
Algorithmen und Datenstrukturen

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Tanya Braun (Übungen)

sowie viele Tutoren

Sortierung in linearer Zeit

- Sortieren: Geht es doch noch schneller als in $\Omega(n \log n)$ Schritten?
- Man muss „schärfere“ Annahmen über das Problem machen können ...
 - z.B. Schlüssel in n Feldelementen aus dem Bereich $[1..n]$
- ... oder Nebenbedingungen „abschwächen“
 - z.B. die In-situ-Einschränkung aufgeben
- Zentrale Idee: Vermeide Vergleiche!

Seward, H. H. (1954), "2.4.6 Internal Sorting by Floating Digital Sort", Information sorting in the application of electronic digital computers to business operations, Master's thesis, Report R-232, Massachusetts Institute of Technology, Digital Computer Laboratory, pp. 25–28

A. Andersson, T. Hagerup, S. Nilsson, R. Raman, Sorting in Linear Time?, J. Comput. Syst. Sci. 57(1): 74–93, 1998

Danksagung

- Nachfolgende Präsentationen sind inspiriert durch CS 3343/3341 Analysis of Algorithms 2013
- http://www.cs.utsa.edu/~jruan/teaching/cs3343_spring_2013/index.html

Sortieren durch Zählen / Counting-Sort

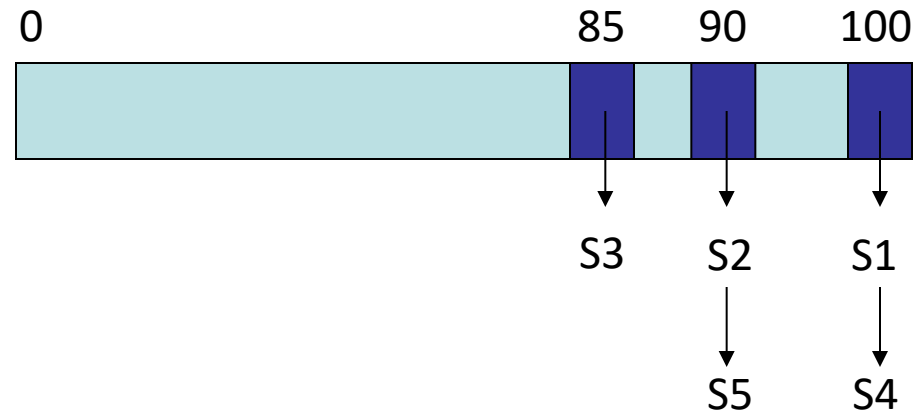
- **Wissen:**
Schlüssel fallen in einen kleinen Zahlenbereich
- **Beispiel 1:** Sortiere eine Menge von Studierenden nach Examensbewertungen (Scores sind Zahlen)
 - 1000 Studenten
 - Maximum score: 100
 - Minimum score: 0
- **Beispiel 2:** Sortiere Studierende nach dem ersten Buchstaben des Nachnamens
 - Anzahl der Studierenden: viele
 - Anzahl der Buchstaben: 26

Counting-Sort

- **Eingabe:** $A[1 \dots n]$, wobei $A[j] \in \{1, 2, \dots, k\}$.
 - **Ausgabe:** $B[1 \dots n]$, sortiert.
 - **Hilfsspeicher:** $C[1 \dots k]$.
-
- Kein In-situ-Sortieralgorithmus
 - Benötigt $\Theta(n+k)$ zusätzliche Speicherplätze

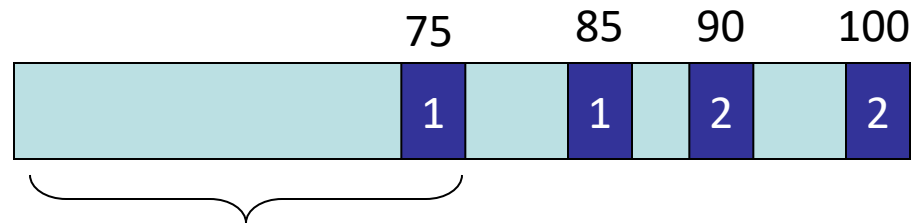
Intuition

- S1: 100
- S2: 90
- S3: 85
- S4: 100
- S5: 90
- ...



... S3 ... S2, S5, ..., S1, S4

Intuition



50 Studierende mit Score ≤ 75

Was ist der Rang (von klein auf groß) für einen Studenten mit Score 75?

50

200 Studierende mit Score ≤ 90

Was ist der Rang für einen Studenten mit Score 90?

200 or 199

Counting-Sort

1.for $i \leftarrow 1$ to k
do $C[i] \leftarrow 0$

Initialisiere

2.for $j \leftarrow 1$ to n
do $C[A[j]] \leftarrow C[A[j]] + 1$

Zähle

▷ $C[i] = |\{\text{key} = i\}|$

3.for $i \leftarrow 2$ to k
do $C[i] \leftarrow C[i] + C[i-1]$

Bestimme Summe

▷ $C[i] = |\{\text{key} \leq i\}|$

4.for $j \leftarrow n$ downto 1
do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

Ordne neu

Counting-Sort Beispiel

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :				

<i>B</i> :					
------------	--	--	--	--	--

Schleife 1: Initialisierung

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	0	0	0	0

<i>B</i> :					
------------	--	--	--	--	--

1. **for** $i \leftarrow 1$ **to** k
 do $C[i] \leftarrow 0$

Schleife 2: Zähle

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	0	0	0	1

<i>B</i> :					
------------	--	--	--	--	--

2. **for** $j \leftarrow 1$ **to** n
 do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright C[i] = |\{\text{key} = i\}|$

Schleife 2: Zähle

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	1	0	0	1

<i>B</i> :					
------------	--	--	--	--	--

2. for $j \leftarrow 1$ **to** n
 do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright C[i] = |\{\text{key} = i\}|$

Schleife 2: Zähle

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	1	0	1	1

<i>B</i> :					
------------	--	--	--	--	--

2. **for** $j \leftarrow 1$ **to** n
 do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright C[i] = |\{\text{key} = i\}|$

Schleife 2: Zähle

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	1	0	1	2

<i>B</i> :					
------------	--	--	--	--	--

2. **for** $j \leftarrow 1$ **to** n
 do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright C[i] = |\{\text{key} = i\}|$

Schleife 2: Zähle

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	1	0	2	2

<i>B</i> :					
------------	--	--	--	--	--

2. **for** $j \leftarrow 1$ **to** n
 do $C[A[j]] \leftarrow C[A[j]] + 1$ $\triangleright C[i] = |\{\text{key} = i\}|$

Schleife 3: Berechne Summe

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	1	0	2	2

<i>B</i> :					
------------	--	--	--	--	--

<i>C'</i> :	1	1	2	2
-------------	---	---	---	---

3. for $i \leftarrow 2$ to k

do $C[i] \leftarrow C[i] + C[i-1]$

$\triangleright C[i] = |\{\text{key} \leq i\}|$

Schleife 3: Berechne Summe

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	1	0	2	2

<i>B</i> :					
------------	--	--	--	--	--

<i>C'</i> :	1	1	3	2
-------------	---	---	---	---

3. for $i \leftarrow 2$ to k

do $C[i] \leftarrow C[i] + C[i-1]$

$\triangleright C[i] = |\{\text{key} \leq i\}|$

Schleife 3: Berechne Summe

	1	2	3	4	5
<i>A</i> :	4	1	3	4	3

	1	2	3	4
<i>C</i> :	1	0	2	2

<i>B</i> :					
------------	--	--	--	--	--

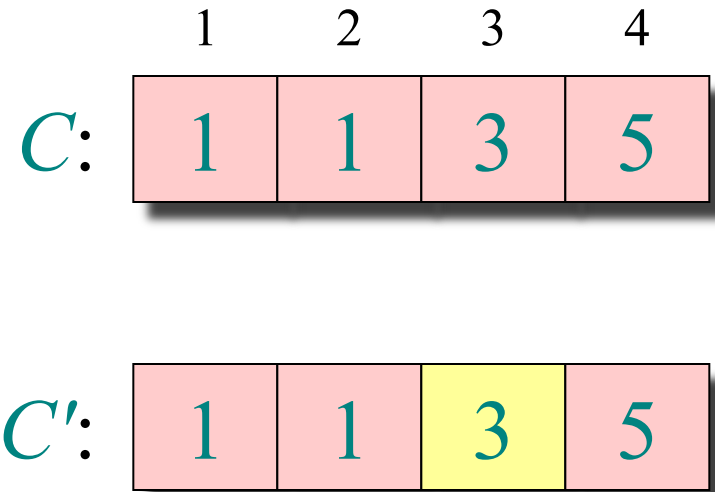
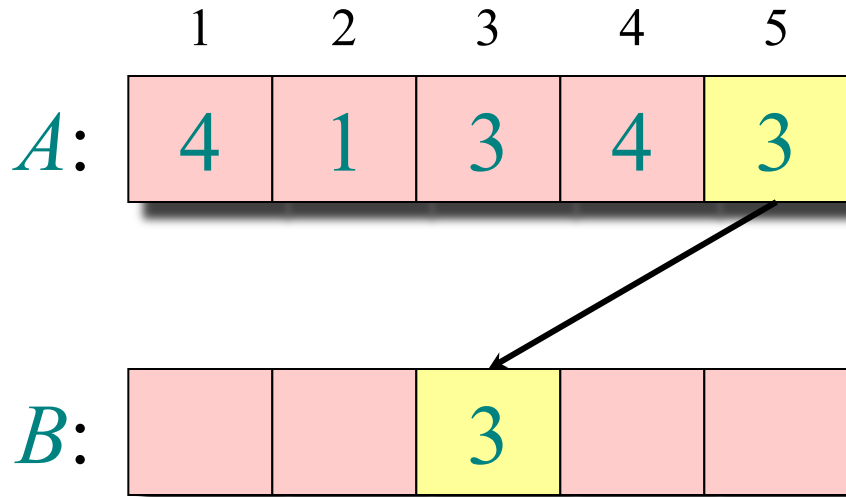
<i>C'</i> :	1	1	3	5
-------------	---	---	---	---

3. for $i \leftarrow 2$ to k

do $C[i] \leftarrow C[i] + C[i-1]$

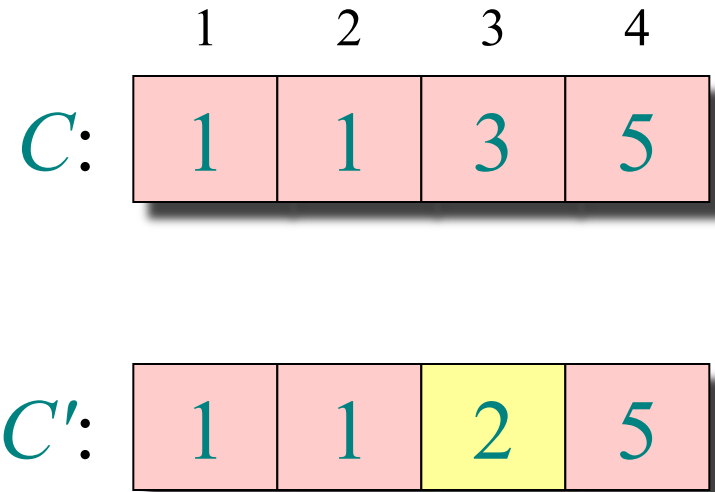
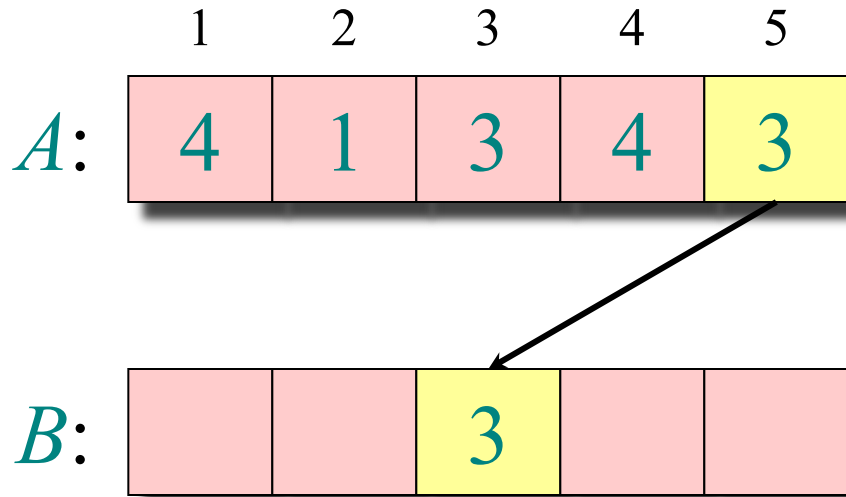
$\triangleright C[i] = |\{\text{key} \leq i\}|$

Schleife 4: Ordne neu



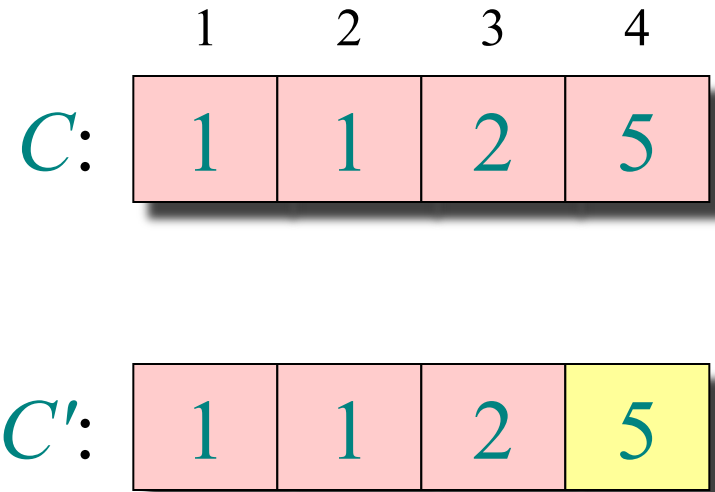
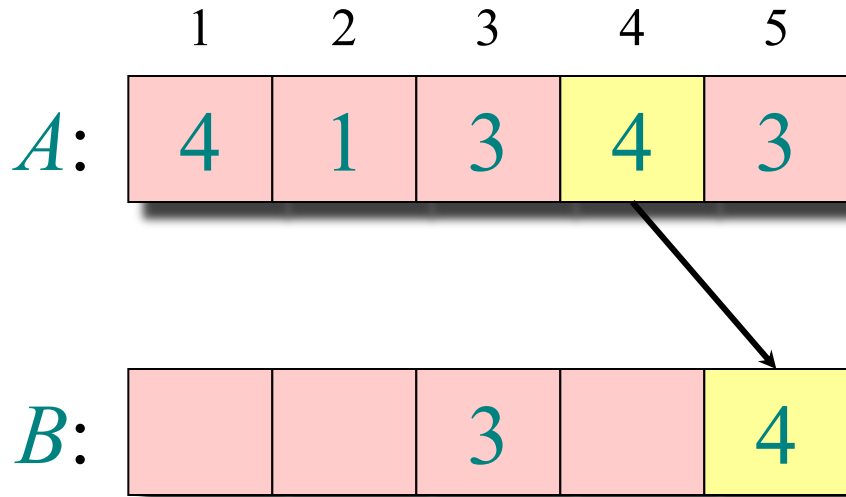
4. for $j \leftarrow n$ downto 1
 do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu



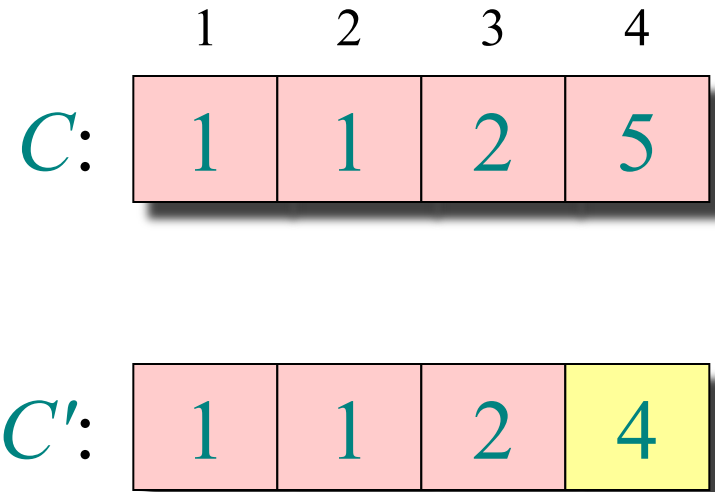
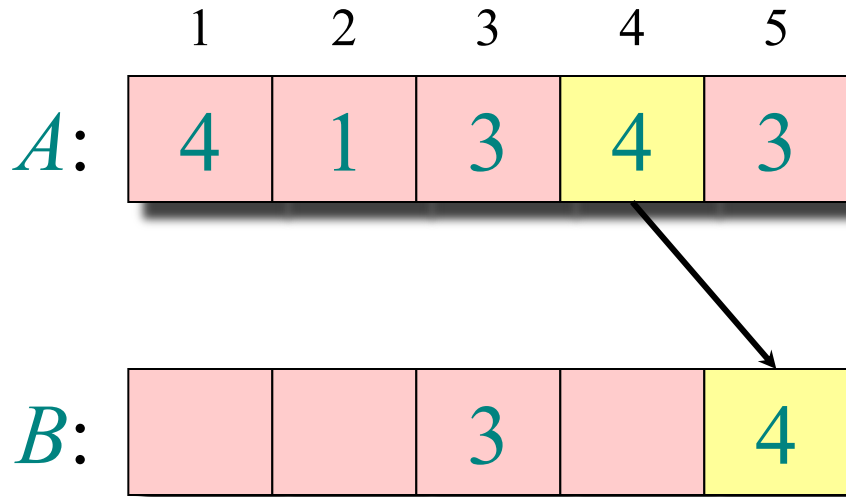
4. for $j \leftarrow n$ downto 1
 do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu



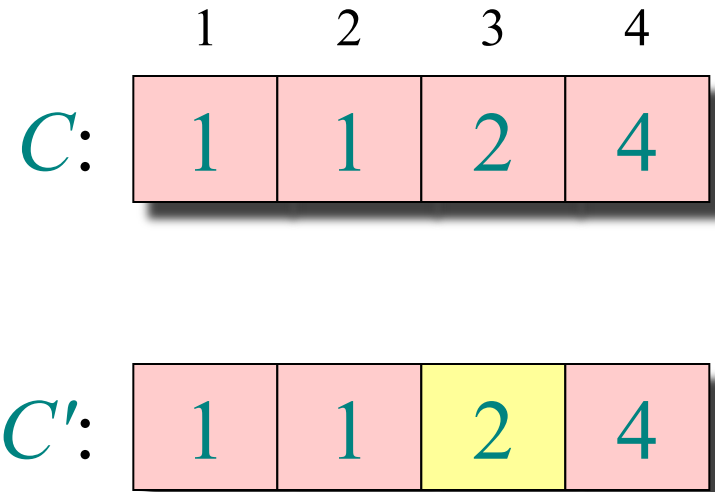
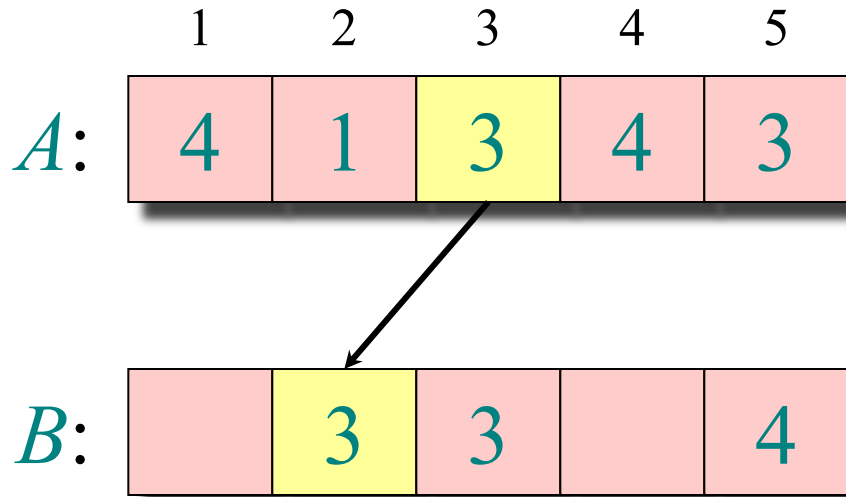
```
4. for  $j \leftarrow n$  downto 1
    do  $B[C[A[j]]] \leftarrow A[j]$ 
        $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

Schleife 4: Ordne neu



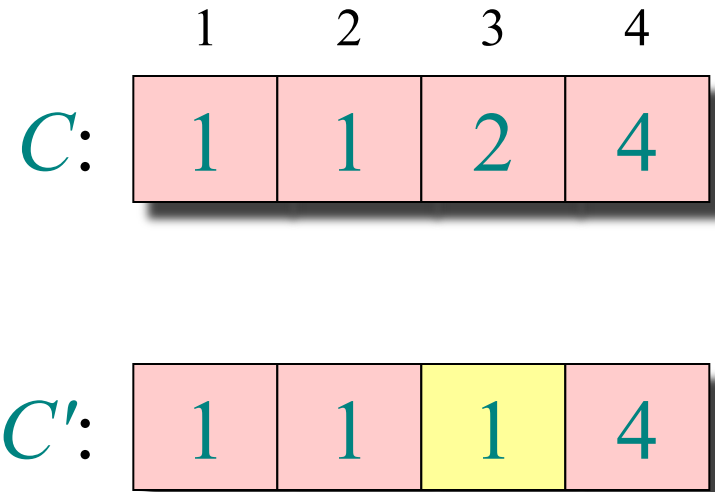
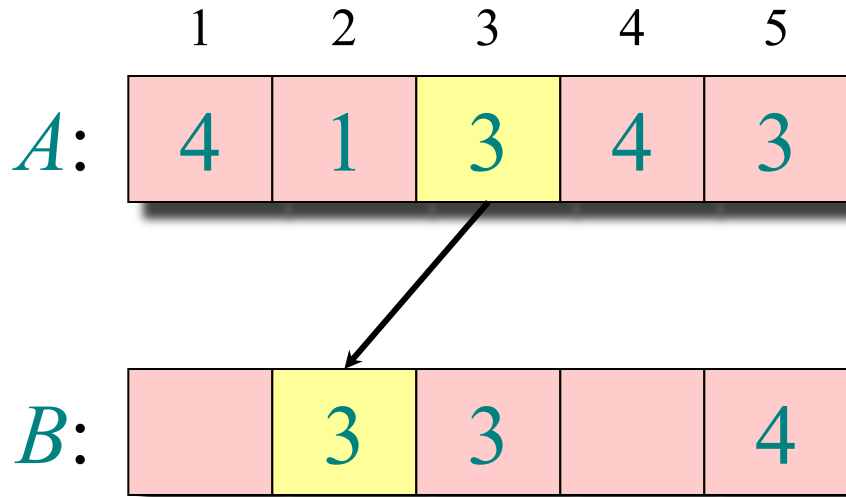
```
4. for  $j \leftarrow n$  downto 1
    do  $B[C[A[j]]] \leftarrow A[j]$ 
        $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

Schleife 4: Ordne neu



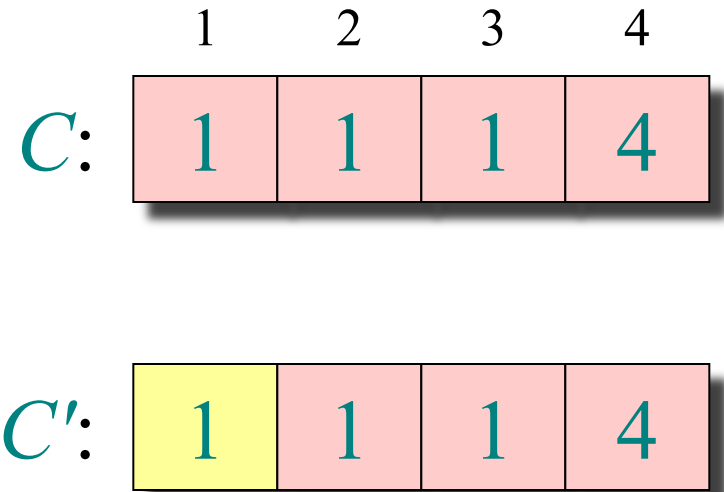
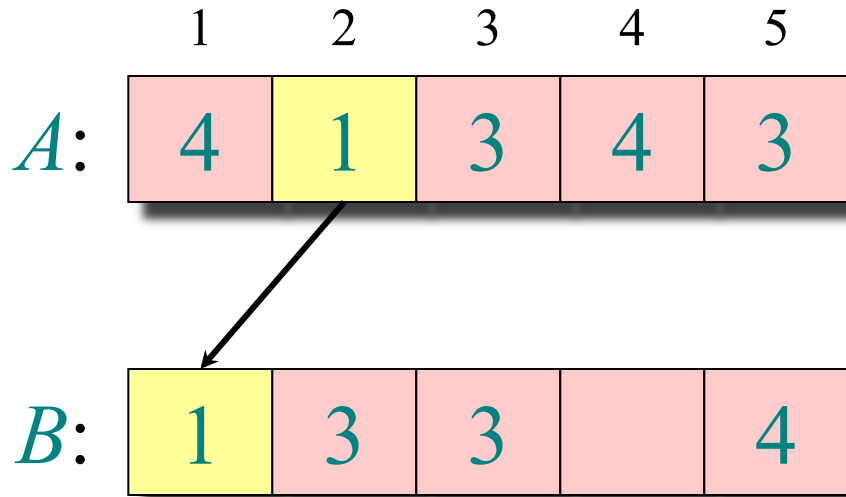
4. for $j \leftarrow n$ downto 1
 do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu



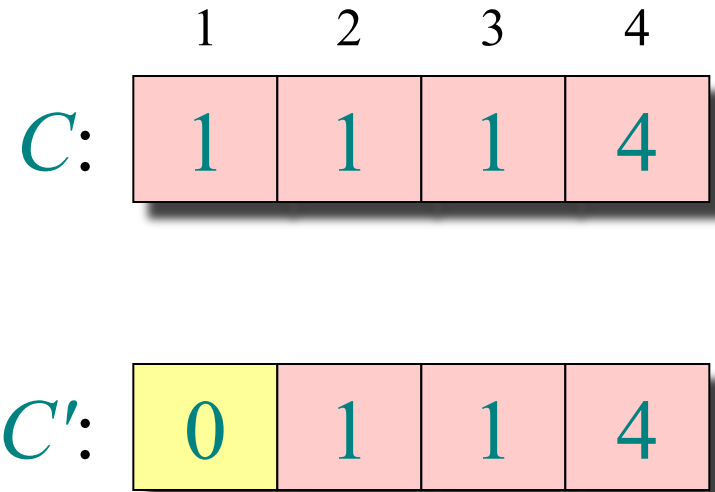
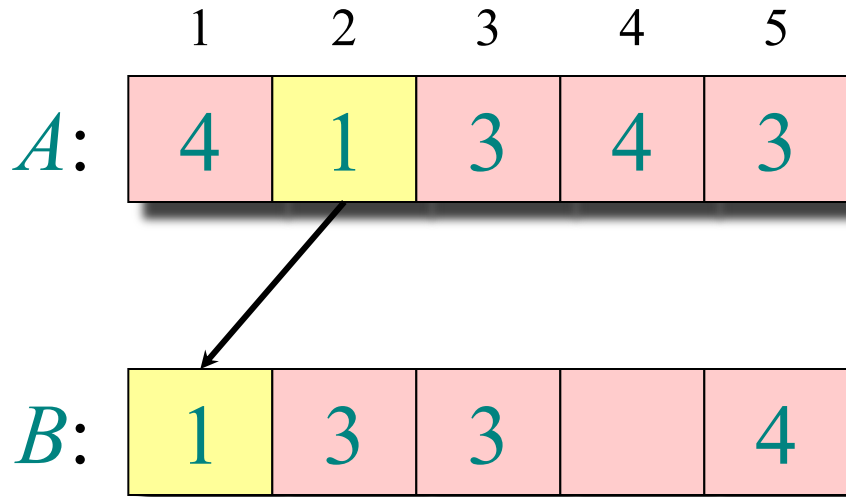
```
4. for  $j \leftarrow n$  downto 1
    do  $B[C[A[j]]] \leftarrow A[j]$ 
        $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

Schleife 4: Ordne neu



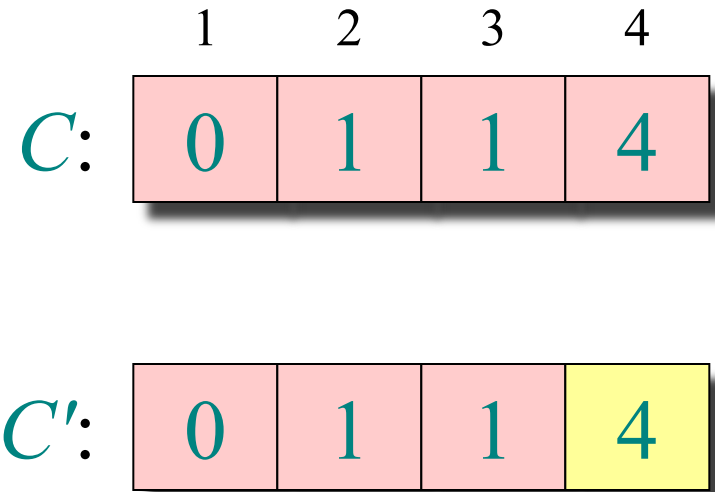
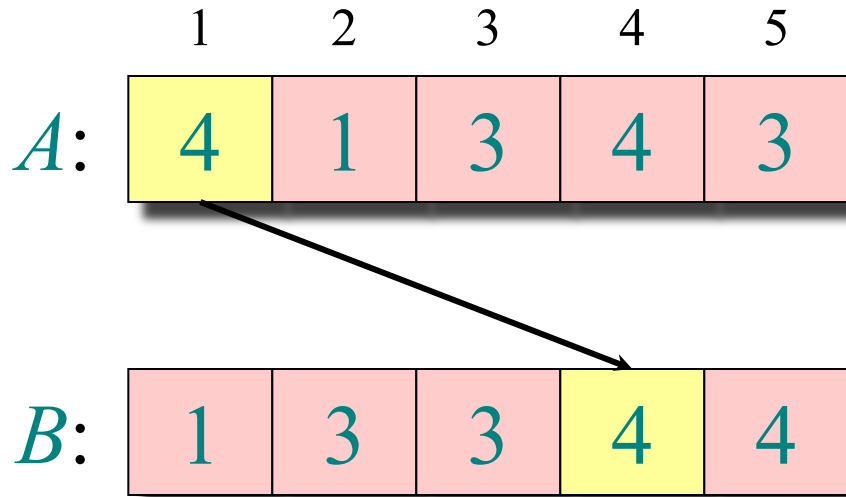
4. for $j \leftarrow n$ downto 1
 do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu



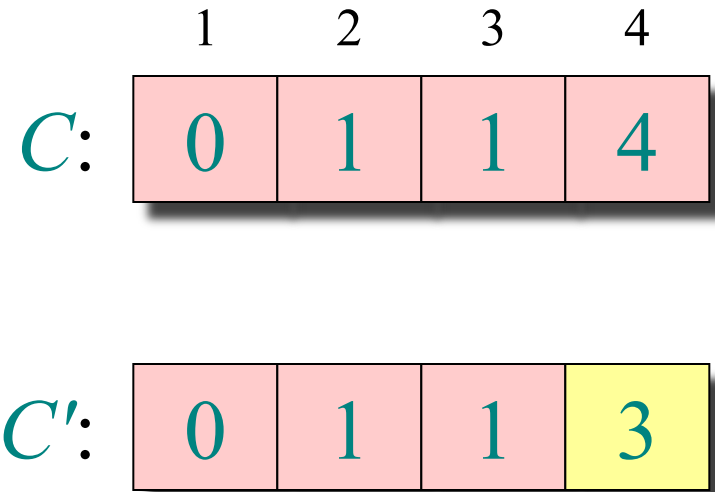
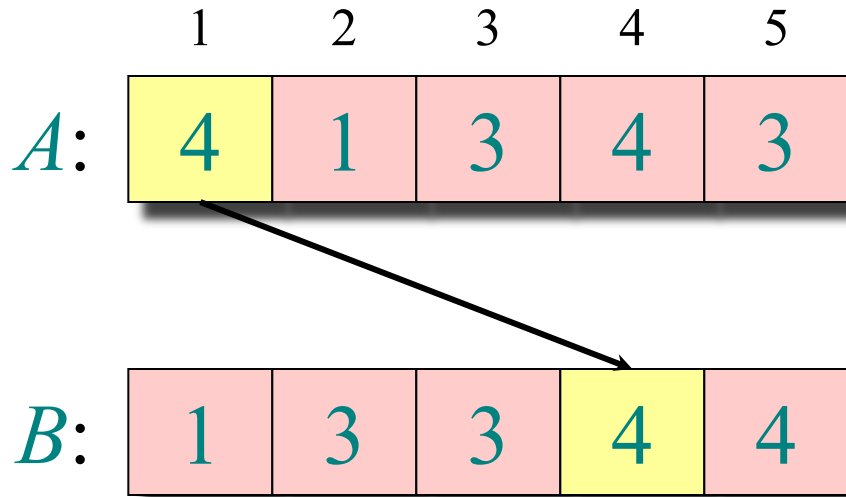
4. for $j \leftarrow n$ downto 1
 do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu



4. for $j \leftarrow n$ downto 1
 do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu



4. for $j \leftarrow n$ downto 1
 do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

Counting-Sort Algorithmus

```
1: procedure COUNTING-SORT( $A, B, k$ )
2:   for  $i \leftarrow 1$  to  $k$  do
3:      $C[i] \leftarrow 0$ 
4:   for  $j \leftarrow 1$  to  $\text{length}(A)$  do
5:      $C[A[j]] \leftarrow C[A[j]] + 1$ 
6:    $\triangleright C[i]$  enthält nun die Anzahl der Elemente, die gleich  $i$  sind.
7:   for  $i \leftarrow 2$  to  $k$  do
8:      $C[i] \leftarrow C[i] + C[i - 1]$ 
9:    $\triangleright C[i]$  enthält nun die Anzahl der Elemente, die kleiner oder gleich  $i$  sind.
10:  for  $j \leftarrow \text{length}(A)$  downto 1 do
11:     $B[C[A[j]]] \leftarrow A[j]$ 
12:     $C[A[j]] \leftarrow C[A[j]] - 1$ 
```

Analyse

$\Theta(k)$ { **1.**for $i \leftarrow 1$ to k
do $C[i] \leftarrow 0$

$\Theta(n)$ { **2.**for $j \leftarrow 1$ to n
do $C[A[j]] \leftarrow C[A[j]] + 1$

$\Theta(k)$ { **3.**for $i \leftarrow 2$ to k
do $C[i] \leftarrow C[i] + C[i-1]$

$\Theta(n)$ { **4.**for $j \leftarrow n$ downto 1
do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

$\Theta(n + k)$

Laufzeit: Wodurch wird sie reduziert?

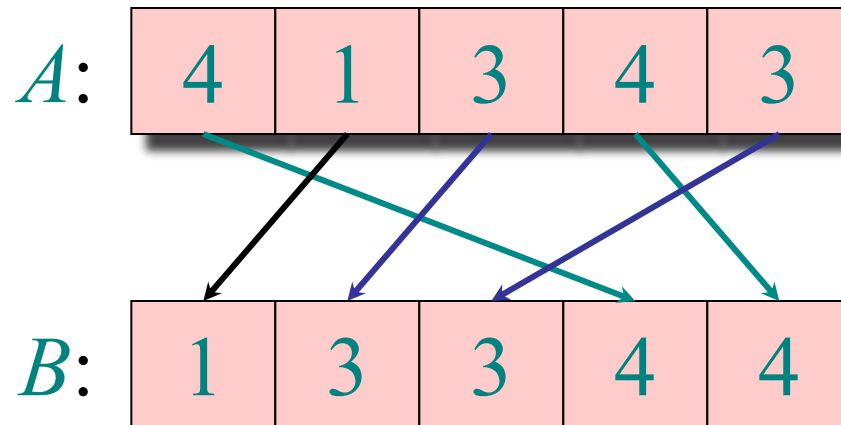
- Falls $k = O(n)$, dann braucht Counting-Sort $\Theta(n)$ viele Schritte.
- Aber theoretisch braucht doch Sortierung $\Omega(n \log n)$ viele Schritte!
- Gibt es ein Problem mit der Theorie?

Antwort:

- *Sortieren durch Vergleichen* liegt in $\Omega(n \log n)$
- Counting-Sort macht keine Vergleiche
- Counting-Sort verteilt einfach

Stabiles Sortieren

Counting-Sort ist **stabil**: die Eingabeordnung für gleiche Schlüssel bleibt bestehen



Warum ist das wichtig?

Welche andere Algorithmen haben diese Eigenschaft?

Name		Address		
Last	First	Street	City	State
Bayless	Andrew	West Ave	Houston	TX
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Mirabal	Renato	FIRST ST	Columbus	OH
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Moon	Ryan	EAST AVE.	Madison	WI

Name		Address		
Last	First	Street	City	State
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Bayless	Andrew	West Ave	Houston	TX
Devineni	Soujanya	Northwestern Ave	Houston	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Moon	Ryan	EAST AVE.	Madison	WI

Name		Address		
Last	First	Street	City	State
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Dunne	Brendan	EAST AVE.	Dallas	TX
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Bayless	Andrew	West Ave	Houston	TX
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Benitez	Michael	North Ave	Los Angeles	CA
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Moon	Ryan	EAST AVE.	Madison	WI
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Edwards	Brian	De Zavala Rd	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA

Name		Address		
Last	First	Street	City	State
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Guerra	John	DALLAS AVE.	Austin	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA
Dunne	Brendan	EAST AVE.	Dallas	TX
Moon	Ryan	EAST AVE.	Madison	WI
Mirabal	Renato	FIRST ST	Columbus	OH
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Godfrey	Daryl	MAIN ST	Austin	TX
Benitez	Michael	North Ave	Los Angeles	CA
Devineni	Soujanya	Northwestern Ave	Houston	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Dangelo	David	Third St	Detroit	MI
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Bayless	Andrew	West Ave	Houston	TX

Name		Address		
Last	First	Street	City	State
Bayless	Andrew	West Ave	Houston	TX
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Mirabal	Renato	FIRST ST	Columbus	OH
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Moon	Ryan	EAST AVE.	Madison	WI

Name		Address		
Last	First	Street	City	State
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Guerra	John	DALLAS AVE.	Austin	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA
Dunne	Brendan	EAST AVE.	Dallas	TX
Moon	Ryan	EAST AVE.	Madison	WI
Mirabal	Renato	FIRST ST	Columbus	OH
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Godfrey	Daryl	MAIN ST	Austin	TX
Benitez	Michael	North Ave	Los Angeles	CA
Devineni	Soujanya	Northwestern Ave	Houston	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Dangelo	David	Third St	Detroit	MI
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Bayless	Andrew	West Ave	Houston	TX

Name		Address		
Last	First	Street	City	State
Guerra	John	DALLAS AVE.	Austin	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Dunne	Brendan	EAST AVE.	Dallas	TX
Dawood	Hussam	Lincoln Rd	Detroit	MI
Dangelo	David	Third St	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Bayless	Andrew	West Ave	Houston	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Benitez	Michael	North Ave	Los Angeles	CA
Moon	Ryan	EAST AVE.	Madison	WI
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA

Name		Address			
Last	First	Street	City	State	
Martinez	Juan	OAK CLIFF	Pheonix	AZ	
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA	
Benitez	Michael	North Ave	Los Angeles	CA	
Chu	Henry	East Ave	San Diego	CA	
Dawood	Hussam	Lincoln Rd	Detroit	MI	
Dangelo	David	Third St	Detroit	MI	
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH	
Mirabal	Renato	FIRST ST	Columbus	OH	
Guerra	John	DALLAS AVE.	Austin	TX	
Godfrey	Daryl	MAIN ST	Austin	TX	
Dunne	Brendan	EAST AVE.	Dallas	TX	
Devineni	Soujanya	Northwestern Ave	Houston	TX	
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX	
Bayless	Andrew	West Ave	Houston	TX	
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX	
Edwards	Brian	De Zavala Rd	San Antonio	TX	
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX	
Honeycutt	Richard	Southwest Ave	San Antonio	TX	
Guevara	Clovis	University Pkwy	San Antonio	TX	
Mayo	Nathan	UTSA BLVD	San Antonio	TX	
Moon	Ryan	EAST AVE.	Madison	WI	

Name		Address			
Last	First	Street	City	State	
Martinez	Juan	OAK CLIFF	Pheonix	AZ	
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA	
Benitez	Michael	North Ave	Los Angeles	CA	
Chu	Henry	East Ave	San Diego	CA	
Dawood	Hussam	Lincoln Rd	Detroit	MI	
Dangelo	David	Third St	Detroit	MI	
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH	
Mirabal	Renato	FIRST ST	Columbus	OH	
Guerra	John	DALLAS AVE.	Austin	TX	
Godfrey	Daryl	MAIN ST	Austin	TX	
Dunne	Brendan	EAST AVE.	Dallas	TX	
Devineni	Soujanya	Northwestern Ave	Houston	TX	
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX	
Bayless	Andrew	West Ave	Houston	TX	
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX	
Edwards	Brian	De Zavala Rd	San Antonio	TX	
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX	
Honeycutt	Richard	Southwest Ave	San Antonio	TX	
Guevara	Clovis	University Pkwy	San Antonio	TX	
Mayo	Nathan	UTSA BLVD	San Antonio	TX	
Moon	Ryan	EAST AVE.	Madison	WI	

Name		Address		
Last	First	Street	City	State
Bayless	Andrew	West Ave	Houston	TX
Benitez	Michael	North Ave	Los Angeles	CA
Chu	Henry	East Ave	San Diego	CA
Dangelo	David	Third St	Detroit	MI
Dawood	Hussam	Lincoln Rd	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Dunne	Brendan	EAST AVE.	Dallas	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Guerra	John	DALLAS AVE.	Austin	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Mirabal	Renato	FIRST ST	Columbus	OH
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Moon	Ryan	EAST AVE.	Madison	WI

Name		Address		
Last	First	Street	City	State
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Guerra	John	DALLAS AVE.	Austin	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA
Dunne	Brendan	EAST AVE.	Dallas	TX
Moon	Ryan	EAST AVE.	Madison	WI
Mirabal	Renato	FIRST ST	Columbus	OH
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Dawood	Hussam	Lincoln Rd	Detroit	MI
Godfrey	Daryl	MAIN ST	Austin	TX
Benitez	Michael	North Ave	Los Angeles	CA
Devineni	Soujanya	Northwestern Ave	Houston	TX
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Dangelo	David	Third St	Detroit	MI
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Bayless	Andrew	West Ave	Houston	TX

Name		Address		
Last	First	Street	City	State
Guerra	John	DALLAS AVE.	Austin	TX
Godfrey	Daryl	MAIN ST	Austin	TX
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH
Mirabal	Renato	FIRST ST	Columbus	OH
Dunne	Brendan	EAST AVE.	Dallas	TX
Dawood	Hussam	Lincoln Rd	Detroit	MI
Dangelo	David	Third St	Detroit	MI
Devineni	Soujanya	Northwestern Ave	Houston	TX
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX
Bayless	Andrew	West Ave	Houston	TX
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA
Benitez	Michael	North Ave	Los Angeles	CA
Moon	Ryan	EAST AVE.	Madison	WI
Martinez	Juan	OAK CLIFF	Pheonix	AZ
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX
Edwards	Brian	De Zavala Rd	San Antonio	TX
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX
Honeycutt	Richard	Southwest Ave	San Antonio	TX
Guevara	Clovis	University Pkwy	San Antonio	TX
Mayo	Nathan	UTSA BLVD	San Antonio	TX
Chu	Henry	East Ave	San Diego	CA

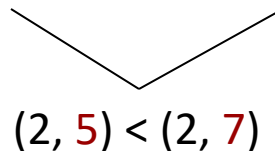
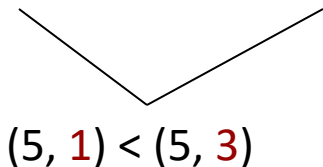
Name		Address			
Last	First	Street	City	State	
Martinez	Juan	OAK CLIFF	Pheonix	AZ	
Halbeisen	Gerald	FOREST CIRCLE	Los Angeles	CA	
Benitez	Michael	North Ave	Los Angeles	CA	
Chu	Henry	East Ave	San Diego	CA	
Dawood	Hussam	Lincoln Rd	Detroit	MI	
Dangelo	David	Third St	Detroit	MI	
Hohmann	Shawn	COLLEGE PKWY	Cleveland	OH	
Mirabal	Renato	FIRST ST	Columbus	OH	
Guerra	John	DALLAS AVE.	Austin	TX	
Godfrey	Daryl	MAIN ST	Austin	TX	
Dunne	Brendan	EAST AVE.	Dallas	TX	
Devineni	Soujanya	Northwestern Ave	Houston	TX	
Modebe	Nnaemeka	RIVERSIDE ST	Houston	TX	
Bayless	Andrew	West Ave	Houston	TX	
Hernandez	Monica	COLLEGE PKWY	San Antonio	TX	
Edwards	Brian	De Zavala Rd	San Antonio	TX	
Hawkins	Richard	RIVERSIDE ST	San Antonio	TX	
Honeycutt	Richard	Southwest Ave	San Antonio	TX	
Guevara	Clovis	University Pkwy	San Antonio	TX	
Mayo	Nathan	UTSA BLVD	San Antonio	TX	
Moon	Ryan	EAST AVE.	Madison	WI	

Stabiles Sortieren

- Die meisten $\Theta(n^2)$ -Sortialgorithmen sind stabil
 - oder können “einfach” stabil gemacht werden
- Die meisten $\Theta(n \log n)$ -Sortialgorithmen sind nicht stabil
 - Ausnahme: Merge-Sort
- Generischer Ansatz, Stabilität zu erzeugen:
 - Verwende zwei Schlüssel, der zweite ist der originale Index des Elements
 - Wenn zwei Elemente gleich, vergleiche zweite Schlüsselkomponenten

[5, 6, 5, 1, 2, 3, 2, 6]

[(5, 1), (6, 2), (5, 3), (1, 4), (2, 5), (3, 6), (2, 7), (6, 8)]



Wie kann man sehr große Zahlen sortieren?

198099109123518183599
340199540380128115295
384700101594539614696
382408360201039258538
614386507628681328936
738148652090990369197
987084087096653020299
185664124421234516454
785392075747859131885
530995223593137397354
267057490443618111767
795293581914837377527
815501764221221110674
142522204403312937607
718098797338329180836
856504702326654684056
982119770959427525245
528076153239047050820
305445639847201611168
478334240651199238019

Jede Zeile sei eine
"lange" Zahl, eine
Gensequenz oder....

Zahlen dieser Art sind zu groß für den Integer-Datentyp,
sie werden als "Zeichenkette" repräsentiert

Verwende vergleichsbasiertes Sortieren mit einer
Zeichenkettenvergleichsfunktion

if $A[i] < A[j]$ wird zu if $\text{vergleiche}(A[i], A[j]) < 0$ mit

```
function vergleiche(s, t)
  for i = 1 to max( { length(s), length(t) } ) do
    if s[i] < t[i] then
      return -1
    else if s[i] > t[i] then
      return 1
  return 0
```

Was sind die Kosten des Vergleichs von zwei Zeichenketten
der Länge d? $O(d)$

Gesamtkosten: $O(d \cdot n \log n)$

Radix-Sort

- Ähnlich wie das Sortieren von Adressbüchern
- Handle jede Zahl als Sortierschlüssel
- Starte vom Least-significant-Bit

Most significant



Least significant



198099109123518183599
340199540380128115295
384700101594539614696
382408360201039258538
614386507628681328936

Jede Zeile entspricht
einer “langen” Zahl

Radix-Sort wurde schon **1887** in Arbeiten von Herman
Hollerith zu Volkszählungsmaschinen verwendet

Radix-Sort: Illustration

- Hier ein vereinfachtes Beispiel:

Jede Zeile entspricht einer
langen Zahl

7 4 2
7 4 8
0 5 4
6 8 8
4 1 2
2 3 0
9 3 5
1 1 6
1 6 1
4 3 4
3 8 5
6 6 6
0 3 1
0 1 3
3 6 5
1 7 3
0 1 6

Radix-Sort: Illustration

- Sortiere nach letzter Zahl:



<u>2</u>	<u>3</u>	<u>0</u>
<u>1</u>	<u>6</u>	<u>1</u>
<u>0</u>	<u>3</u>	<u>1</u>
<u>7</u>	<u>4</u>	<u>2</u>
<u>4</u>	<u>1</u>	<u>2</u>
<u>0</u>	<u>1</u>	<u>3</u>
<u>1</u>	<u>7</u>	<u>3</u>
<u>0</u>	<u>5</u>	<u>4</u>
<u>4</u>	<u>3</u>	<u>4</u>
<u>9</u>	<u>3</u>	<u>5</u>
<u>3</u>	<u>8</u>	<u>5</u>
<u>3</u>	<u>6</u>	<u>5</u>
<u>1</u>	<u>1</u>	<u>6</u>
<u>6</u>	<u>6</u>	<u>6</u>
<u>0</u>	<u>1</u>	<u>6</u>
<u>7</u>	<u>4</u>	<u>8</u>
<u>6</u>	<u>8</u>	<u>8</u>

Radix-Sort: Illustration

- Sortiere nach zweitletzter Zahl:

↓

<u>4</u>	<u>1</u>	<u>2</u>
<u>0</u>	<u>1</u>	<u>3</u>
<u>1</u>	<u>1</u>	<u>6</u>
<u>0</u>	<u>1</u>	<u>6</u>
<u>2</u>	<u>3</u>	<u>0</u>
<u>0</u>	<u>3</u>	<u>1</u>
<u>4</u>	<u>3</u>	<u>4</u>
<u>9</u>	<u>3</u>	<u>5</u>
<u>7</u>	<u>4</u>	<u>2</u>
<u>7</u>	<u>4</u>	<u>8</u>
<u>0</u>	<u>5</u>	<u>4</u>
<u>1</u>	<u>6</u>	<u>1</u>
<u>3</u>	<u>6</u>	<u>5</u>
<u>6</u>	<u>6</u>	<u>6</u>
<u>1</u>	<u>7</u>	<u>3</u>
<u>3</u>	<u>8</u>	<u>5</u>
<u>6</u>	<u>8</u>	<u>8</u>

Radix-Sort: Illustration

- Sortiere nach erster Zahl:



<u>0</u>	<u>1</u>	<u>3</u>
<u>0</u>	<u>1</u>	<u>6</u>
<u>0</u>	<u>3</u>	<u>1</u>
<u>0</u>	<u>5</u>	<u>4</u>
<u>1</u>	<u>1</u>	<u>6</u>
<u>1</u>	<u>6</u>	<u>1</u>
<u>1</u>	<u>7</u>	<u>3</u>
<u>2</u>	<u>3</u>	<u>0</u>
<u>3</u>	<u>6</u>	<u>5</u>
<u>3</u>	<u>8</u>	<u>5</u>
<u>4</u>	<u>1</u>	<u>2</u>
<u>4</u>	<u>3</u>	<u>4</u>
<u>6</u>	<u>6</u>	<u>6</u>
<u>6</u>	<u>8</u>	<u>8</u>
<u>7</u>	<u>4</u>	<u>2</u>
<u>7</u>	<u>4</u>	<u>8</u>
<u>9</u>	<u>3</u>	<u>5</u>

Zeitkomplexität

- Sortierung jeder der d Spalten mit Counting-Sort
- Gesamtkosten: $d(n + k)$
 - Wähle $k = 10$
 - Gesamtkosten: $\Theta(dn)$
- Partitionierung der d Zahlen in z.B. in Dreiergruppen
 - Gesamtkosten: $(n + 10^3)d/3$
- Wir arbeiten mit Binärzahlen anstellen von Dezimalen
 - Partitionierung der d Bits in Gruppen von r Bits
 - Gesamtkosten: $(n + 2^r)d/r$
 - Wähle $r = \log n$
 - Gesamtkosten: $O(dn / \log n)$
 - Vergleiche mit $O(dn \log n)$ für das einfache Verfahren
- Aber: Radix-Sort hat hohen konstanten Faktor

Platzkomplexität

- Verwendung von Counting-Sort
- Daher zusätzlicher Speicher nötig: $\theta(n)$

Listen als abstrakte Datentypen (ADTs)

Denotation: [4, 2, 9]

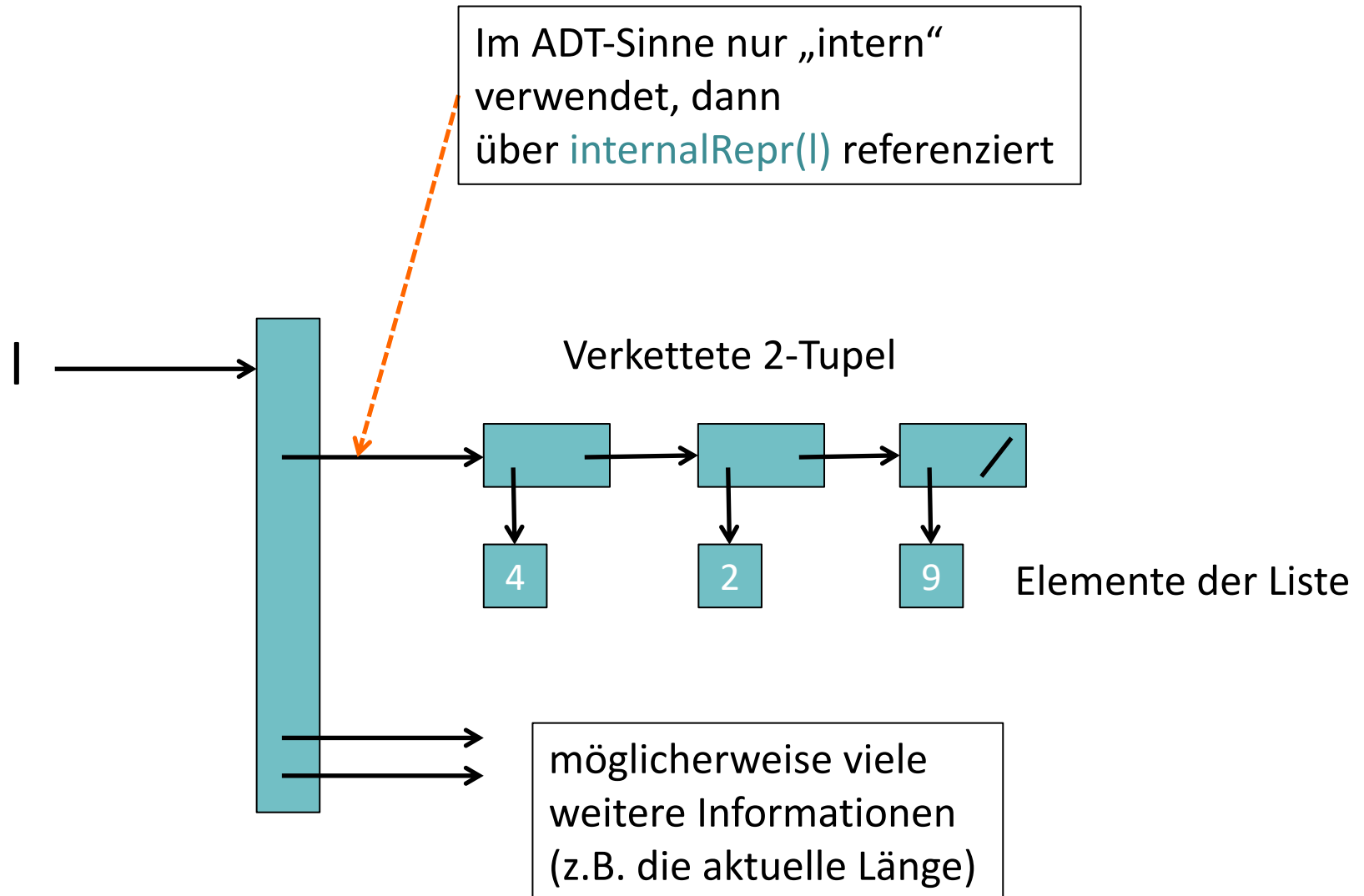
Operationen:

- function **makeList**() liefert neue Liste (am Anfang leer)
- procedure **insert**(*e*, *l*) fügt Element *e* am Anfang in Liste *l* ein, verändert *l*
- procedure **delete**(*e*, *l*) löscht Element *e* sofern enthalten, verändert *l*, wenn ein Element gelöscht wird
- function **first**(*l*) gibt Last-in-Element zurück (Fehler, wenn *l* leer)
- procedure **deleteFirst**(*l*) löscht Last-in-Element in *l* (Fehler, wenn *l* leer)
- function **length**(*l*) gibt Anzahl der Elemente in *l* zurück
- function **mtList?**(*l*) gibt **true** zurück, wenn *l* leer ist, sonst **false**

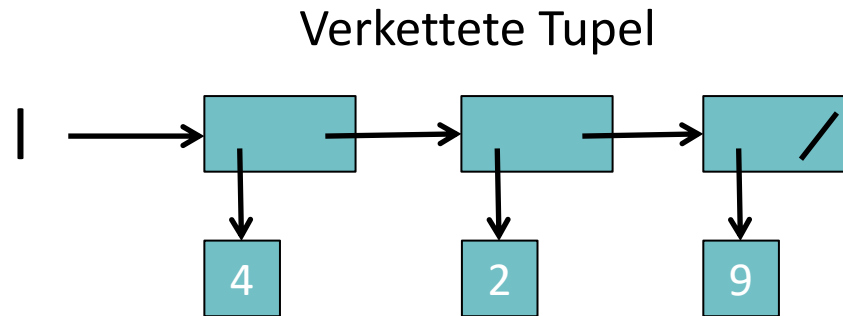
Iteration (last-in first-out):

- for *e* in *l* do ... *oder auch* for *e* ∈ *l* do ...

Listen intern (Beispiel)



Listen als Glaskästen



- (e, l) liefert Tupel mit Element e und Liste l
- $()$ oder auch $[]$ steht für ein besonderes Tupel, das die leere Liste symbolisiert (auch nil genannt)
- **Beispiele:**
 - $[4, 2, 9] = (4, (2, (9, ())))$ $[4] = (4, ())$ $[] = ()$
- Sei $l_1 = [4, 2, 9]$, dann **Zugriff** mit $(e, l_2) := l_1$
dann gilt: $e = 4$ und $l_2 = [2, 9] = (2, (9, ()))$

Listen als Glaskästen

Denotation: [4, 2, 9]

Operationen:

- **function** **cons**(*e*, *l*) fügt Element *e* am Anfang in Liste *l* ein, verändert *l* nicht
- **function** **first**(*l*) gibt die erste Komponente des Tupels zurück (Fehler, wenn *l* leer)
- **function** **rest**(*l*) gibt die zweite Komponente des Tupels zurück (Fehler, wenn *l* leer)
- **function** **length**(*l*) gibt Anzahl der Elemente in *l* zurück
- **function** **mt?**(*l*) gibt **true** zurück, wenn *l* = () ist, sonst **false**

Iteration (last-in first-out):

- for *e* in *l* do ... *oder auch* for *e* ∈ *l* do ...

Kellerspeicher / Stapelspeicher / Stack

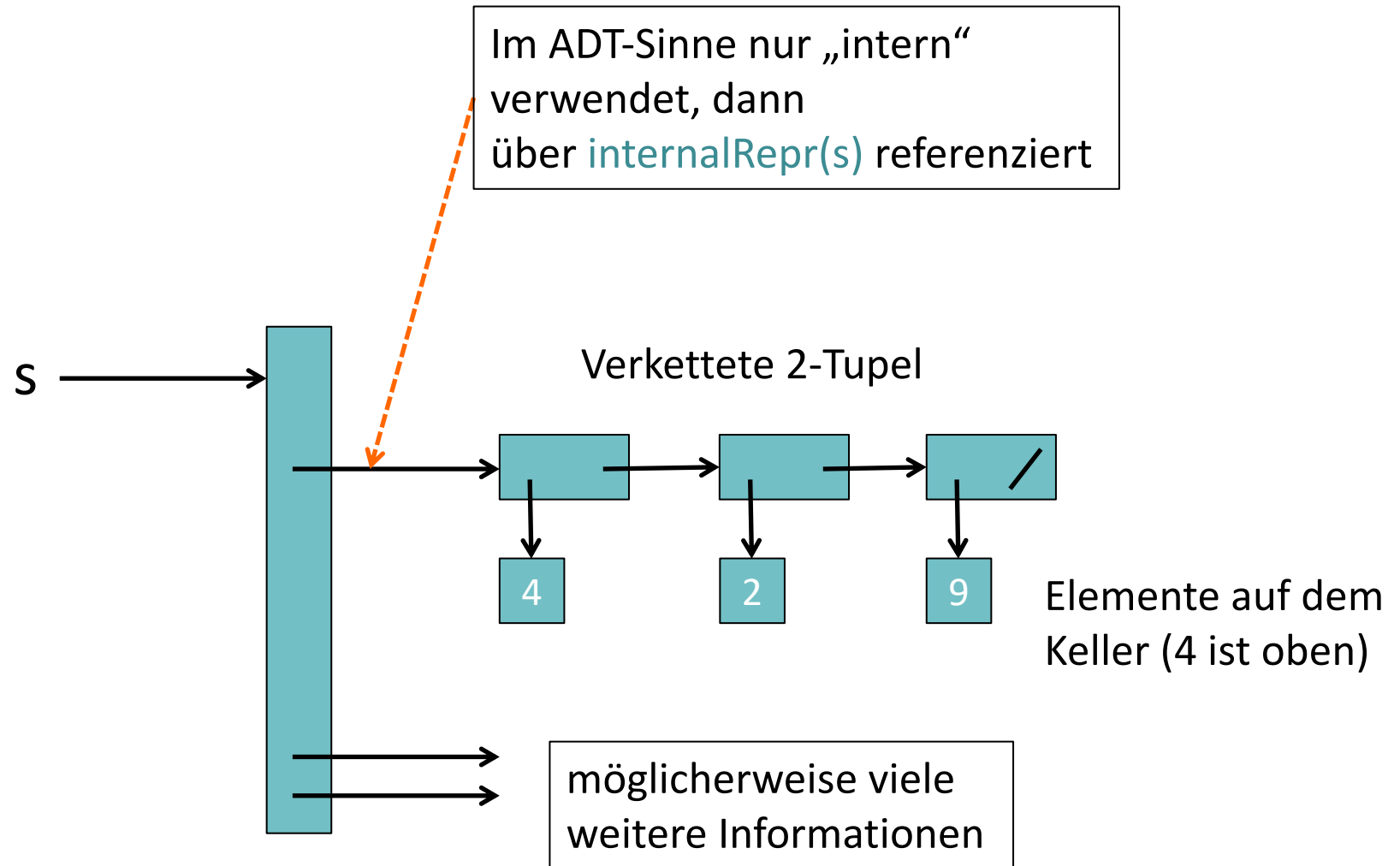
Denotation: [4, 2, 9] (4 ist "oben")

Operationen:

- function **makeStack()** liefert leeren Keller
- procedure **push**(e, s) fügt Element e oben in den Keller s ein, verändert s
- function **top**(s) gibt oberes Element zurück (Fehler, wenn s leer)
- procedure **pop**(s) löscht Top-Element in s (Fehler, wenn s leer)
- function **mtStack?**(s) gibt true zurück, wenn s leer ist, sonst false

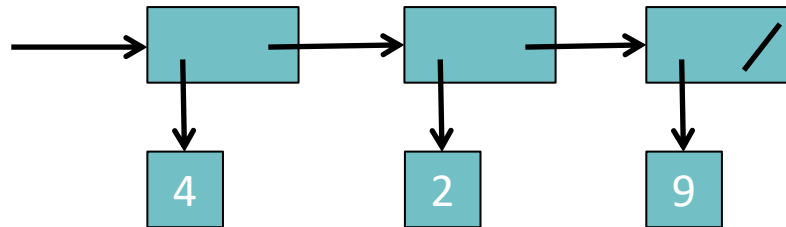
Iteration: nicht vorgesehen (evtl. wie Liste)

Keller intern (Beispiel)



Kellerspeicher als Glaskästen?

Verkettete Tupel



Schlange / Queue (First-in-First-out-Speicher)

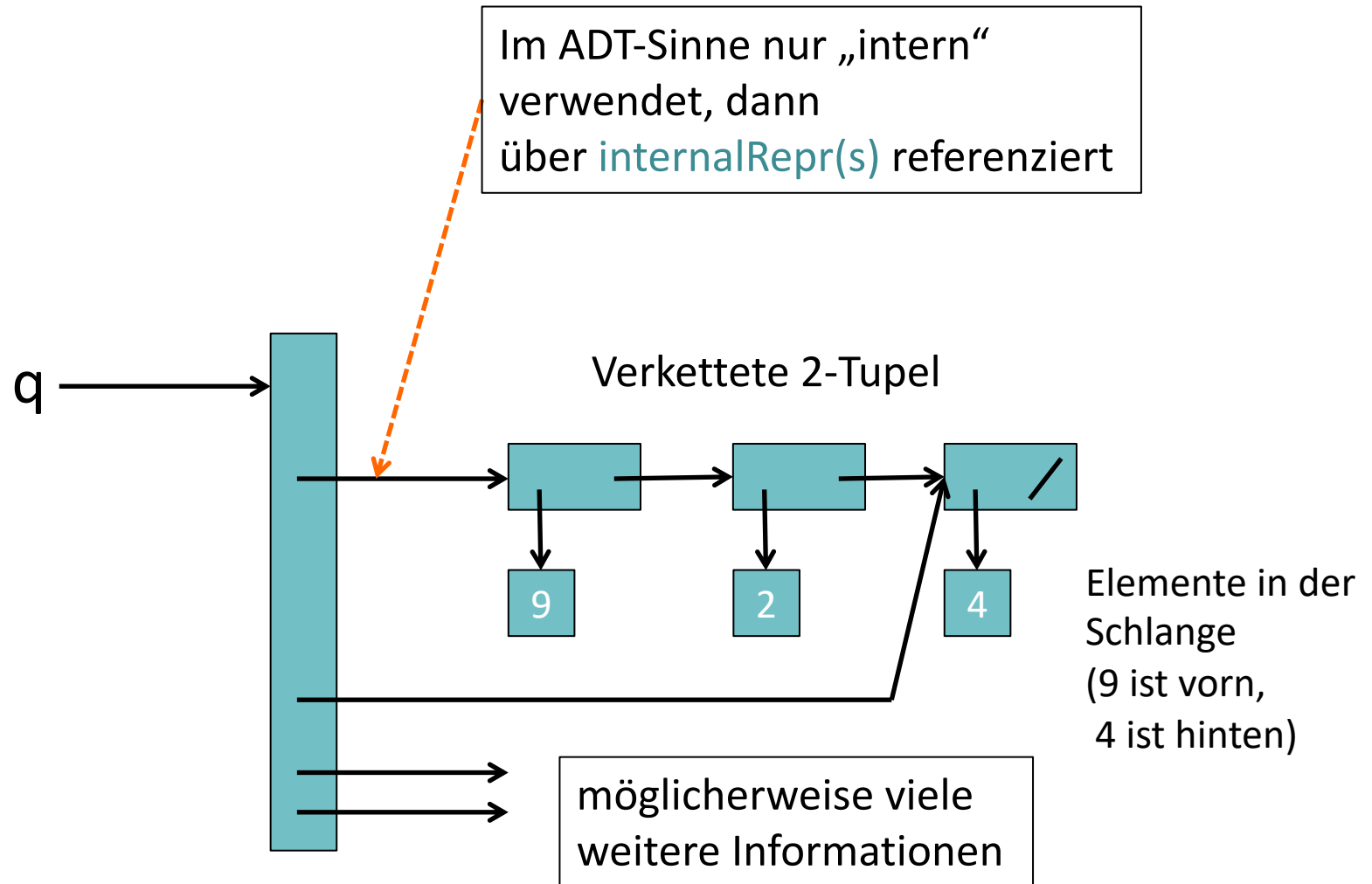
Denotation: [4, 2, 9] (4 ist "hinten", 9 ist "vorn", kommt zuerst dran)

Operationen:

- **function** **makeQueue**() liefert leere Warteschlange
- **procedure** **enqueue**(e, q) fügt Element e hinten in die Schlange q ein, verändert q
- **function** **next**(q) gibt vorderes Element zurück, verändert q nicht
- **function** **dequeue** (q) gibt vorderes Element zurück, verändert q
- **function** **mtQueue?**(q) gibt **true** zurück, wenn q leer ist, sonst **false**

Iteration: nicht vorgesehen (evtl. wie Liste)

Queue intern (Beispiel)



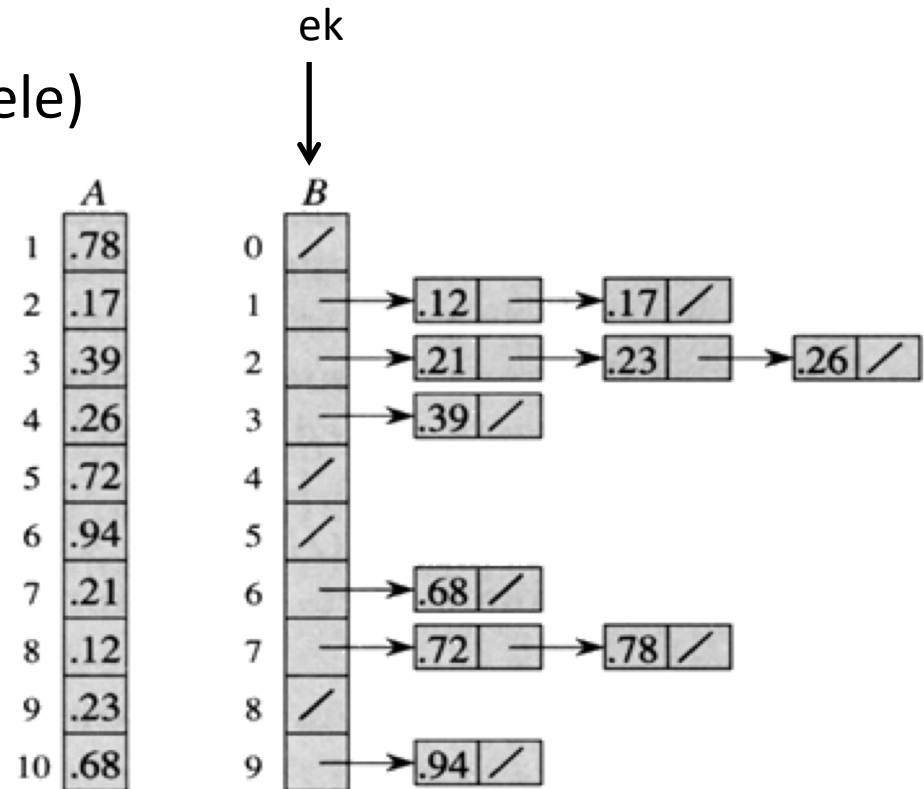
Bucket-Sort

1. procedure BUCKET-SORT (A)
2. $n \leftarrow \text{length}(A)$, $k \leftarrow$ Anzahl der Eimer
3. for $i = 1$ to n do
4. Füge $A[i]$ in den richtigen Eimer ein
5. for $i = 1$ to k do
6. Sortiere i -ten Eimer mit einer vergleichsbasierten Sortierfunktion
7. Hänge die Eimer in der richtigen Ordnung hintereinander



Wie wollen wir die Eimerkette implementieren?

- Verkettete Liste oder Feld für Eimer**kette**?
- Verkettete Listen oder Felder für **Einzeleimer**?
 - Verkettete Listen sparen Platz (einige Eimer haben kaum Einträge, andere haben viele)
 - Aber mit verketteten Listen können wir “schnelle” Sortierverfahren wie Heap-Sort oder Quicksort nicht verwenden
 - **Sortierte Listen!**



Analyse von Bucketsort



- Sei $S(m)$ die Anzahl der Vergleiche für einen Eimer mit m Schlüsseln
- Setze n_i auf die Anzahl der Schlüssel im i -ten Eimer
- Gesamtzahl der Vergleiche = $\sum_{i=1}^k S(n_i)$ bei k Eimern

Analyse (2)

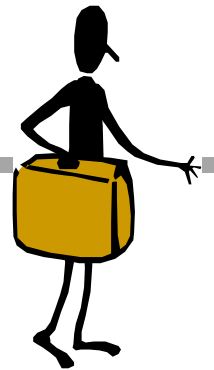
- Sei $S(m) \in \Theta(m \log m)$
- Falls die Schlüssel gleichmäßig verteilt sind, beträgt die Eimergröße n/k
- Gesamtzahl der Vergleiche
 - $= k(n/k) \log(n/k)$
 - $= n \log(n/k)$
- Falls $k=n/10$, dann reichen $n \log(10)$ Vergleiche (Laufzeit ist linear in n)

Lineare Sortierung: Einsicht

Je mehr man über das Problem weiß, desto eher kann man einen optimalen Algorithmus entwerfen

- Gesucht ist ein Verfahren **S**, so dass $\{ \mathbf{P} \} \mathbf{S} \{ \mathbf{Q} \}$ gilt (Notation nach **Hoare**)
 - Vorbedingung: **P** = ?
 - Invarianten („Axiome“): **I** = ?
 - Nachbedingung: **Q** = $\forall 1 \leq i < j \leq n: A[i] \leq A[j]$
 - Nebenbedingungen: ?

Zusammenfassung



- Bisher behandelt:
 - Sortieren durch Vergleichen (vorige Sitzungen)
 - Sortieren durch Verteilen (lineares Sortieren)
- Es kommt:
 - Prioritätswarteschlangen
 - MinHeaps (zum Vergleich mal anders herum)
 - Binomiale Heaps (effiziente Vereinigung von Heaps)
 - Fibonacci Heaps (Einführung der amortisierten Analyse)