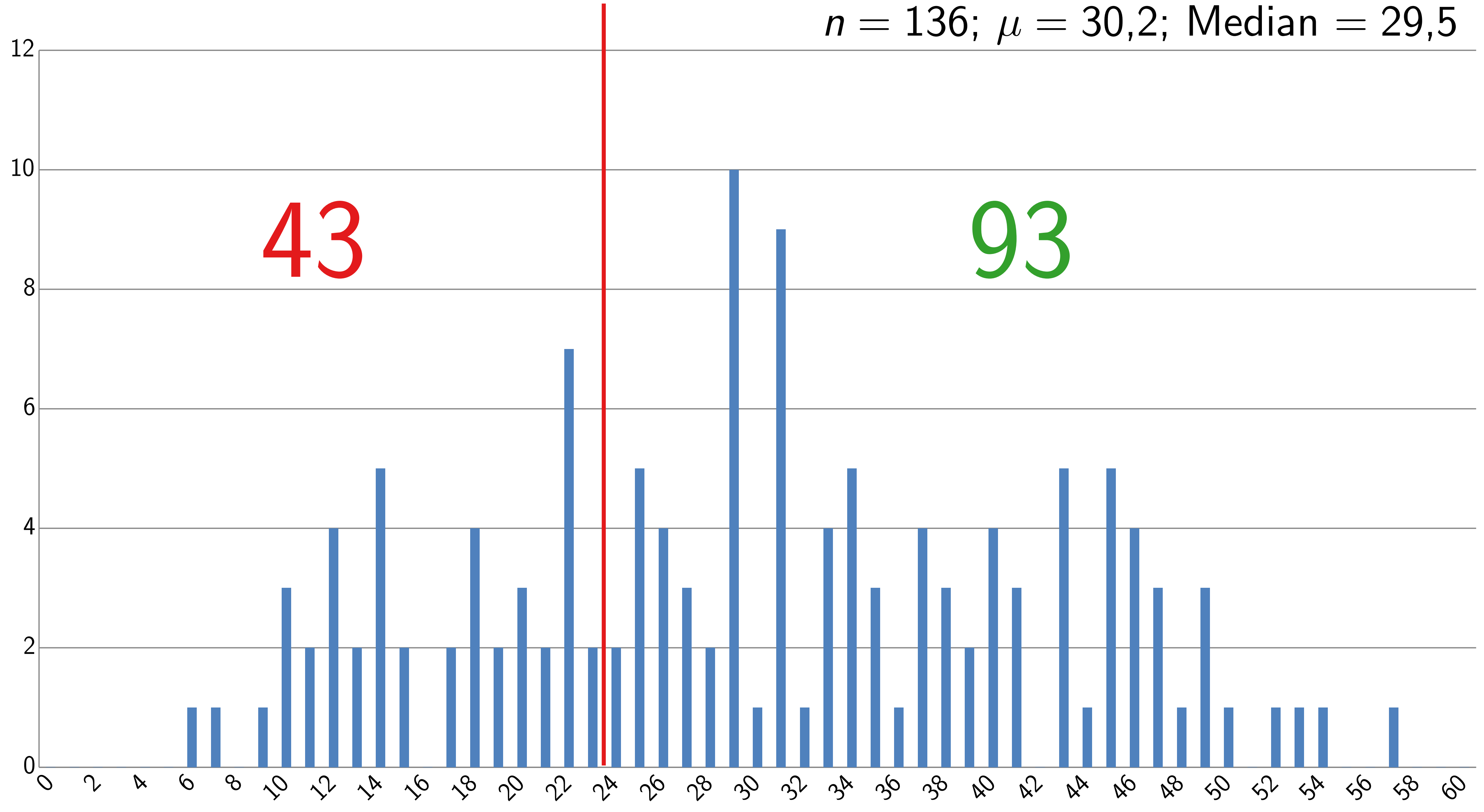
Wintersemester 2024

1. Zwischentest: Punkteverteilung

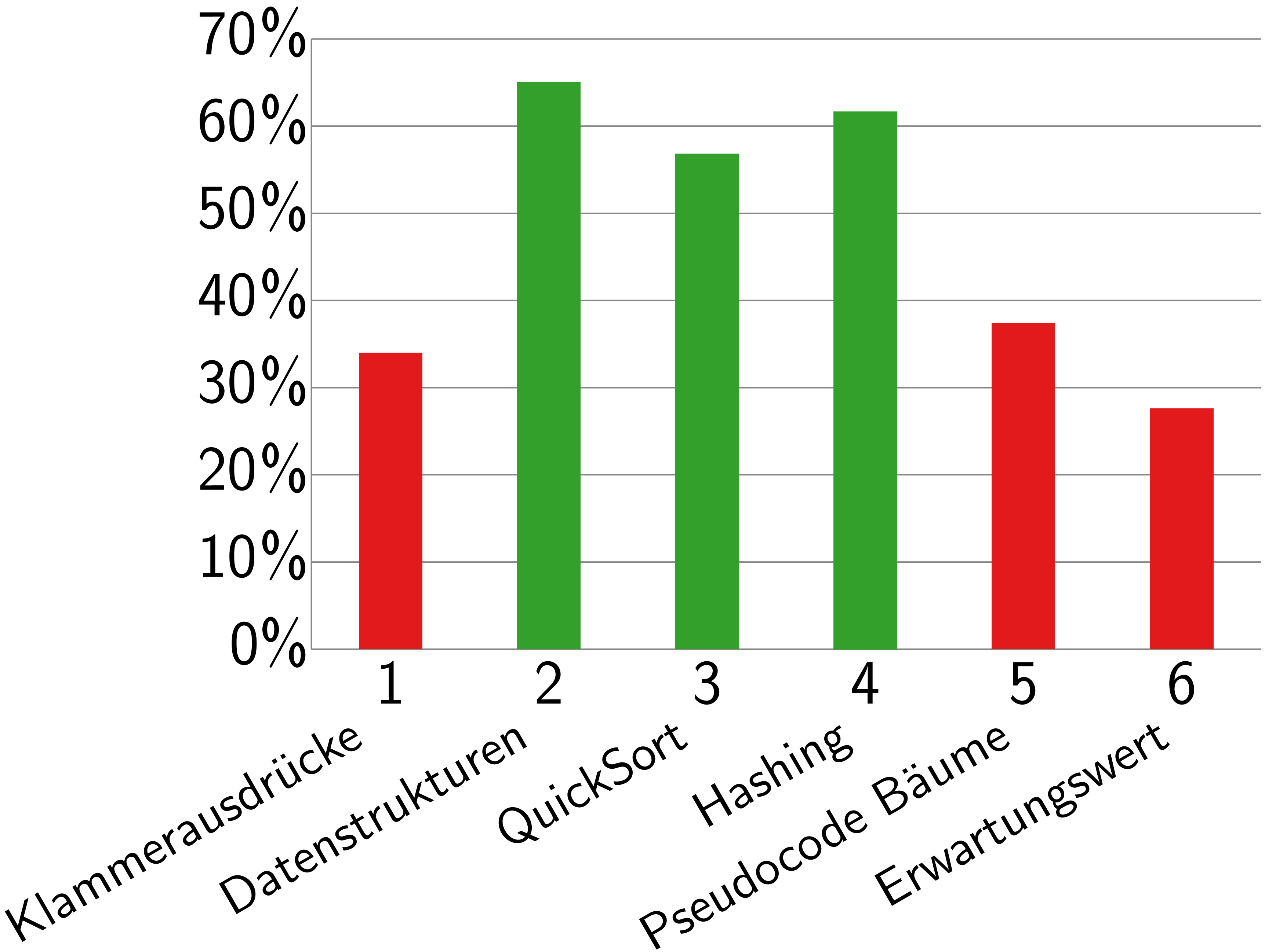


$n = 157$ arithmetisches Mittel $= 22{,}9$ Median $= 19{,}5$

2. Zwischentest: Punkteverteilung



2. Zwischentest: Aufgabenübersicht



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.

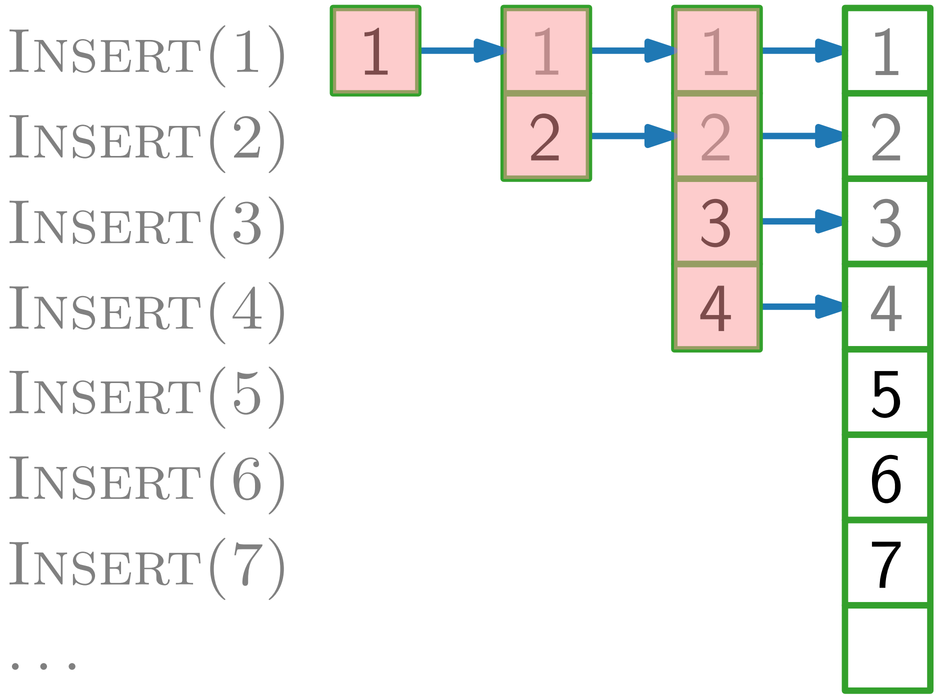
Problem: Was tun, wenn man die maximale Anzahl zu speichernder Elemente vorab nicht kennt?

Lösung: **Dynamische** Tabellen!

Dynamische Tabellen

Idee.

- 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.

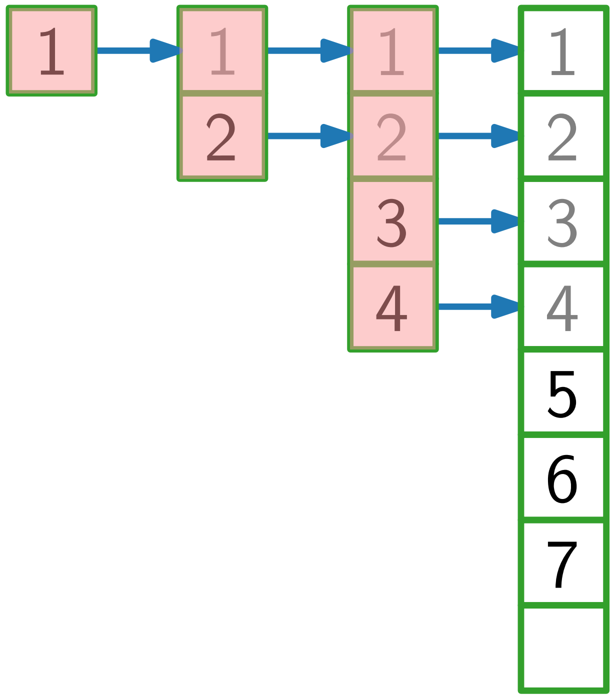
- Tabelle wird genau $\lceil \log_2 n \rceil$ mal kopiert.
- Im schlimmsten (letzten!) Fall ist der Aufwand $\Theta(n)$.

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

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

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

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 – *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

- 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!

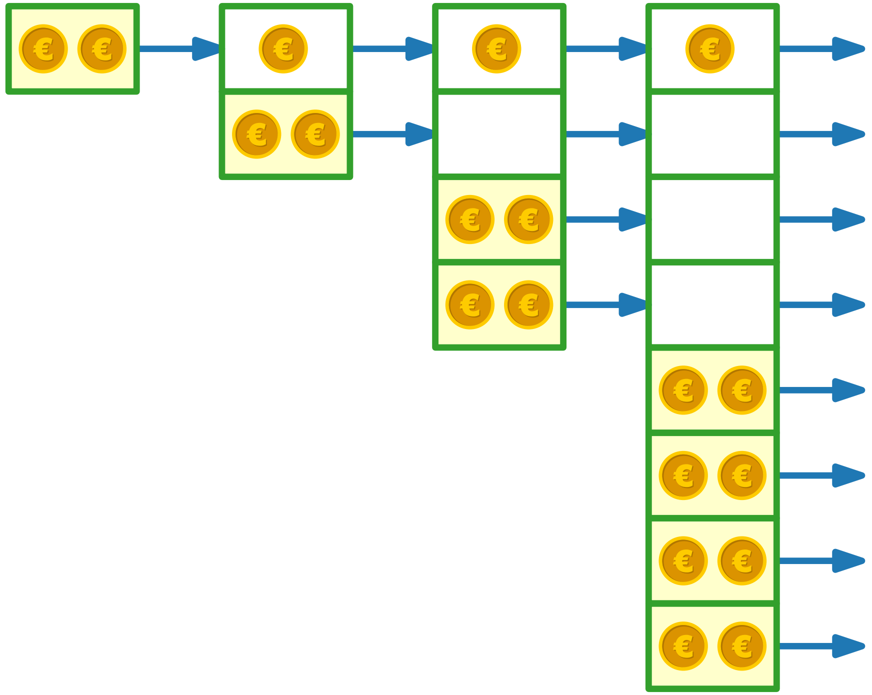
  Dann gilt $\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$.

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

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()	

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.

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.

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

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

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

Was sind die amort. Kosten?

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

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

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

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

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

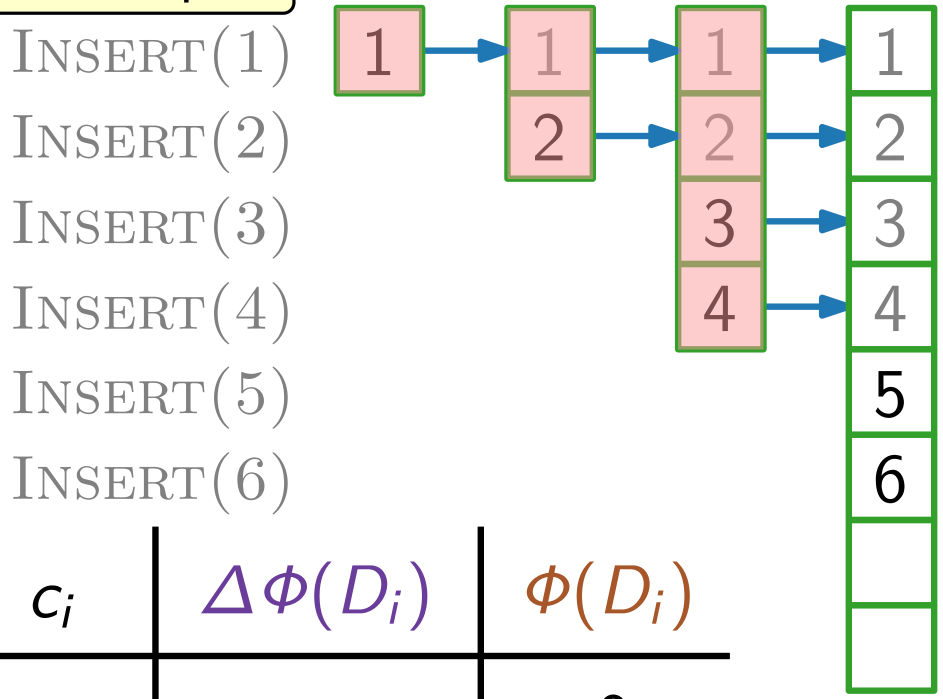
Also: **Amortisierte** Kosten pro Operation $\Theta(\boxed{1})$.
$\Rightarrow$ **Tatsächliche** Kosten für n Operationen im worst case $\Theta(n)$.

Potentialmethode für dynamische Tabellen

Idee.

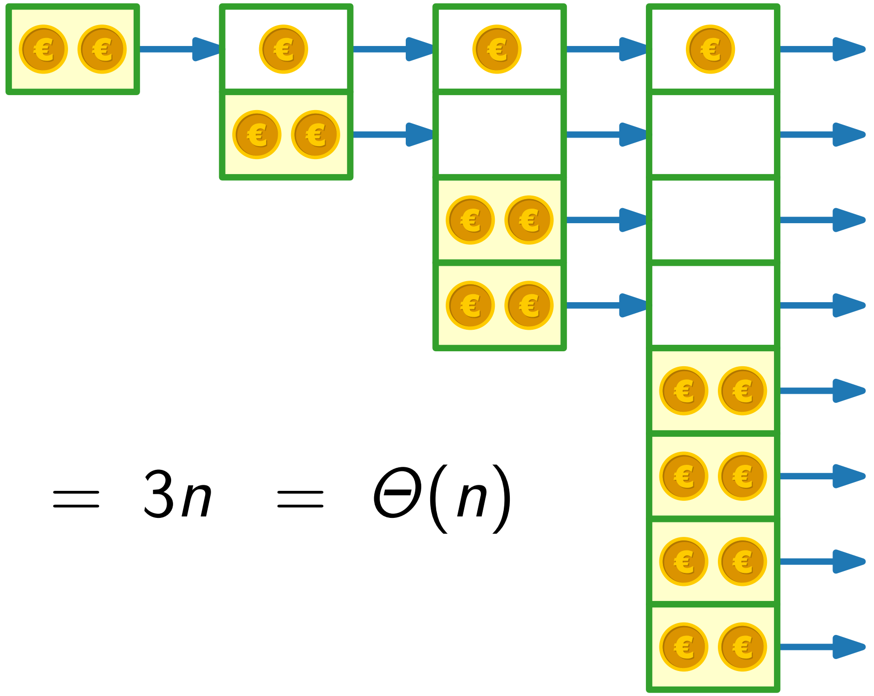
- Kein Kopieren $\Rightarrow \Delta\Phi(D_i) = 2$
- Kopieren $\Rightarrow \Delta\Phi(D_i) = 2 - (i-1)$
- $\Rightarrow \Phi(D_i) = 1 + 2 \cdot \mathit{size}_i - \textit{table-size}_i$

$i-1$ Elemente werden kopiert



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

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



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!*

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.