

2.3 Sortieren

- 2.3.1 Einleitung
- 2.3.2 Einfache Sortierverfahren
- 2.3.3 Höhere Sortierverfahren
- 2.3.4 Komplexität von Sortierverfahren
- 2.3.5 Spezielle Sortierverfahren



Quick-Sort

- Idee
 - Divide & Conquer
 - Wähle ein Element R aus
 - Teile Folge in zwei Sub-Folgen
 $R_L \leq R$ und $R_R \geq R$
 - Sortiere R_L und R_R durch rekursiven Aufruf
 - Setze Teilfolgen zusammen: $R_L \oplus R \oplus R_R$



Wähle ein (Pivot-)Element R und löse Problem rekursiv so, dass alle Elemente rechts von R größer und alle Elemente links von R kleiner als R sind. Dabei sind die Teillisten R_L und R_R in sich NICHT sortiert. Im nächsten Schritt wird dann rekursiv dieser Algorithmus auf die Teillisten angewandt.

Quick-Sort

h	k	j	e	n	o	l	d	c	b	i	f	a	g	m
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

3  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort

h	k	j	e	n	o	l	d	c	b	i	f	a	g	m
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

4  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort

h	k	j	e	n	o	l	d	c	b	i	f	a	g	m
d	g	a	e	f	b	c	h	l	o	i	n	j	k	m

 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 **RWTH AACHEN UNIVERSITY**

Quick-Sort

h	k	j	e	n	o	l	d	c	b	i	f	a	g	m
d	g	a	e	f	b	c	h	l	o	i	n	j	k	m

 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 **RWTH AACHEN UNIVERSITY**

Quick-Sort

h	k	j	e	n	o	l	d	c	b	i	f	a	g	m
d	g	a	e	f	b	c	h	l	o	i	n	j	k	m
b	c	a	d	f	e	g	h	j	k	i	l	n	o	m



7 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer



Quick-Sort

h	k	j	e	n	o	l	d	c	b	i	f	a	g	m
d	g	a	e	f	b	c	h	l	o	i	n	j	k	m
b	c	a	d	f	e	g	h	j	k	i	l	n	o	m



8 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer



Quick-Sort

h	k	j	e	n	o	l	d	c	b	i	f	a	g	m
d	g	a	e	f	b	c	h	l	o	i	n	j	k	m
b	c	a	d	f	e	g	h	j	k	i	l	n	o	m
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Das Sortieren hat letztendlich in der Unterteilung der Liste jeweils stattgefunden.

Quick-Sort: Algorithmus

- QuickSort(A[l..r])
 - if $l < r$ then
 - $m \leftarrow \text{Partition}(A[l..r])$
 - QuickSort(A[l..m-1])
 - QuickSort(A[m+1..r])

- Aufruf mit: QuickSort(A[1..n])

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Quick-Sort: Algorithmus

- Partition(A[l..r])
 - i ← l-1; j ← r
 - while i < j do
 - i ← i+1
 - while A[i] < A[r] do
 - i ← i+1
 - j ← j-1
 - while A[j] > A[r] do
 - j ← j-1
 - swap(A[i], A[j])
 - swap(A[i], A[r])
 - return i



Quick-Sort: Algorithmus

- **Achtung:** Verwende Sentinel, um korrekte Terminierung der inneren Schleife zu garantieren.
- Aufruf mit
 - A[0] ← -∞
 - QuickSort(A[1..n])
- while A[i] < A[r] do
 - i ← i+1
- while A[j] > A[r] do
 - j ← j-1

Bricht spätestens für i = r ab

Bricht spätestens für j = l - 1 ab



Sentinel sorgt dafür, dass die j-Schleife nicht über die Grenzen hinaus laufen kann, so dass Terminierung gewährleistet ist.

Quick-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



Quick-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



Quick-Sort: Beispiel

g	a	e	n	o	c	d	l	b	i	f	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

25  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	o	c	d	l	b	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

26  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	o	c	d	l	b	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

27  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	o	c	d	l	b	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

28  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	o	c	d	l	b	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

29  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	l	o	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

30  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	l	o	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

31  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	l	o	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

32  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	l	o	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

33  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	l	o	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i
↑_j

34  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	l	o	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑ ↑

35  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	l	d	o	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑ ↑

36  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	l	o	i	n	j	k	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑ ↑

37  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	h	o	i	n	j	k	m	L
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑ ↑

38  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

39  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

40  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

g	a	e	f	b	c	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

41  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	e	f	b	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

42  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	e	f	b	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

43  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	e	f	b	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

44  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	e	f	b	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

45  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	f	e	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

46  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	f	e	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

47  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	f	e	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i
↑_j

48  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	f	e	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j ↑_i

49  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	f	b	e	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j ↑_i

50  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	f	e	g	d	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j ↑_i

51  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	d	e	g	f	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j ↑_i

52  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	d	e	g	f	h	o	i	n	j	k	m	L
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

53  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	d	e	g	f	h	o	i	n	j	k	m	L
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

54  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	d	e	g	f	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

55  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

a	c	b	d	e	g	f	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_i ↑_j

56  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

a	c	b	d	e	g	f	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j
↑_i

57  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

a	c	b	d	e	g	f	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j ↑_i

58  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

c	a	b	d	e	g	f	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j ↑_i

59  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Beispiel

a	c	b	d	e	g	f	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j ↑_i

60  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

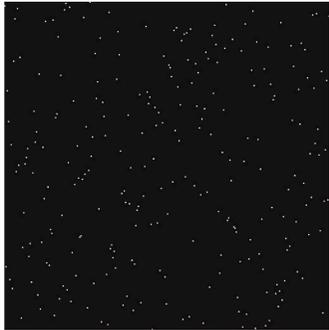
Quick-Sort: Beispiel

a	b	c	d	e	g	f	h	o	i	n	j	k	m	l
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑_j ↑_i

61 Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Quick-Sort: Beispiel



62 Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Es bauen sich quadratähnliche Strukturen auf. Pivotelement ist jeweils Punkt zwischen zwei Quadraten.
Die Quadrate entsprechen dann den R_L und R_R . Dies wird sukzessive ausgeführt.

Quick-Sort: Aufwand

- Aufwand
 - Best Case:
 - $T(n) \approx 2 \times T(n/2) + O(n) \Rightarrow O(n \times \log n)$
 - (Folge zerfällt immer in zwei gleich große Teile)
 - Worst Case:
 - $T(n) \approx T(n-1) + O(n) \Rightarrow O(n^2)$
 - (Folge ist bereits auf- oder absteigend sortiert)
 - Average Case: ...

63
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Worst-Case tritt ein, wenn die Folge bereits sortiert ist. Der Grund ist, dass alle Elemente in (sortierter) Teilliste größer sind als das Pivotelement.

Quick-Sort: Aufwand

- Aufwand
 - Average case

$$T(n) \approx O(n) + \frac{1}{n} \left(\sum_{i=0}^{n-1} T(i) + T(n-1-i) \right)$$

$$= O(n) + \frac{2}{n} \sum_{i=0}^{n-1} T(i)$$

64
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Als erstes: Partitionierung. Diese hat lineare Laufzeit (vgl. Folie 11). Das Sortieren wird dann jeweils auf beide Teillisten ($T(i)$ und $T(n-1-i)$) ausgeführt. $1/n$ kommt daher, dass angenommen wird, dass die Element gleichverteilt sind und das Pivotelement jeden beliebigen Wert aus der unsortierten Teilliste annehmen kann.

Quick-Sort: Aufwand

$$nT(n) = cn^2 + 2 \sum_{i=0}^{n-1} T(i)$$

$$(n-1)T(n-1) = c(n-1)^2 + 2 \sum_{i=0}^{n-2} T(i)$$

$$nT(n) - (n-1)T(n-1) = cn^2 - c(n-1)^2 + 2T(n-1)$$

$$nT(n) = 2cn - c + (n+1)T(n-1)$$

65  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Trick, um das Sigma loszuwerden. Wenn Formel für n gilt, dann auch für (n-1) (zweite Zeile). Subtrahiere diese Formel nun von der oben genannten. Dadurch kommen wir auf die Formel in der unteren Zeile.

Quick-Sort: Aufwand

$$T(n) = c' + \frac{n+1}{n}T(n-1)$$

$$= c' + \frac{n+1}{n}c' + \frac{n+1}{n-1}T(n-2)$$

$$= c' + \frac{n+1}{n}c' + \frac{n+1}{n-1}c' + \frac{n+1}{n-2}T(n-3)$$

$$= c' \left(\frac{n+1}{n+1} + \frac{n+1}{n} + \frac{n+1}{n-1} + \dots + \frac{n+1}{1} \right)$$

$$= c'(n+1) \sum_{i=1}^{n+1} \frac{1}{i}$$

$$\leq c''(n+1) \log(n+1)$$

$$= O(n \log n)$$

66  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Quick-Sort: Aufwand

$$\begin{aligned}
 \sum_{i=1}^{n+1} \frac{1}{i} &\leq 1 + \int_1^{n+1} \frac{1}{x} dx \\
 &= 1 + \ln(x) \Big|_1^{n+1} \\
 &= 1 + \ln(n+1) \\
 &= O(\log(n+1))
 \end{aligned}$$

67
FRIEDRICH-ALEXANDER UNIVERSITÄT

Quick-Sort: Aufwand

- Average Case: $T(n) = O(n \times \log n)$
- Heuristiken zur Verbesserung der Performanz
 - Pivot-Element: $(A[l]+A[r])/2, \dots$
 - Bei kleinen Folgen auf Insertion-Sort wechseln (oder auch nicht: Cache-Kohärenz)
 - Iterative Implementierung mit Stack

68
FRIEDRICH-ALEXANDER UNIVERSITÄT

Wenn man davon ausgeht, dass die Listen schon partiell vorsortiert sind, ist es besser, das Pivotelement jeweils aus der Mitte jeder Teilliste zu wählen (oder als Pivoelement, welches nicht gezwungenerweise in der Liste existieren muss), da dann die Wahrscheinlichkeit höher ist, dass die Elemente davor kleiner und dahinter größer als das Pivotelement sind.

Merge-Sort

- Idee
 - Quick-Sort ist schnell, wenn die beiden Teilfolgen nach Partition() ungefähr die gleiche Größe haben.
 - Teile die Folge **ohne** Vorsortieren in zwei gleiche Teile
 - Sortiere die Teilfolgen
 - Kombiniere die Teilergebnisse



Merge-Sort (rekursiv)

- MergeSort(A[l..r])
 - if $l < r$ then
 - $m \leftarrow (l+r)/2$
 - MergeSort(A[l..m])
 - MergeSort(A[m+1..r])
 - Merge(A[l..m,m+1..r])
- Aufruf mit: MergeSort(A,1,n)

vgl. mit
Quick-Sort



Merge-Sort (rekursiv)

- Merge($A[l..m, m+1..r]$)
 - new $B[l..r-l+1]$; $i \leftarrow 1$; $j \leftarrow l$; $k \leftarrow m+1$
 - while $j \leq m$ and $k \leq r$ do
 - if $A[j] \leq A[k]$ then
 - $B[i++] \leftarrow A[j++]$
 - else
 - $B[i++] \leftarrow A[k++]$
 - while $j \leq m$ do $B[i++] \leftarrow A[j++]$
 - while $k \leq r$ do $B[i++] \leftarrow A[k++]$
 - for $i \leftarrow l$ to r do
 - $A[i] \leftarrow B[i-l+1]$

71

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

A wird unterteilt in zwei Listen: eine von Index l bis m und die andere von m+1 bis r.

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

72

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---



Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

k	j	e	n
---	---	---	---



Merge-Sort: Beispiel

k j e n o c d l b i f a g m h

k j e n o c d l

k j e n

k j

75  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort: Beispiel

k j e n o c d l b i f a g m h

k j e n o c d l

k j e n

k j

k

76  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

k	j	e	n
---	---	---	---

k	j
---	---

k

j

k

j

77

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

k	j	e	n
---	---	---	---

j	k
---	---

k

j

78

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

k	j	e	n
---	---	---	---

j	k
---	---

79

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

k	j	e	n
---	---	---	---

j	k	e	n
---	---	---	---

80

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

k	j	e	n
---	---	---	---

j	k
---	---

e	n
---	---

e

n

81

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

k	j	e	n
---	---	---	---

j	k
---	---

e	n
---	---

e

n

82

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

k	j	e	n
---	---	---	---

j	k
---	---

e	n
---	---

83

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n
---	---	---	---

j	k
---	---

e	n
---	---

84

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n
---	---	---	---



Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n	o	c	d	l
---	---	---	---	---	---	---	---



Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n	o	c	d	l
---	---	---	---	---	---	---	---

o	c
---	---

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

**FRIEDRICH-ALEXANDER
UNIVERSITÄT**

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n	o	c	d	l
---	---	---	---	---	---	---	---

o	c
---	---

o	c
---	---

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

**FRIEDRICH-ALEXANDER
UNIVERSITÄT**

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n	o	c	d	l
---	---	---	---	---	---	---	---

c	o
---	---

o	c
---	---

89

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n	o	c	d	l
---	---	---	---	---	---	---	---

c	o
---	---

90

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n	o	c	d	l
---	---	---	---	---	---	---	---

c	o	d	l
---	---	---	---

d	l
---	---

91

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n	o	c	d	l
---	---	---	---	---	---	---	---

c	o	d	l
---	---	---	---

d	l
---	---

92

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n
---	---	---	---

o	c	d	l
---	---	---	---

c	o
---	---

d	l
---	---

d

l

93

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

k	j	e	n	o	c	d	l
---	---	---	---	---	---	---	---

e	j	k	n
---	---	---	---

o	c	d	l
---	---	---	---

c	o
---	---

d	l
---	---

94

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort: Beispiel

k j e n o c d l b i f a g m h

k j e n o c d l

e j k n c d l o

c o d l



Merge-Sort: Beispiel

k j e n o c d l b i f a g m h

k j e n o c d l

e j k n c d l o



Merge-Sort: Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

c	d	e	j	k	l	n	o
---	---	---	---	---	---	---	---

e	j	k	n
---	---	---	---

c	d	l	o
---	---	---	---

97  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort: Beispiel

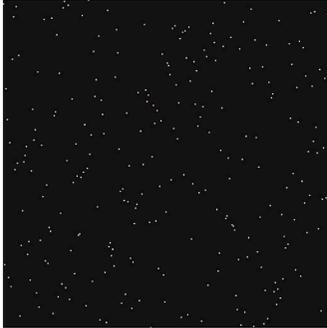
k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

c	d	e	j	k	l	n	o
---	---	---	---	---	---	---	---

usw.

98  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (rekursiv)



99
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Es entstehen immer sortierte Teillisten in Form von „Diagonalen“.

Merge-Sort: Aufwand

- Das Mischen zweier Folgen der Länge L_1 und L_2 erfordert höchstens L_1+L_2 Vergleiche und genau $2 \times (L_1+L_2)$ Kopieroperationen ($i++$ in jedem Schleifendurchlauf)
- Summe aller Folgenlängen auf jeder Rekursionsstufe = n
- Anzahl der Rekursionsstufen = $\log(n)$
- Best = Worst = Average Case = $O(n \times \log n)$
- Aber: Doppelter Speicherbedarf !

100
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Am besten wäre es, wenn der Algorithmus mit nur einem Array anstatt zwei auskommen würde. Besser: Iterative Implementierung.

Merge-Sort (iterativ)

- Die Zerlegung verändert die Folge nicht.
- Die top-down Rekursion steuert nur die Reihenfolge beim Mischen.
- Fasse die unsortierte Liste als Folge von sortierten einelementigen Folgen auf.
- Mische die Teilfolgen bottom-up



Merge-Sort (iterativ)

```
• MergeSort(A[1..n])
  k ← 1
  while k < n do
    i ← 1
    while i + k - 1 < n do
      l ← i
      m ← i + k - 1
      r ← Min(n, i + 2*k - 1)
      Merge(A[ l..m, m+1..r ])
      i ← i + 2*k
    k ← 2*k
```



Nehme an, die gesamte Liste besteht aus n Teillisten der Länge 1. Sortiere Teillisten von immer größer werdender Länge. 1, 2, 4, 8, usw.

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑

j	k
---	---

105  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

↑

j	k	e	n
---	---	---	---

106  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o									

107  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l							

108  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i					

↑

109  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f			

↑

110  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	

↑

111  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	h

↑

112  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Merge-Sort (iterativ): Beispiel

k j e n o c d l b i f a g m h
j k e n c o d l b i a f g m h



Merge-Sort (iterativ): Beispiel

k j e n o c d l b i f a g m h
j k e n c o d l b i a f g m h



Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	h
e	j	k	n											

↑

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	h
e	j	k	n	c	d	l	o							

↑

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	h
e	j	k	n	c	d	l	o	a	b	f	i			

↑

117  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	h
e	j	k	n	c	d	l	o	a	b	f	i	g	h	m

↑

118  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	h
e	j	k	n	c	d	l	o	a	b	f	i	g	h	m



Merge-Sort (iterativ): Beispiel

k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	h
e	j	k	n	c	d	l	o	a	b	f	i	g	h	m



Merge-Sort (iterativ): Beispiel

k j e n o c d l b i f a g m h

j k e n c o d l b i a f g m h

e j k n c d l o a b f i g h m

↑
c d e j k l n o

121  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort (iterativ): Beispiel

k j e n o c d l b i f a g m h

j k e n c o d l b i a f g m h

e j k n c d l o a b f i g h m

c d e j k l n o ↑
a b f g h i m

122  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Merge-Sort (iterativ): Beispiel

k j e n o c d l b i f a g m h

j k e n c o d l b i a f g m h

e j k n c d l o a b f i g h m

c d e j k l n o a b f g h i m

123



Datenstrukturen und Algorithmen

Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT

Merge-Sort (iterativ): Beispiel

k j e n o c d l b i f a g m h

j k e n c o d l b i a f g m h

e j k n c d l o a b f i g h m

c d e j k l n o a b f g h i m

↑
a b c d e f g h i j k l m n o

124



Datenstrukturen und Algorithmen

Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT

Merge-Sort (iterativ): Beispiel

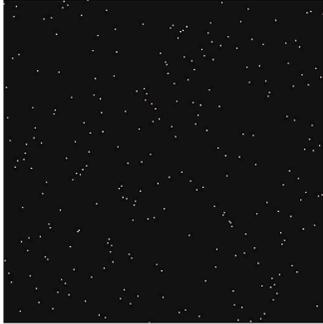
k	j	e	n	o	c	d	l	b	i	f	a	g	m	h
j	k	e	n	c	o	d	l	b	i	a	f	g	m	h
e	j	k	n	c	d	l	o	a	b	f	i	g	h	m
c	d	e	j	k	l	n	o	a	b	f	g	h	i	m
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o

125

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Merge-Sort (iterativ)



126

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Der Algorithmus spart sich die Rekursion und fängt direkt in der tiefsten Ebene an. Es findet eine Breiten- anstatt einer Tiefensuche (wie im rekursiven Ansatz) statt.

Heap-Sort

- Idee
 - Bei Selection-Sort entsteht der $O(n^2)$ -Aufwand durch die lineare Suche nach dem kleinsten (oder größten) Element in der Rest-Liste.
 - Mit einer Prioritätsschlange (Heap) kann man das kleinste Element schneller finden ($O(\log n)$ -Aufwand zum Wiederherstellen der Heap-Bedingung).



Heap-Definition (reprise)

- (Links-)vollständiger Binärbaum in Array-Darstellung
 - Vaterknoten $A[i]$ hat die Nachfolger $A[2 \times i]$ und $A[2 \times i + 1]$
- Heap-Bedingung
 - Node $\geq \max(\text{Left-Subtree})$ und Node $\geq \max(\text{Right-Subtree})$
 - $A[i] \geq A[2 \times i]$ und $A[i] \geq A[2 \times i + 1]$



Heap-Definition (reprise)

- (Links-)vollständiger Binärbaum in Array-Darstellung
 - Vaterknoten $A[i]$ hat die Nachfolger $A[2 \times i]$ und $A[2 \times i + 1]$
- Heap-Bedingung
 - Node $\geq \max(\text{Left-Subtree})$ und Node $\geq \max(\text{Right-Subtree})$
 - $A[\lfloor i/2 \rfloor] \geq A[i]$



Heap-Definition (reprise)

- Ein Array A hat die Heap-Eigenschaft ab Index p , wenn $A[\lfloor i/2 \rfloor] \geq A[i]$ ab Index $2 \times p$ gilt (weil dann $\lfloor i/2 \rfloor = p$).
- Insbesondere: jedes beliebige Array $A[1..n]$ hat die Heap-Eigenschaft ab Index $\lfloor n/2 \rfloor + 1$.



In der zweiten Hälfte des Arrays sind nur Blattknoten, für die die Bedingung egal ist.

Heap-Sort

- Konvertiere ein beliebiges Array in einen Heap ($A[\lfloor i/2 \rfloor] \geq A[i]$).
- Entferne sukzessive das Top-Element und stelle die Heap-Eigenschaft wieder her (vgl. Deq()-Operation für Prioritätsschlangen in Abschnitt 1.4).
- Heap liefert sortierte Elemente.



Heap-Sort

- Sei $A[1..n]$ ein Heap, Top-Element $A[1]$
- Nach dem Entfernen des Top-Elementes wird $A[n]$ nach $A[1]$ kopiert und durch „Versickern“ die Heap-Eigenschaft wieder hergestellt.
- Danach benutzt der Heap noch die Array-Elemente $A[1..n-1]$.
- $A[n]$ ist also frei und das entfernte Top-Element kann dort abgelegt werden.



Bei dieser Implementierung wird kein zusätzlicher Speicher gebraucht, da die sortierten Elemente jeweils ans Ende des Arrays geschrieben werden.

Heap-Sort: Algorithmus

- ConvertToHeap(A[1..n])
 for $i = \lfloor n/2 \rfloor$ downto 1 do
 Sink(A[1..n],i)
- Wenn die Schleife bis $i=p$ durchgelaufen ist, hat das Array A ab Index p die Heap-Eigenschaft.

133  Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Heap-Sort: Algorithmus

- Sink(A[1..n],i)
 while $i \leq \lfloor n/2 \rfloor$ do
 $j \leftarrow 2*i$
 if $j < n$ and $A[j+1] > A[j]$ then
 $j \leftarrow j+1$
 if $A[j] > A[i]$ then
 Swap(A[j],A[i])
 $i \leftarrow j$
 else
 $i \leftarrow n$

134  Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Heap-Sort: Algorithmus

- HeapSort(A[1..n])
 - ConvertToHeap(A[1..n])
 - for i ← n downto 1 do
 - Swap(A[1],A[i])
 - Sink(A[1..i-1],1)

135

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Im Prinzip ist Heap-Sort eine intelligente Variante des Selection-Sort. Man wählt immer das größte/kleinste Element der unsortierten Teilliste und hängt es an die entsprechende Stelle im sortierten Teil der Liste.

Heap-Sort: Beispiel

11	7	27	19	35	44	14	41
----	---	----	----	----	----	----	----

136

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Heap-Sort: Beispiel

Phase I: Aufbau des Heaps

11	7	27	19	35	44	14	41
----	---	----	----	----	----	----	----

```

graph TD
    11[11] --- 7[7]
    11 --- 27[27]
    7 --- 19[19]
    7 --- 35[35]
    19 --- 41[41]
    27 --- 44[44]
    27 --- 14[14]
  
```

137 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer FRIEDRICH-ALEXANDER
UNIVERSITÄT

Heap-Sort: Beispiel

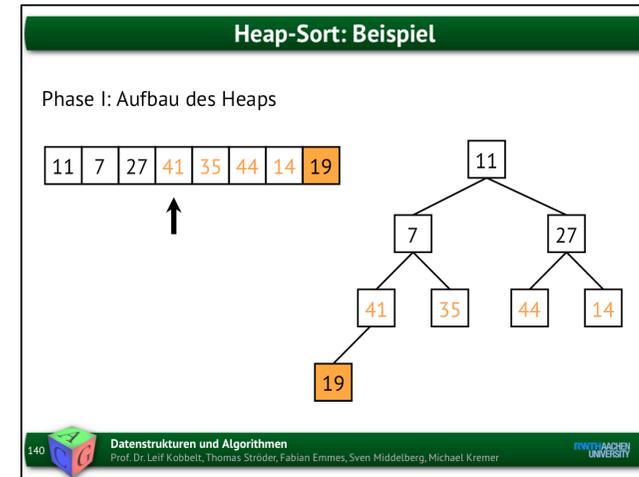
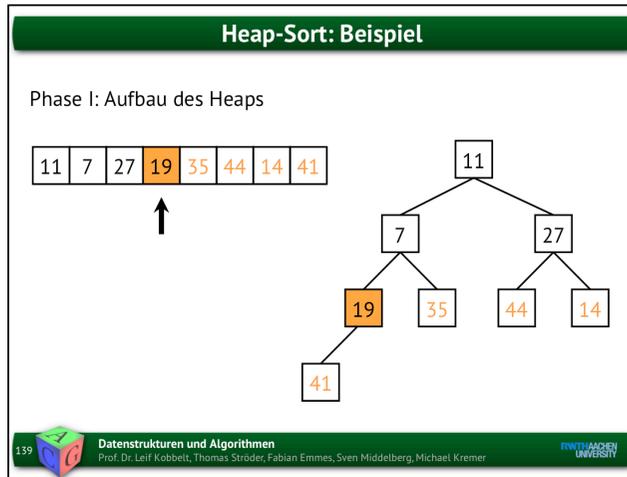
Phase I: Aufbau des Heaps

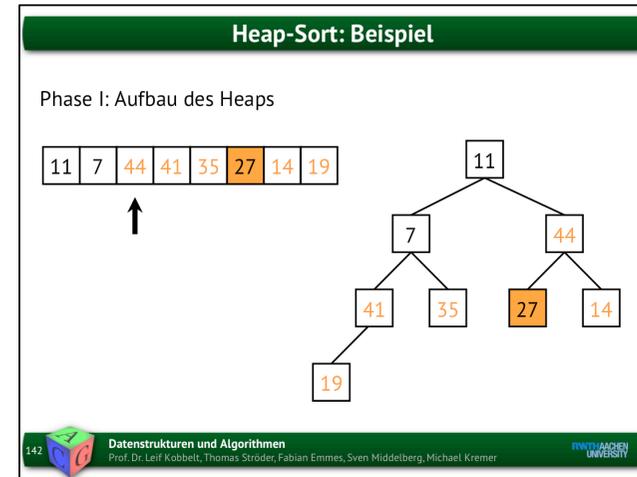
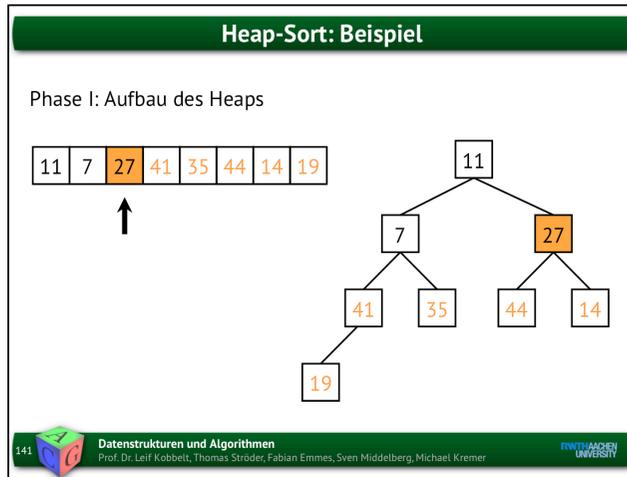
11	7	27	19	35	44	14	41
----	---	----	----	----	----	----	----

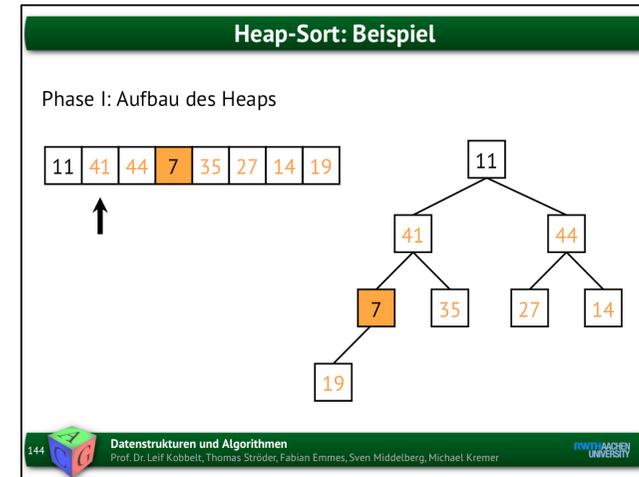
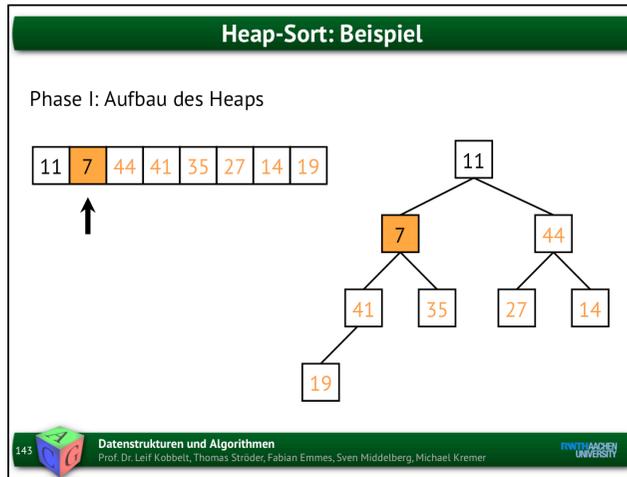
```

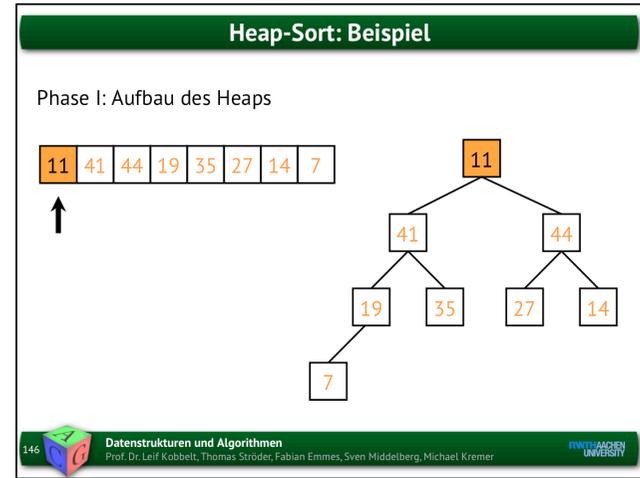
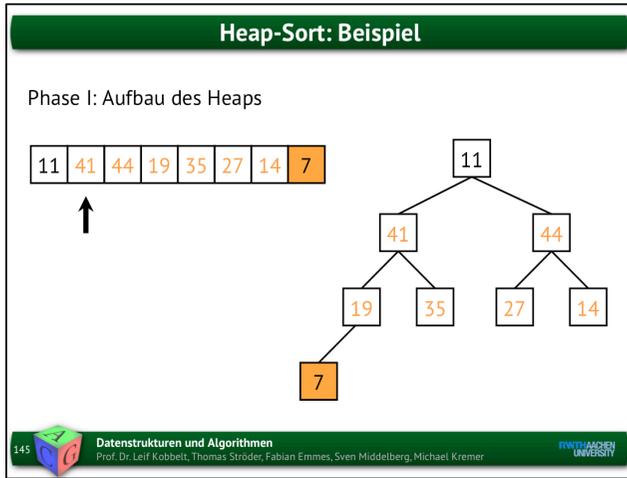
graph TD
    11[11] --- 7[7]
    11 --- 27[27]
    7 --- 19[19]
    7 --- 35[35]
    19 --- 41[41]
    27 --- 44[44]
    27 --- 14[14]
  
```

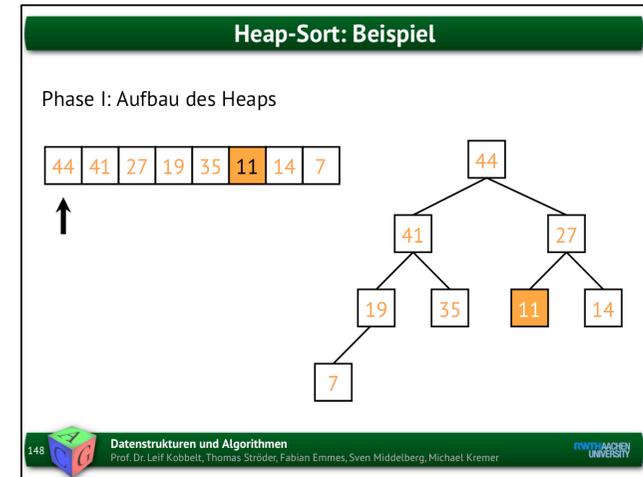
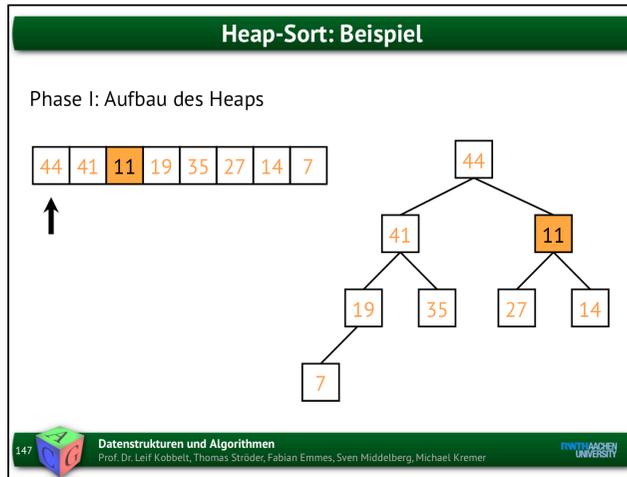
138 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer FRIEDRICH-ALEXANDER
UNIVERSITÄT











Heap-Sort: Beispiel

Phase I: Aufbau des Heaps

44	41	27	19	35	11	14	7
----	----	----	----	----	----	----	---

```

graph TD
    44 --> 41
    44 --> 27
    41 --> 19
    41 --> 35
    27 --> 11
    27 --> 14
    19 --> 7
          
```

149

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Heap-Sort: Beispiel

Phase II: Selection-Sort

44	41	27	19	35	11	14	7
----	----	----	----	----	----	----	---

```

graph TD
    44 --> 41
    44 --> 27
    41 --> 19
    41 --> 35
    27 --> 11
    27 --> 14
    19 --> 7
          
```

150

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Heap-Sort: Beispiel

Phase II: Selection-Sort

7	41	27	19	35	11	14	44
---	----	----	----	----	----	----	----

```

graph TD
    7[7] --- 41[41]
    7 --- 27[27]
    41 --- 19[19]
    41 --- 35[35]
    27 --- 11[11]
    27 --- 14[14]
            
```

151

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Heap-Sort: Beispiel

Phase II: Selection-Sort

41	7	27	19	35	11	14	44
----	---	----	----	----	----	----	----

```

graph TD
    41[41] --- 7[7]
    41 --- 27[27]
    7 --- 19[19]
    7 --- 35[35]
    27 --- 11[11]
    27 --- 14[14]
            
```

152

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Heap-Sort: Beispiel

Phase II: Selection-Sort

41	35	27	19	7	11	14	44
----	----	----	----	---	----	----	----

```

graph TD
    41 --> 35
    41 --> 27
    35 --> 19
    35 --> 7
    27 --> 11
    27 --> 14
            
```

153

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Heap-Sort: Beispiel

Phase II: Selection-Sort

41	35	27	19	7	11	14	44
----	----	----	----	---	----	----	----

```

graph TD
    41 --> 35
    41 --> 27
    35 --> 19
    35 --> 7
    27 --> 11
    27 --> 14
            
```

154

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Heap-Sort: Beispiel

Phase II: Selection-Sort

14	35	27	19	7	11	41	44
----	----	----	----	---	----	----	----

155

Datenstrukturen und Algorithmen

Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Heap-Sort: Beispiel

Phase II: Selection-Sort

35	14	27	19	7	11	41	44
----	----	----	----	---	----	----	----

156

Datenstrukturen und Algorithmen

Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Heap-Sort: Beispiel

Phase II: Selection-Sort

35	19	27	14	7	11	41	44
----	----	----	----	---	----	----	----

```

graph TD
    35 --> 19
    35 --> 27
    19 --> 14
    19 --> 7
    27 --> 11
    style 14 fill:#f4a460
          
```

157

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Heap-Sort: Beispiel

Phase II: Selection-Sort

35	19	27	14	7	11	41	44
----	----	----	----	---	----	----	----

```

graph TD
    35 --> 19
    35 --> 27
    19 --> 14
    19 --> 7
    27 --> 11
    style 27 fill:#f4a460
          
```

158

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Heap-Sort: Beispiel

Phase II: Selection-Sort

11	19	27	14	7	35	41	44
----	----	----	----	---	----	----	----

159 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer FORTHAAGEN UNIVERSITY

Heap-Sort: Beispiel

Phase II: Selection-Sort

27	19	11	14	7	35	41	44
----	----	----	----	---	----	----	----

160 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer FORTHAAGEN UNIVERSITY

Heap-Sort: Beispiel

Phase II: Selection-Sort

27	19	11	14	7	35	41	44
----	----	----	----	---	----	----	----

161

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 FRIEDRICH-SCHILLER
 UNIVERSITÄT

Heap-Sort: Beispiel

Phase II: Selection-Sort

7	19	11	14	27	35	41	44
---	----	----	----	----	----	----	----

162

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 FRIEDRICH-SCHILLER
 UNIVERSITÄT

Heap-Sort: Beispiel

Phase II: Selection-Sort

19	7	11	14	27	35	41	44
----	---	----	----	----	----	----	----

```

graph TD
    19[19] --> 7[7]
    19 --> 11[11]
    7 --> 14[14]
            
```

163

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Heap-Sort: Beispiel

Phase II: Selection-Sort

19	14	11	7	27	35	41	44
----	----	----	---	----	----	----	----

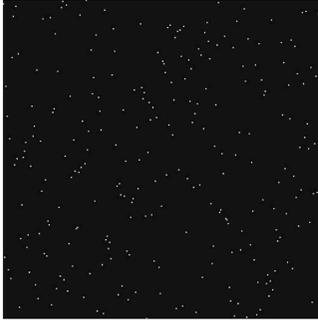
```

graph TD
    19[19] --> 14[14]
    19 --> 11[11]
    14 --> 7[7]
            
```

164

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Heap-Sort: Beispiel



165  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Heap-Sort: Aufwand

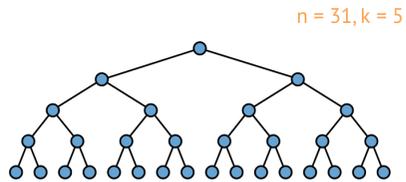
- Aufwand
 - Aufwand zum Aufbau des Heaps
 - Aufwand Selection-Sort

166  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Der Algorithmus hat zwei Phasen: Aufbau und Abbau (sortieren). Bei der Komplexitätsanalyse müssen natürlich beide Teile berücksichtigt werden.

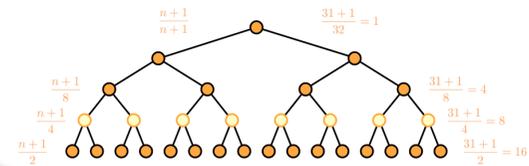
Aufwand: ConvertToHeap()

- ConvertToHeap()
 - Sink()-Operation = O(Höhe des Heap)
 - Sei $n = 2^k - 1$...



Aufwand: ConvertToHeap()

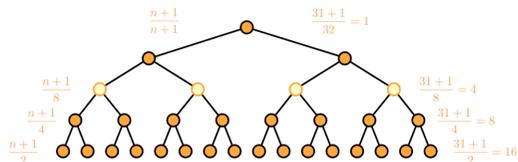
$$\begin{aligned}
 T_1(n) &= c \left(1 \frac{n+1}{4} + 2 \frac{n+1}{8} + \dots + (\log(n+1) - 1) \frac{n+1}{n+1} \right) \\
 &= c \left(1 \frac{n+1}{4} + \dots + i \frac{n+1}{2^{i+1}} + \dots + (k-1) \frac{n+1}{2^k} \right) \\
 &= \frac{c}{2} (n+1) \sum_{i=1}^{k-1} \frac{i}{2^i}
 \end{aligned}$$



Für die Blätter müssen wir nichts machen. Für die nächsten $(n+1)/4$ muss man maximal eine Sink()-Operation ausführen. Danach zwei, usw. Für jedes Element (bis auf die Blätter) muss man maximal $\log n$ Operationen ausführen.

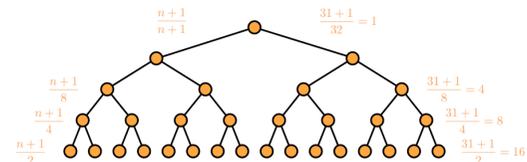
Aufwand: ConvertToHeap()

$$\begin{aligned}
 T_1(n) &= c \left(1 \frac{n+1}{4} + 2 \frac{n+1}{8} + \dots + (\log(n+1) - 1) \frac{n+1}{n+1} \right) \\
 &= c \left(1 \frac{n+1}{4} + \dots + i \frac{n+1}{2^{i+1}} + \dots + (k-1) \frac{n+1}{2^k} \right) \\
 &= \frac{c}{2} (n+1) \sum_{i=1}^{k-1} \frac{i}{2^i}
 \end{aligned}$$



Aufwand: ConvertToHeap()

$$\begin{aligned}
 T_1(n) &= c \left(1 \frac{n+1}{4} + 2 \frac{n+1}{8} + \dots + (\log(n+1) - 1) \frac{n+1}{n+1} \right) \\
 &= c \left(1 \frac{n+1}{4} + \dots + i \frac{n+1}{2^{i+1}} + \dots + (k-1) \frac{n+1}{2^k} \right) \\
 &= \frac{c}{2} (n+1) \sum_{i=1}^{k-1} \frac{i}{2^i}
 \end{aligned}$$



Aufwand: ConvertToHeap()

$$\begin{aligned} T_1(n) &\leq \frac{c}{2}(n+1) \sum_{i=1}^{k-1} \frac{i}{2^i} \\ &= \frac{c}{2}(n+1) \underbrace{\left(2 - \frac{k+1}{2^{k-1}}\right)}_{<2} \\ &= O(n) \end{aligned}$$

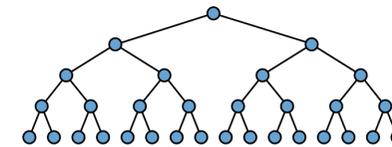


„Für viele Elemente passiert wenig und für wenige Elemente passiert viel“. Aufbauen des Heaps in $O(n)$.

Aufwand: Selection-Sort

- Selection-Sort
 - Kosten für die Wiederherstellung der Heap-Eigenschaft = $O(\text{Höhe des restlichen Heaps})$
 - Sei $n = 2^k - 1 \dots$

$n = 31, k = 5$



Wiederherstellen der Heap-Eigenschaft geschieht allerdings in $O(\log n)$. Von daher ist die Gesamtkomplexität des Heap-Sort-Algorithmus $O(n * \log n)$.

Aufwand: Selection-Sort

- Selection-Sort
 - Kosten für die Wiederherstellung der Heap-Eigenschaft = $O(\text{Höhe des restlichen Heaps})$
 - Sei $n = 2^k - 1 \dots$

173 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Aufwand: Selection-Sort

- Selection-Sort
 - Kosten für die Wiederherstellung der Heap-Eigenschaft = $O(\text{Höhe des restlichen Heaps})$
 - Sei $n = 2^k - 1 \dots$

174 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Aufwand: Selection-Sort

- Selection-Sort
 - Kosten für die Wiederherstellung der Heap-Eigenschaft = $O(\text{Höhe des restlichen Heaps})$
 - Sei $n = 2^k - 1 \dots$

175
FRONTMÄCHER
UNIVERSITY

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Aufwand: Selection-Sort

- Selection-Sort
 - Kosten für die Wiederherstellung der Heap-Eigenschaft = $O(\text{Höhe des restlichen Heaps})$
 - Sei $n = 2^k - 1 \dots$

176
FRONTMÄCHER
UNIVERSITY

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Aufwand: SelectionSort

$$\begin{aligned}
 T_2(n) &\leq c \left((k-1) \frac{n+1}{2} + (k-2) \frac{n+1}{4} + \dots + 0 \frac{n+1}{n+1} \right) \\
 &= c \left((k-1) \frac{n+1}{2} + \dots + (k-i) \frac{n+1}{2^i} + \dots \right) \\
 &= c(n+1) \sum_{i=1}^{k-1} (k-i) 2^{-i} \\
 &= c(n+1) \sum_{i=1}^{k-1} i 2^{i-k} \\
 &= c(n+1) \frac{1}{2^k} \sum_{i=1}^{k-1} i 2^i \\
 &= c(n+1) \frac{1}{2^k} (2^k(k-2) + 2) \\
 &= c(n+1) \left(k-2 + \frac{1}{2^{k-1}} \right) \\
 &= O(n \log n)
 \end{aligned}$$

177  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

k ist die Höhe des Binärbaumes, also $k = \log n$.

Aufwandsabschätzung

- Insgesamt ... worst / average case

$$\begin{aligned}
 T(n) &= T_1(n) + T_2(n) \\
 &= O(n) + O(n \times \log n) \\
 &= O(n \times \log n)
 \end{aligned}$$
- Best case: $O(n)$... alle Elemente gleich
- Vorsortierung wird nicht ausgenutzt

178  **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer 

Best-Case: Es gibt kein Versinken, weil Heap-Eigenschaft immer erfüllt ist.

Vergleich		
	Average	Worst
Quick-Sort	$O(n \times \log n)$	$O(n^2)$
Merge-Sort (doppelter Speicher)	$O(n \times \log n)$	$O(n \times \log n)$
Heap-Sort	$O(n \times \log n)$	$O(n \times \log n)$

179  **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer  **RWTH AACHEN UNIVERSITY**