Algorithmen und Datenstrukturen

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Stefan Werner (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

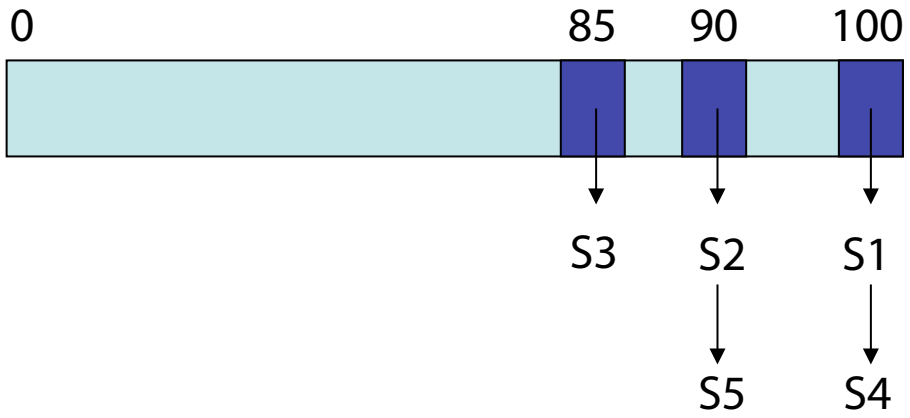
Counting-Sort

- ***Eingabe***: $A[1 \,..\, n]$, wobei $A[j] \in \{1, 2, \ldots, k\}$.
- ***Ausgabe***: $B[1 \,..\, n]$, sortiert.
- ***Hilfsspeicher***: $C[1 \,..\, k]$.

- Kein In-situ-Sortieralgorithmus
- Benötigt Θ (n+k) zusätzliche Speicherplätze

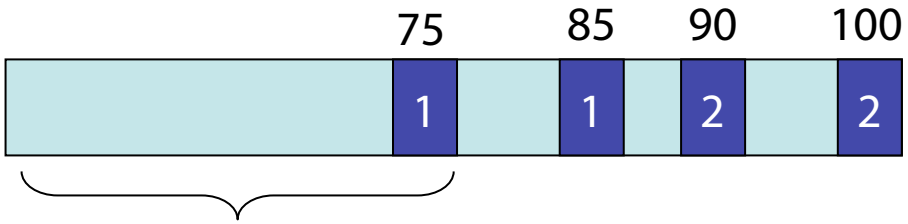
Intuition

- 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

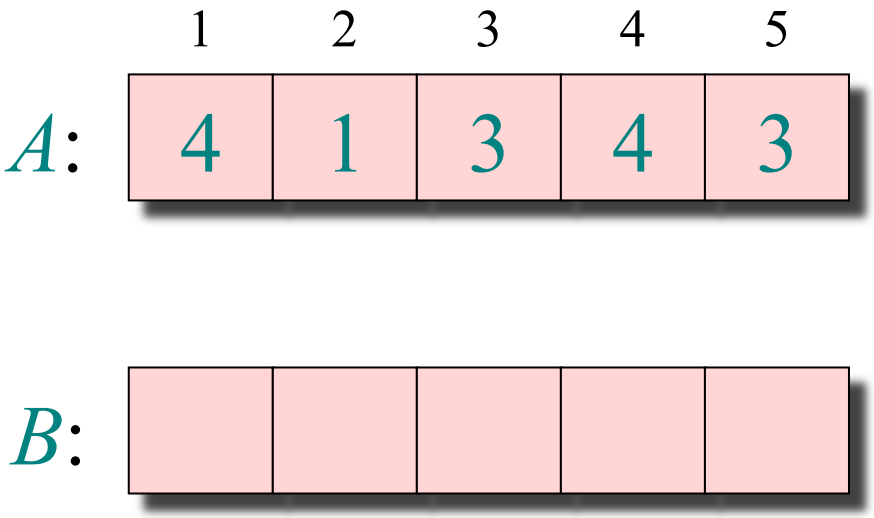
200 Studierende mit Score ≤ 90
Was ist der Rang für einen Studenten mit Score 90?

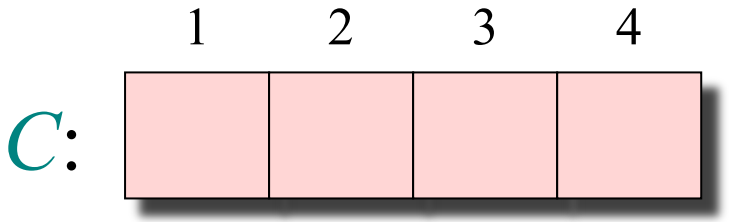
200 or 199

Counting-Sort

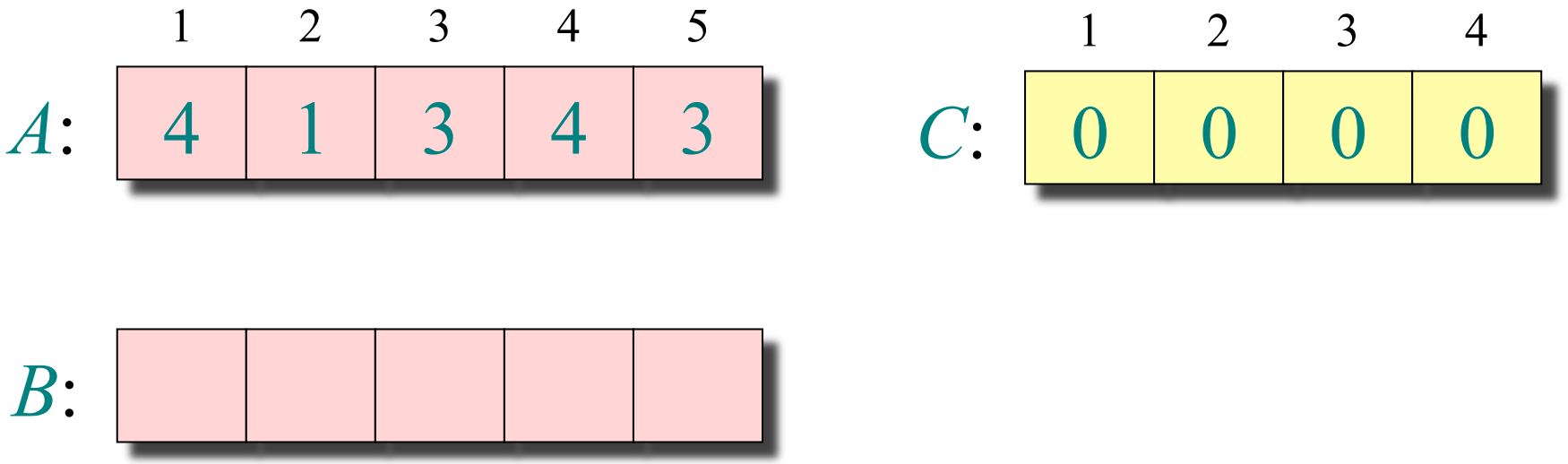
```
1. for i ← 1 to k                                          ▷ Initialisiere
       do C[i] ← 0
2. for j ← 1 to n                                          ▷ Zähle
       do C[A[j]] ← C[A[j]] + 1          ▷ C[i] = |{key = i}|
3. for i ← 2 to k                                          ▷ Bestimme Summe
       do C[i] ← C[i] + C[i–1]           ▷ C[i] = |{key ≤ i}|
4. for j ← n downto 1                                      ▷ Ordne neu
       do B[C[A[j]]] ← A[j]
          C[A[j]] ← C[A[j]] – 1
```

Counting-Sort Beispiel





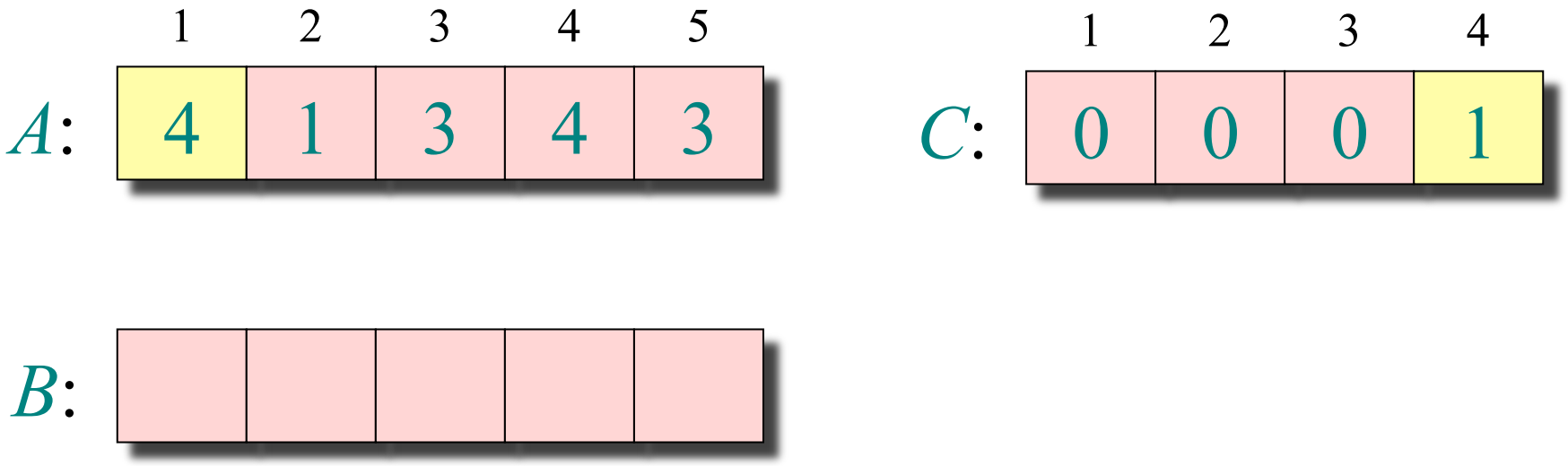
Schleife 1: Initialisierung



1. for $i \leftarrow 1$ **to** k

do $C[i] \leftarrow 0$

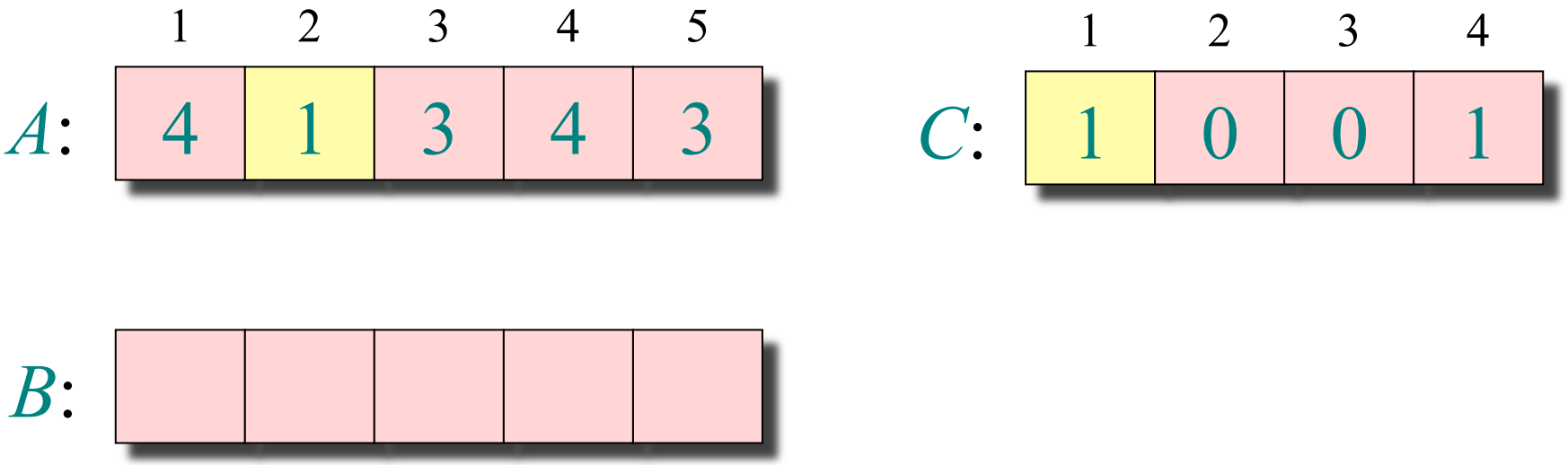
Schleife 2: Zähle



2. **for** $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

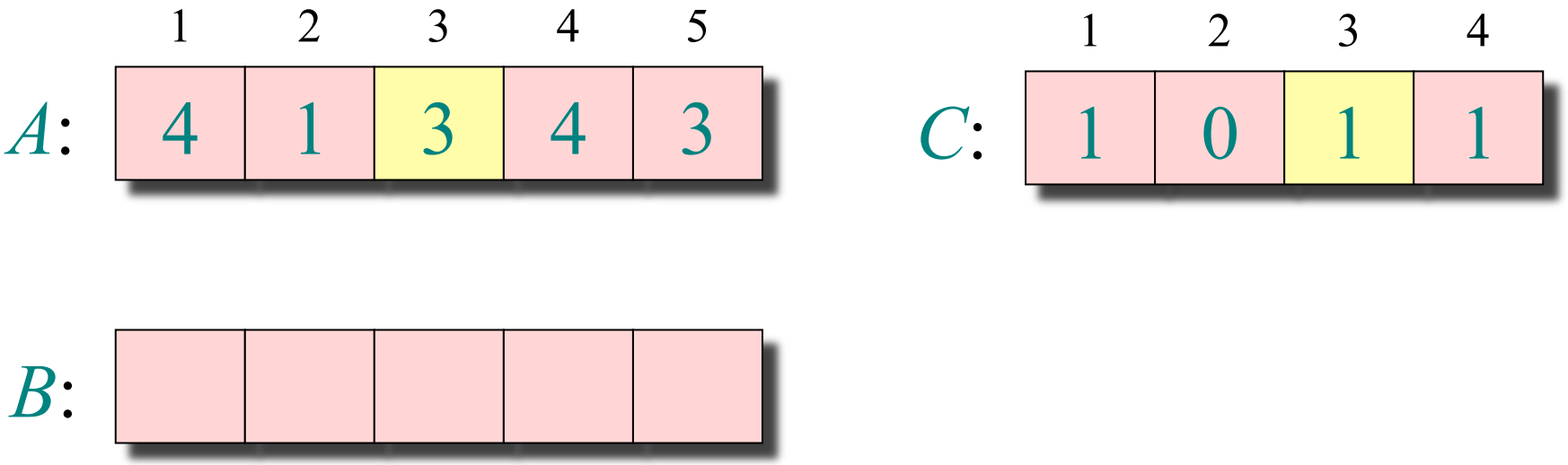
Schleife 2: Zähle



2. **for** $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

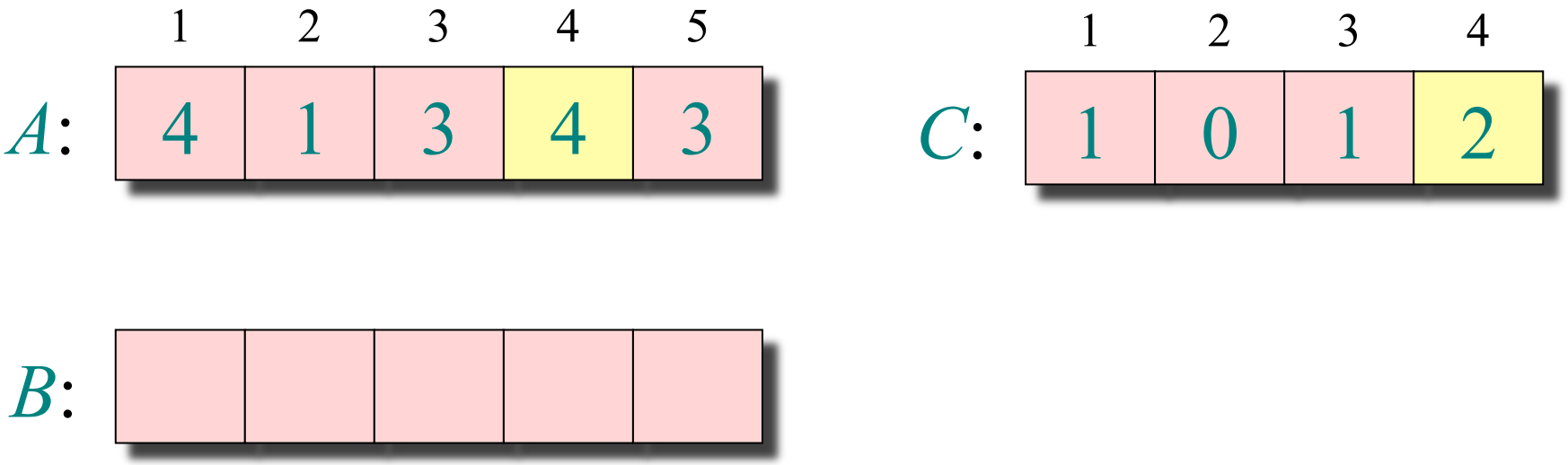
Schleife 2: Zähle



2. **for** $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

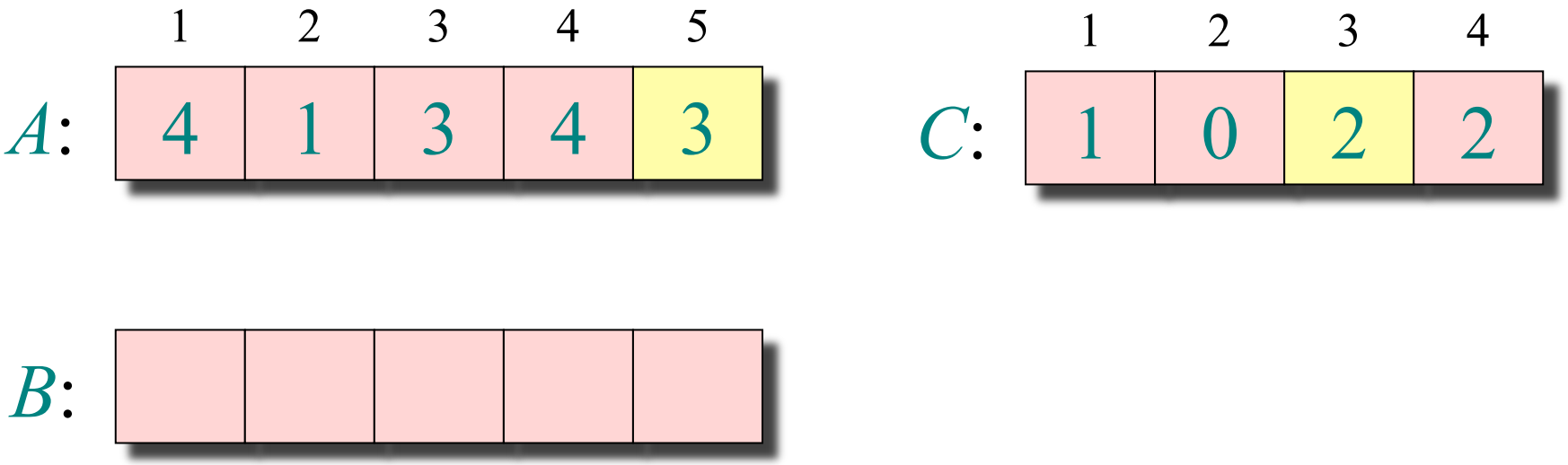
Schleife 2: Zähle



2. **for** $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

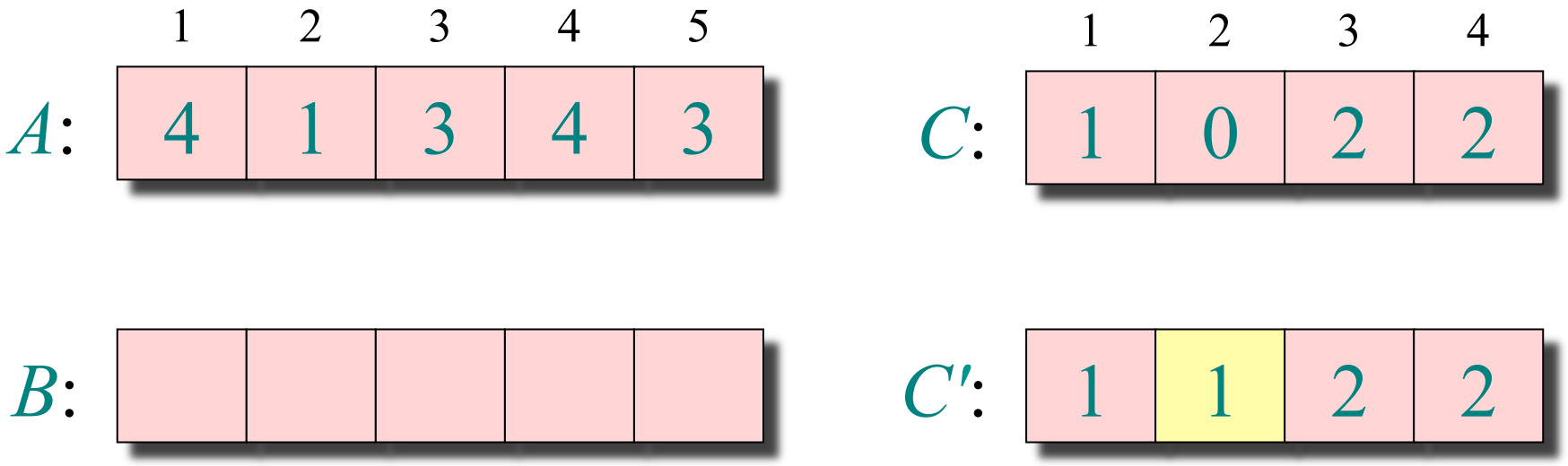
Schleife 2: Zähle



2. **for** $j \leftarrow 1$ **to** n

do $C[A[j]] \leftarrow C[A[j]] + 1$ ▷ $C[i] = |\{\text{key} = i\}|$

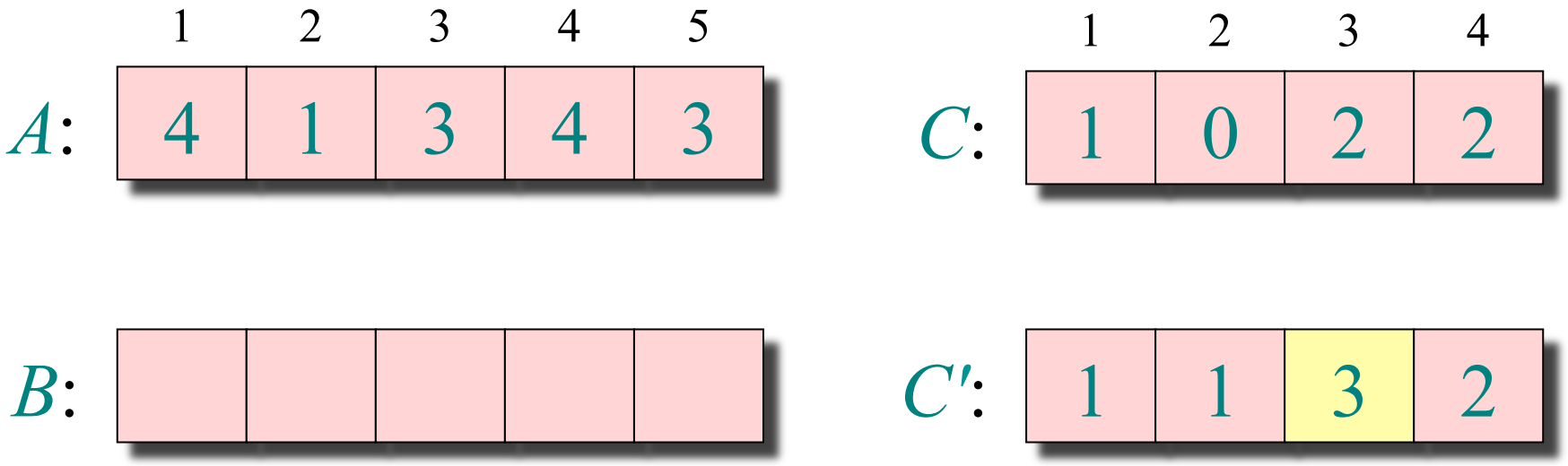
Schleife 3: Berechne Summe



3. **for** $i \leftarrow 2$ **to** k

do $C[i] \leftarrow C[i] + C[i-1]$ ▷ $C[i] = |\{\text{key} \leq i\}|$

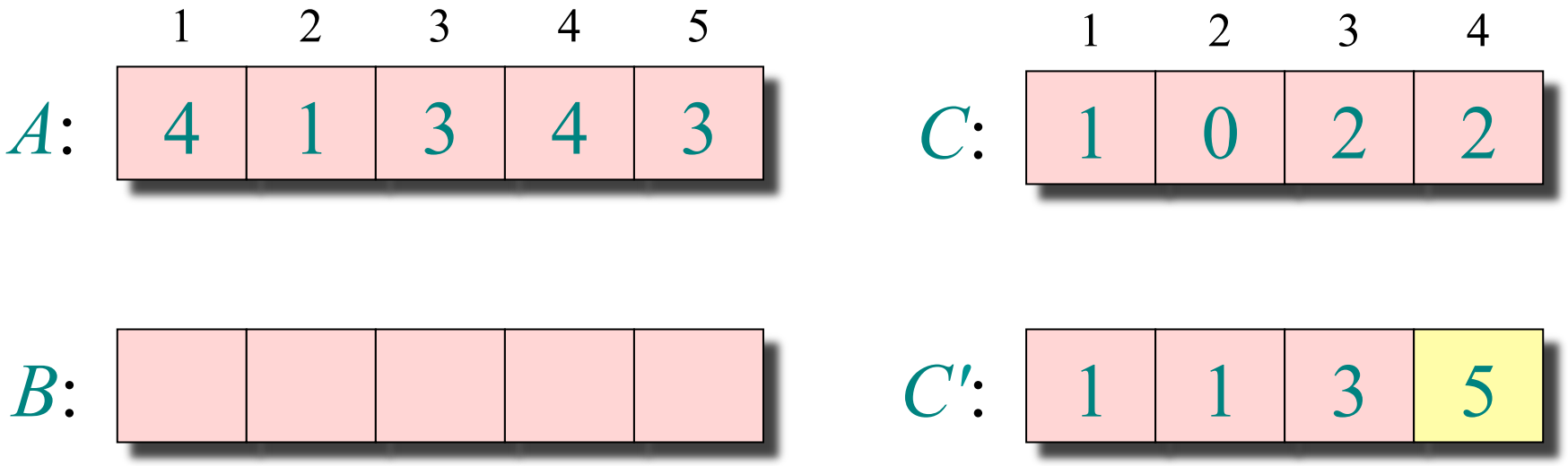
Schleife 3: Berechne Summe



3. **for** $i \leftarrow 2$ **to** k

do $C[i] \leftarrow C[i] + C[i-1]$ ▷ $C[i] = |\{\text{key} \leq i\}|$

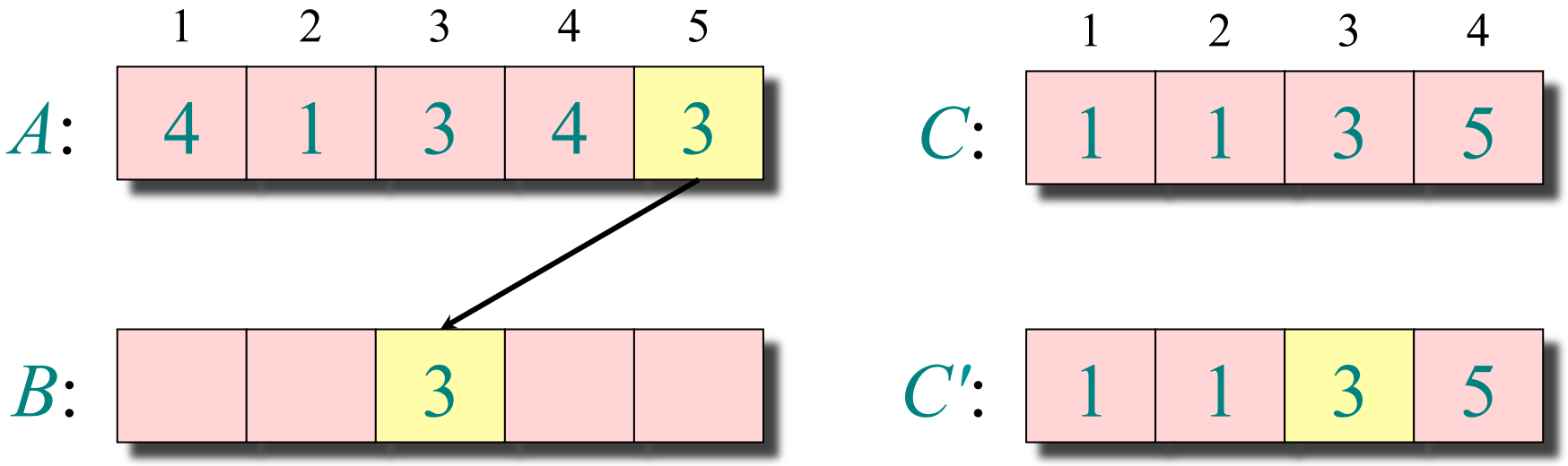
Schleife 3: Berechne Summe



3. **for** $i \leftarrow 2$ **to** k

do $C[i] \leftarrow C[i] + C[i-1]$ ▷ $C[i] = |\{\text{key} \leq i\}|$

Schleife 4: Ordne neu

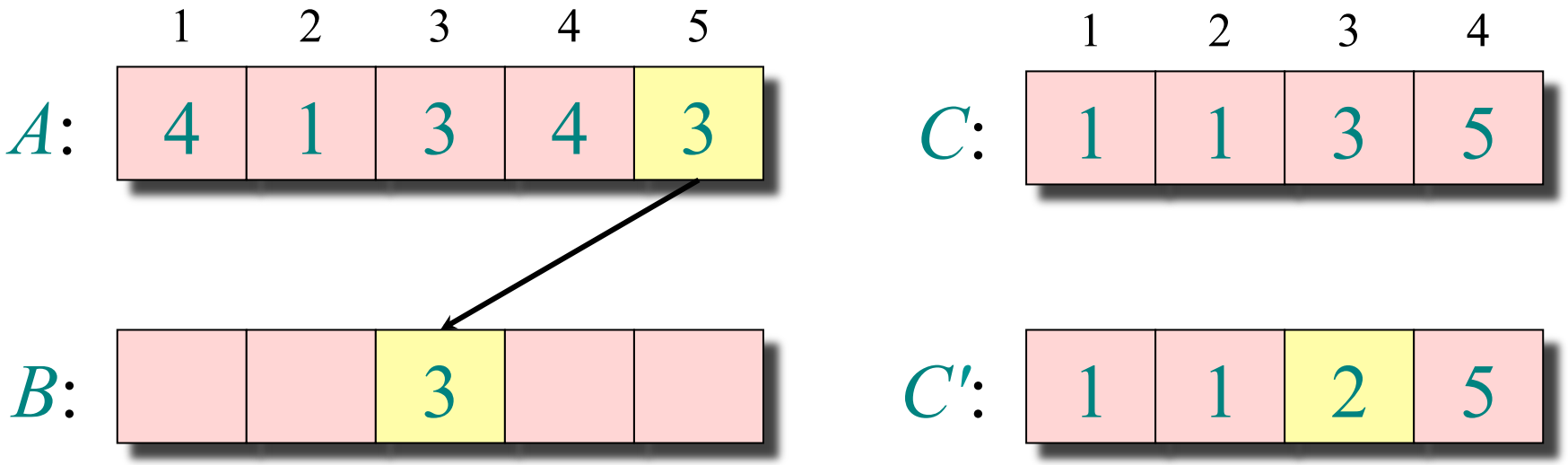


4. for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu

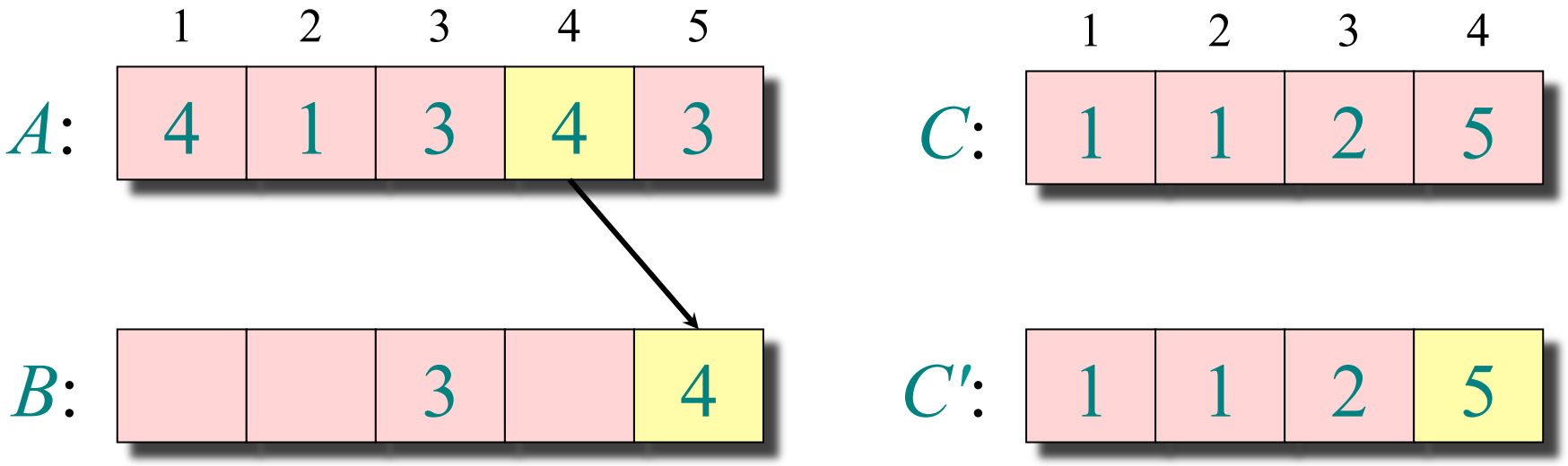


4. for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu

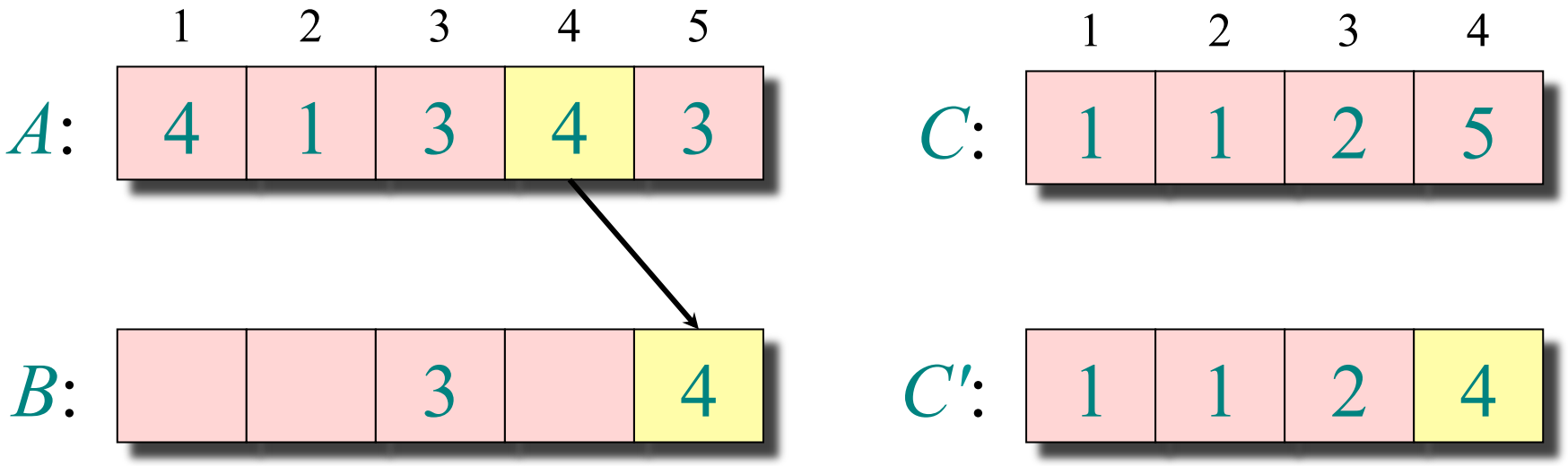


4. for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu

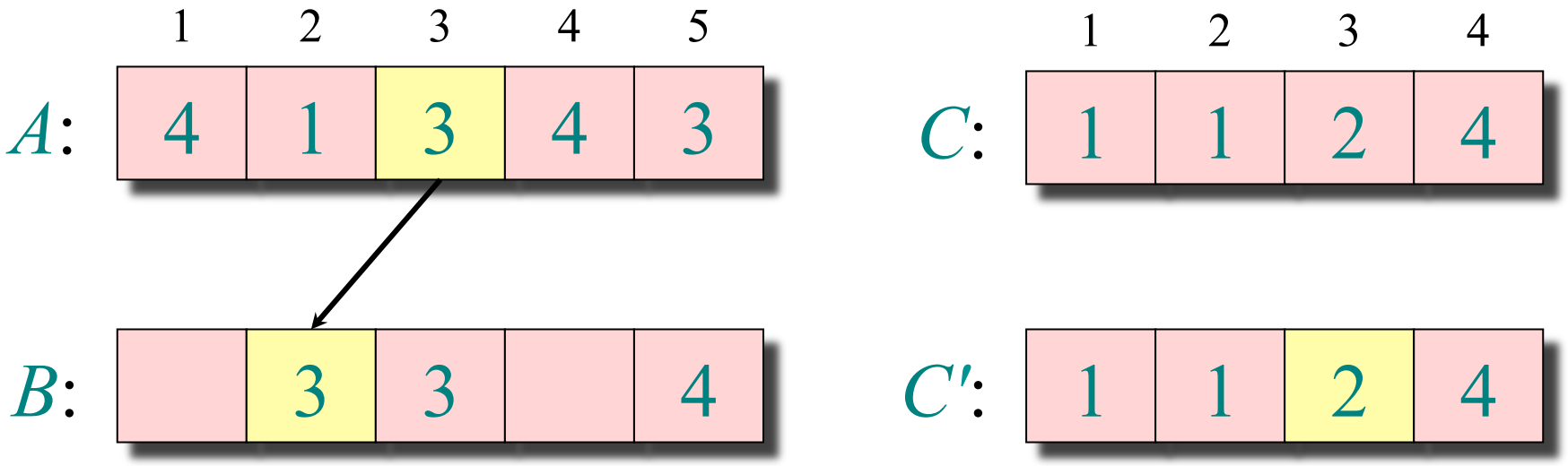


4. for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu

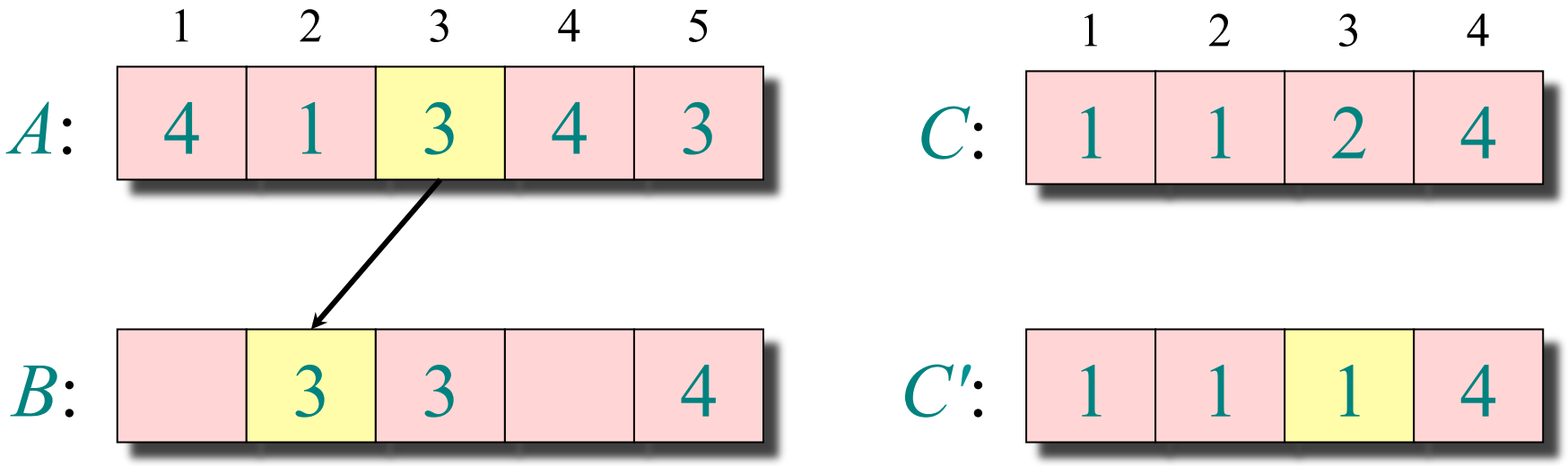


4. for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu

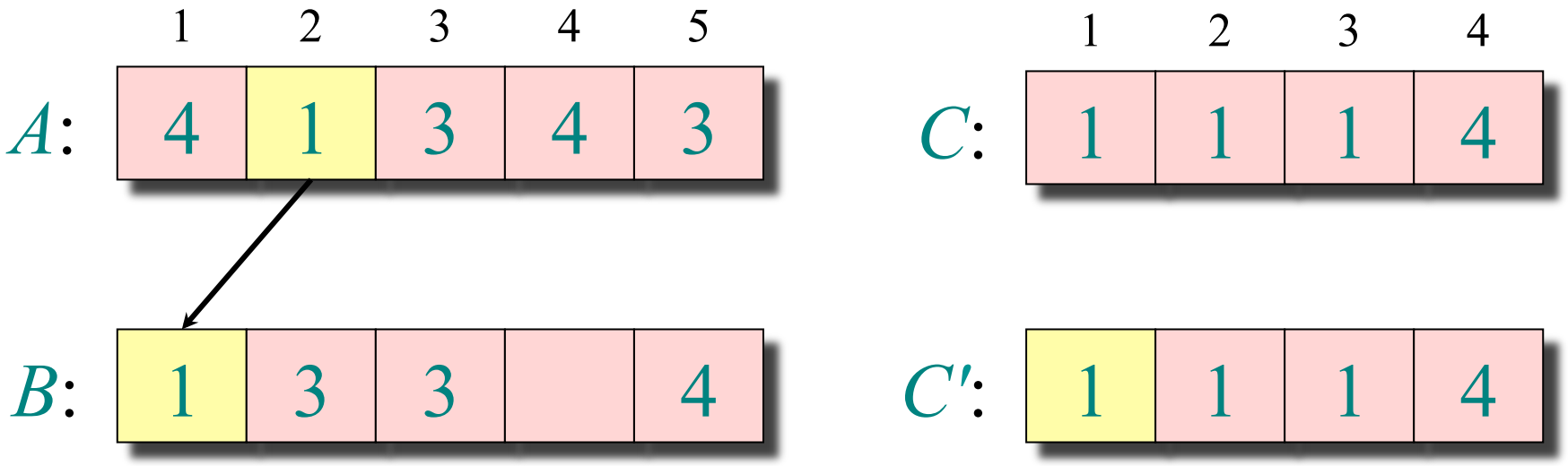


4. for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu

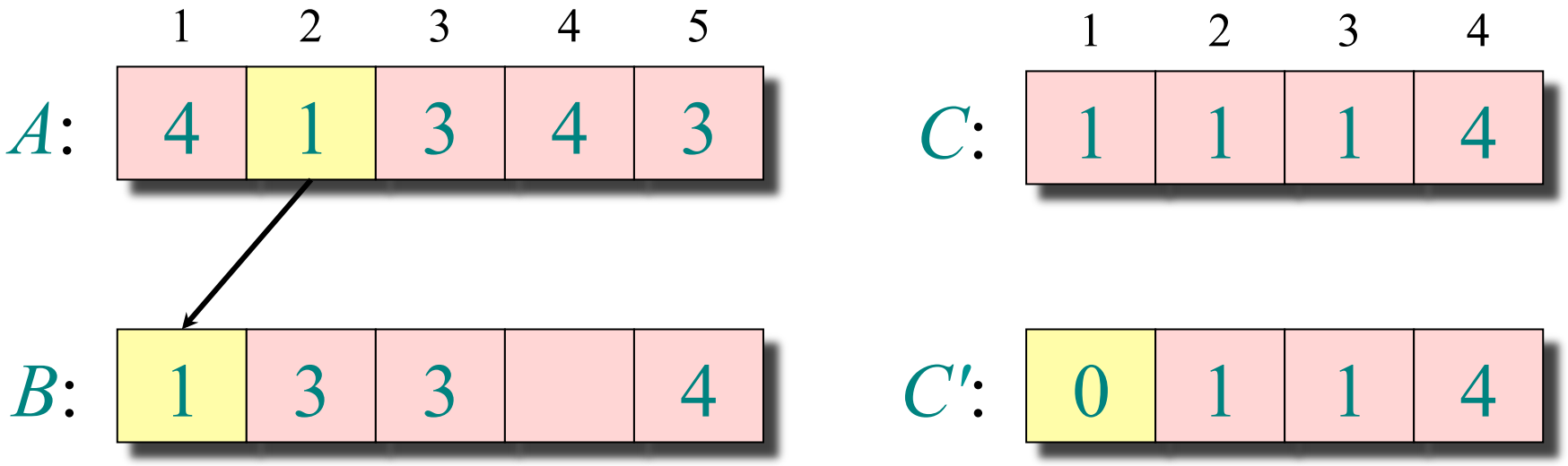


4. for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu

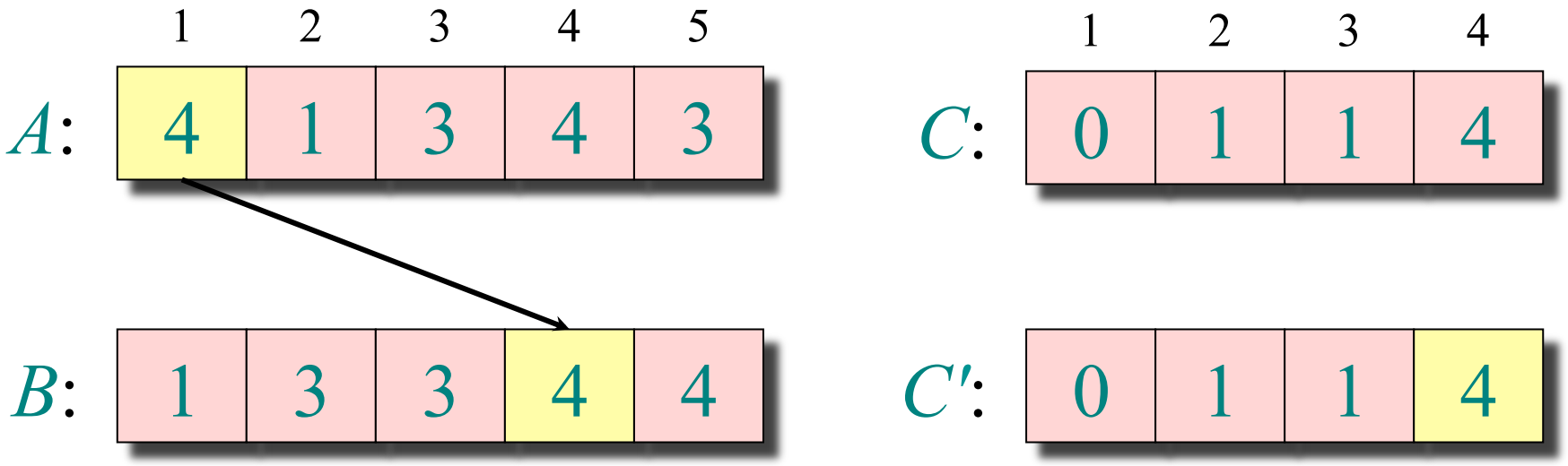


4. for $j \leftarrow n$ **downto** 1

do $B[C[A[j]]] \leftarrow A[j]$

$C[A[j]] \leftarrow C[A[j]] - 1$

Schleife 4: Ordne neu

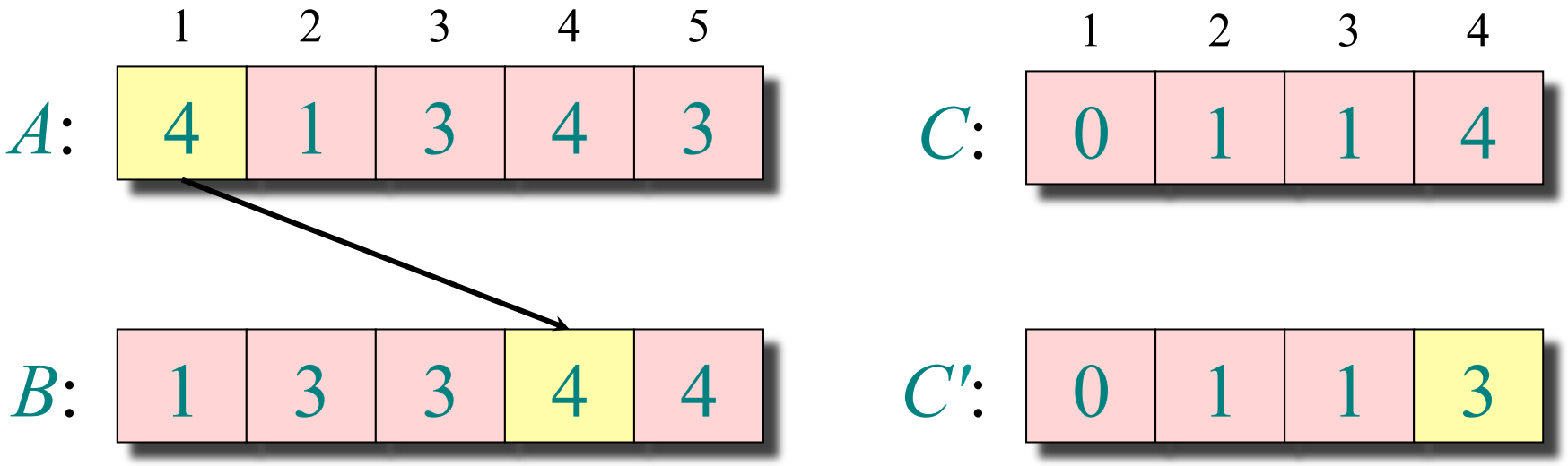


4. for $j \leftarrow n$ **downto** 1

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 ← 1 to k do
3:          C[i] ← 0
4:      for j ← 1 to length(A) do
5:          C[A[j]] ← C[A[j]] + 1
6:      ▷ C[i] enthält nun die Anzahl der Elemente, die gleich i sind.
7:      for i ← 2 to k do
8:          C[i] ← C[i] + C[i − 1]
9:      ▷ C[i] enthält nun die Anzahl der Elemente, die keiner oder gleich i sind.
10:     for j ← length(A) downto 1 do
11:         B[C[A[j]]] ← A[j[
12:         C[A[j]] ← 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 $B[C[A[j]]] \leftarrow \mathrm{A}[j]$
$C[A[j]] \leftarrow C[A[j]] - 1$

$\Theta(n + k)$

Laufzeit: Wodurch wird sie reduziert?

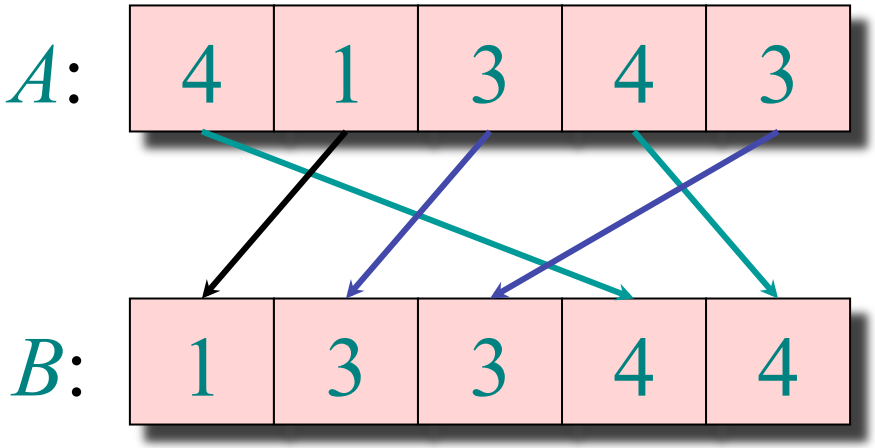
- Falls $k = O(n)$, dann braucht Counting-Sort $\Theta(n)$ viele Schritte.
- Aber theoretisch braucht doch Sortierung $\Omega(n \log n)$ viele Schritte!
- Gibt es ein Problem mit der Theorie?

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 dieses 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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

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

Aufgabe: Sortiere Studierende nach (state, city, street).

Stabiles Sortieren

- Die meisten $\Theta(n^2)$-Sortieralgorithmen sind stabil
  - oder können "einfach" stabil gemacht werden
- Die meisten $\Theta(n \log n)$-Sortieralgorithmen 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)]

(5, 1) < (5, 3)

(2, 5) < (2, 7)

Wie kann man sehr große Zahlen sortieren?

19809910912351818359934019954038012811529538470010159453961469638240836020103925853861438650762868132893673814865209099036919798708408709665302029918566412442123451645478539207557478591318855309952235931373973542670574904436181117677952935819148373775278155017642212211106741425222044033129376077180987973383291808365650470232665468405698211977095942752524552807615323904705082030544563984720161116847833424065119923801959

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 vergleiche(A[i], A[j]) < 0 mit

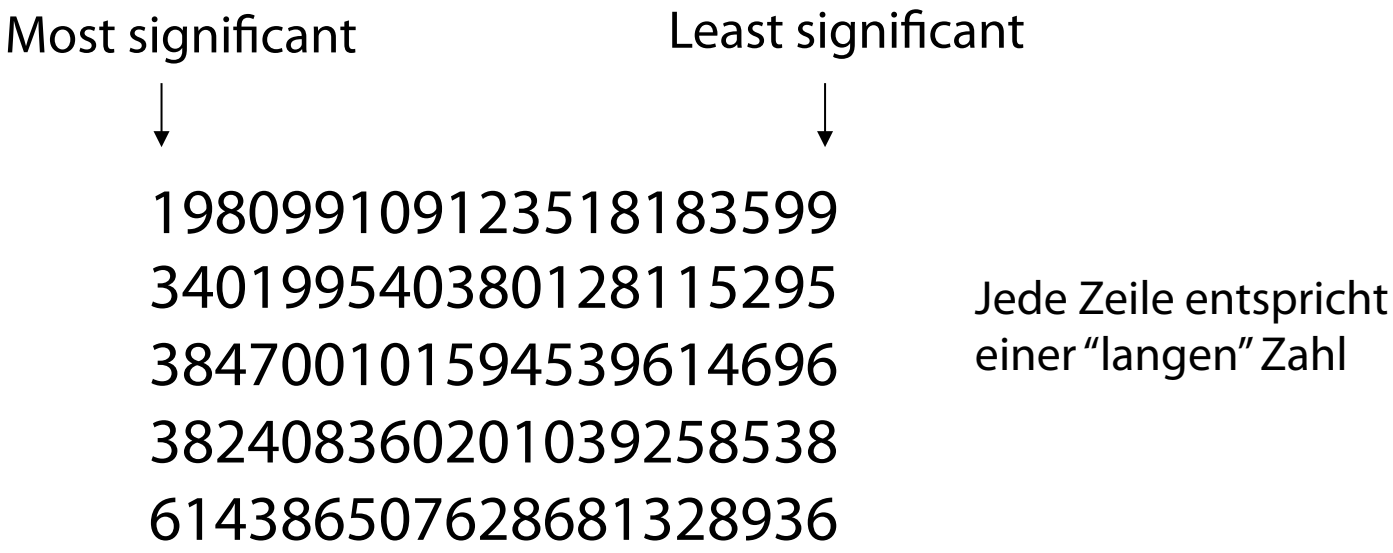
```
function vergleiche(s, t)
  for i = 1 to max( { length(s), length(t) } ) do
    if s[i] < t[i] then
      return -1
      else if s[i] > t[i] then
              return 1
  return 0
```

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

Gesamtkosten: $\Theta(d\ n \log n)$

Radix-Sort

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



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 letzer Zahl:

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

Radix-Sort: Illustration

- Sortiere nach zweitletzer Zahl:

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

Radix-Sort: Illustration

- Sortiere nach erster Zahl:

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

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: $dn\ /\ log\ n$
  - Vergleiche mit $dn\ log\ n$
- Aber: Radix-Sort hat hohen konstanten Faktor

Platzkomplexität

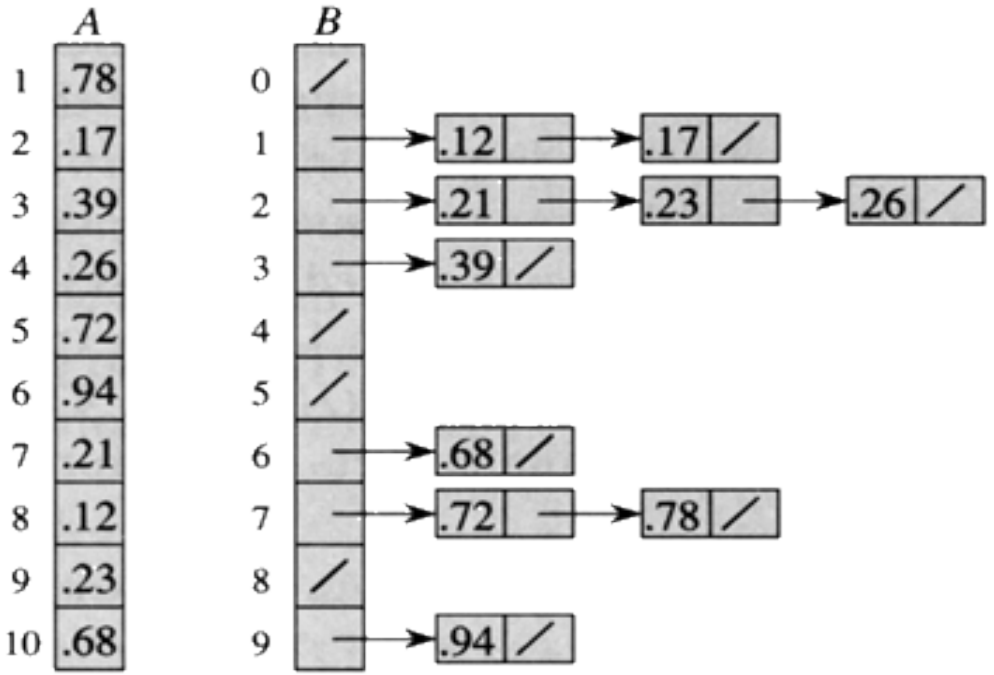
- Verwendung von Counting-Sort
- Daher zusätzlicher Speicher nötig: $\Theta(n)$

Bucket-Sort

1. procedure BUCKET-SORT (A)
2. $n \leftarrow$ 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 Eimer implementieren?

- Verkettete Listen oder Felder?
- 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



Danksagung

Nachfolgende „blaue“ Folien sind aus den Materialien der Vorlesung „Einführung in die Programmierung“ von Stefan Fischer (UzL) übernommen.

Listen (1/11)

type List(T)
import Nat
operators
[ ]: → List
_:_ : T×List → List
head : List → T
tail : List → List
length : List → Nat
axioms ∀ *l*: List, ∀ *x*: T
head (*x*:*l*) = *x*
tail (*x*:*l*) = *l*
length([])=0
length(*x*:*l*)
= succ(length(*l*))

Definition einer abstrakten Klasse Liste:

(Gibt die Schnittstelle vor und kann durch ggf. verschiedene Implementierungen realisiert werden)

Liste
head(): Object {abstract} tail(): Liste {abstract} length(): Integer {abstract} addFirst(Object) {abstract}

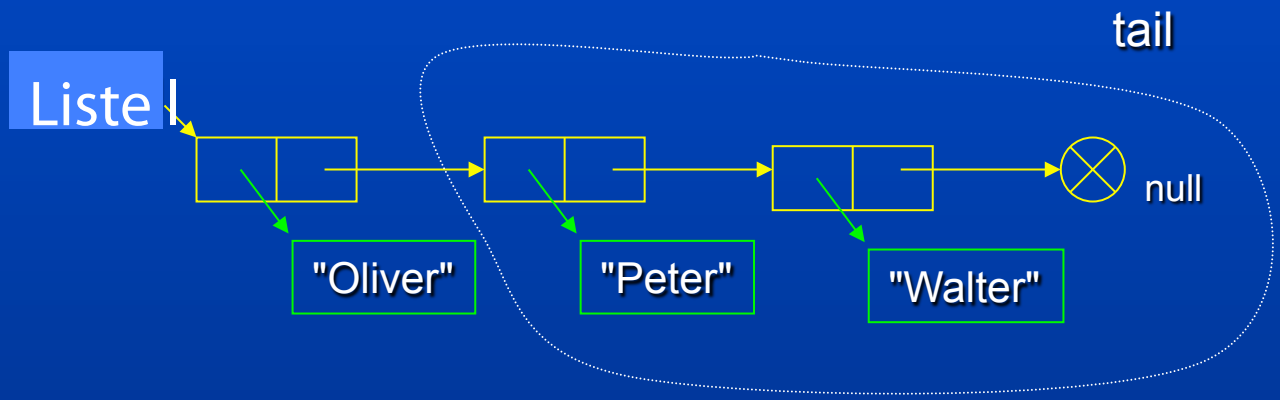
Listen (2/11)

Implementierung (1/8)

1. **Idee: als Verkettung einzelner Objekte (Einfach verkettete Liste)**

Beispiel: Liste von Namen

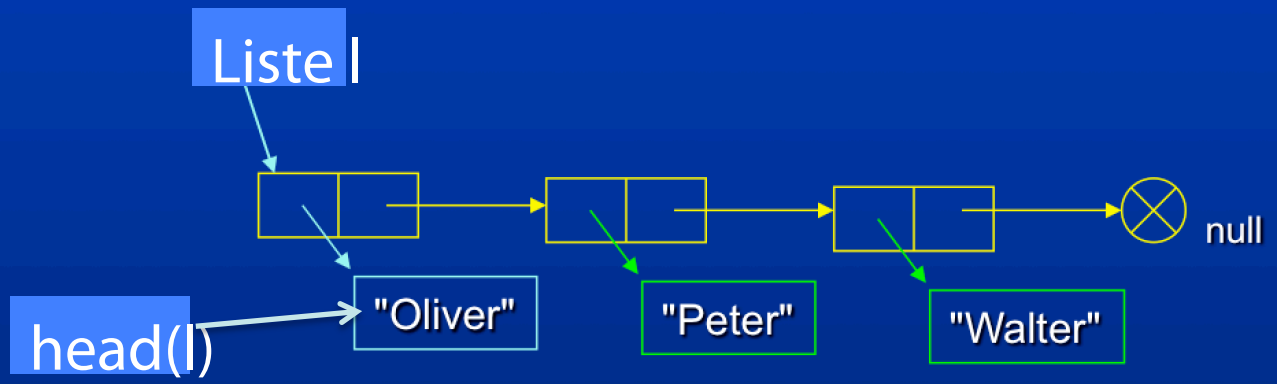
z.B. ["Oliver", "Peter", "Walter"]



Knoten (engl. Node; hier: Listenelement)

Analyse ausgewählter Listenoperationen

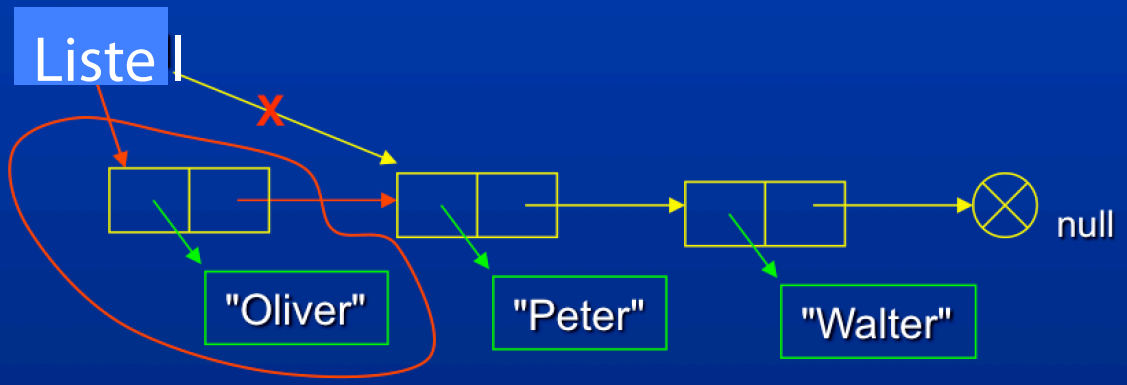
2. **head(): Object**
   //Rückgabe des Objekts am Listenkopf
   Ablauf:
   a. Fallunterscheidung: O(1)
   b. Rückgabe von null: O(1)
   c. Objekt des ersten Listenelements zurückgeben: O(1)
   Insgesamt: O(1)



Analyse ausgewählter Listenoperationen

1. **addFirst(Object)**
   //Anfügen eines Objekts am Listenkopf
   Ablauf:
   a. Erzeugen eines neuen Knoten: O(1)*
   b. Neuen Knoten auf ehemaligen head verweisen lassen: O(1)
   c. "Umlenken" von head: O(1)

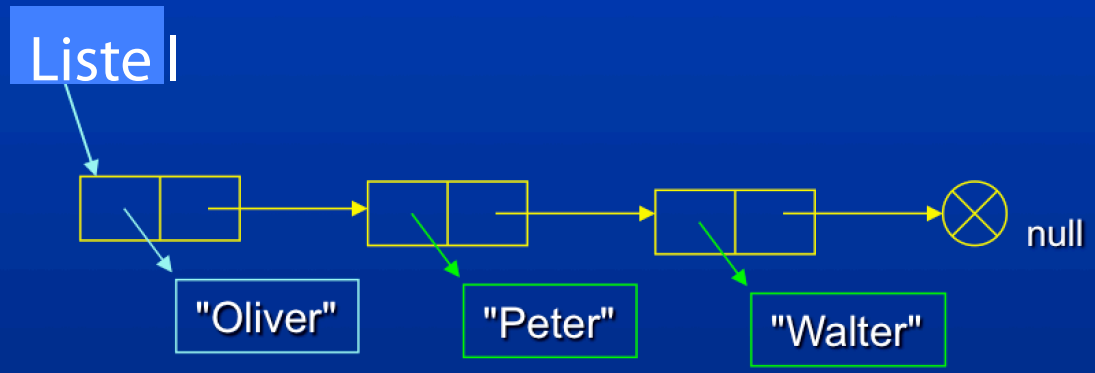
   Insgesamt: O(1)



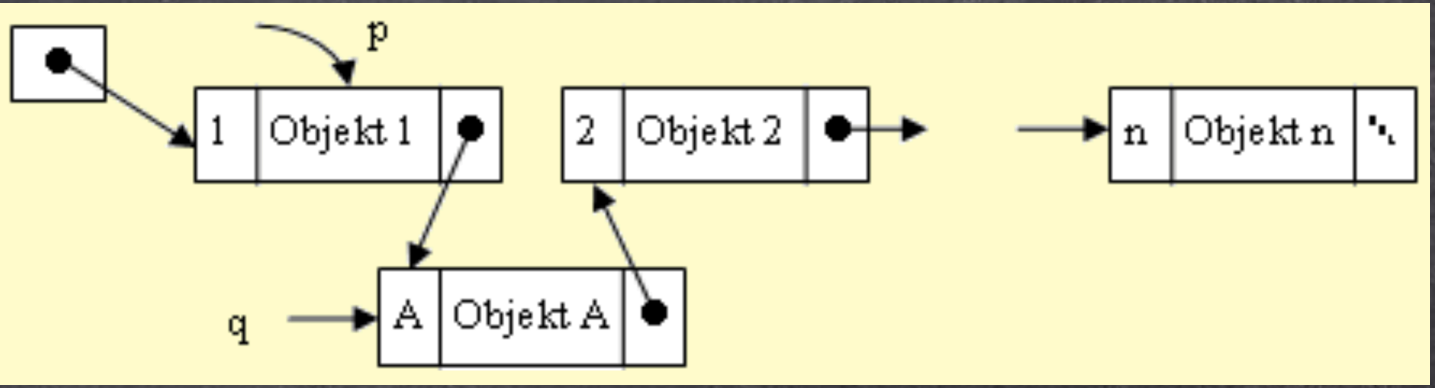
*Wir betrachten im folgenden den average Case

Analyse ausgewählter Listenoperationen

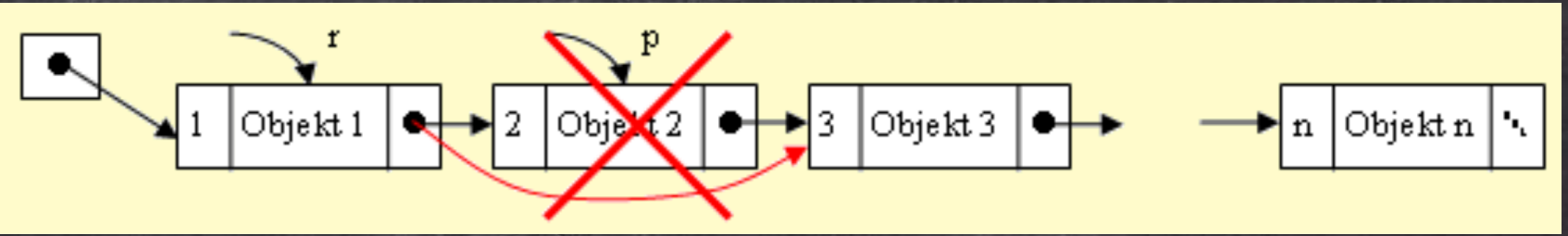
2. **head(): Object**
   //Rückgabe des Objekts am Listenkopf
   Ablauf:
   a. Fallunterscheidung: O(1)
   b. Rückgabe von null: O(1)
   c. Objekt des ersten Listenelements zurückgeben: O(1)
   Insgesamt: O(1)



Einfügen eines Elements in eine verkettete Liste



- Aufwand (im schlimmsten Fall)? O(n)
- Gleiches gilt für's Löschen



Bildquelle: Prof. Scheibl, HTW Berlin

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 = $\sum^{k}_{i=1} S(n_i)$ bei k Eimern

Analyse (2)

- Sei $S(m)=\Theta(m \log m)$
- Falls die Schlüssel gleichmäßig verteilt sind, beträgt die Eimergröße n/k
- Gesamtzahl der Vergleiche

  $= k(n/k) \log(n/k)$

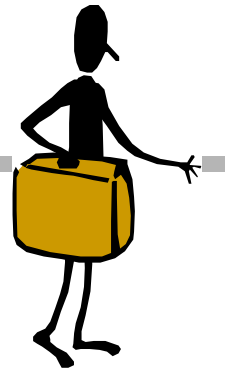
  $= n \log(n/k)$
- Falls $k=n/10$, dann reichen $n \log(10)$ Vergleiche (Laufzeit ist linear in n)

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 **{ P } S { Q }** gilt (Notation nach Hoare)
  - Vorbedingung: **P =?**
  - Invarianten („Axiome“): **I = ?**
  - Nachbedingung: **Q = ∀ 1≤i<j≤n: A[i] ≤ A[j]**
  - Nebenbedingungen: **?**

Zusammenfassung



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