
Vorlesung 7

Vorlesung7

- Lineare Suche (Vorlesung6)
- Binäre Suche
 - Einführung (Vorlesung6)
 - Implementierung
 - Ansatzgebiet.
- Sortieralgorithmus
- Speicher (Bereiche)

Suchen in Datenmengen

Suchen in Datenmengen

- Motivation:

Suchen in Datenbanken, in Wörterbüchern, im WWW, ...

Suchen in Datenmengen

- Motivation:

„Warum soll mich das interessieren?“

Suchen ist „das Wichtigste“ !

Lineare Suche oder Sequentielle Suche

„Naives verfahren“

- **Idee:** Durchlaufe das Array A von vorn nach hinten und vergleiche jede Element des Arrays mit dem gesuchten Element s , solange bis s gefunden wird. (Vorlesung 6)

Lineare Suche

- **Average Case:** Annahme: jede Anordnung ist gleichwahrscheinlich:

$$C_{avg}(n) = \frac{1}{n} \sum_{i=1}^n i = \frac{n+1}{2}$$

Lineare Suche

- **Diskussion:**

lange Suchzeit → nur für kleine n empfehlenswert
einfache Methode auch für einfache Listen
geeignet.

Sortieren

Motivation

- Sortieren von Datensätzen nach verschiedenen Kriterien oder zum Ordnen von Tabellen oder für schnellen Zugriff über binäres Suchen gehören zum täglichen Brot.
- Zudem gehören Sortierverfahren neben Suchverfahren mit zu den am besten und ausführlichsten untersuchten Algorithmen in der Informatik.

Algorithmen: Sortierung

- Aufgabe:

Die Folgenden Zahlen sollen sortiert werden:

Am Anfang: 13 6 27 9 1

Am Ende: 1 6 9 13 27

Algorithmen: Sortierung

Lösungsweg1:

- Sortieren durch direktes Einfügen:

Beschreibung des Verfahren:

1. Das zu sortierende Array wird in einen linken und einen rechten Teil getrennt. Der linke Teil ist eine bereits sortierte Sequenz, in die die Elemente des rechten Teils nacheinander einsortiert werden müssen.

Algorithmen: Sortierung

- Sortieren durch direktes Einfügen:

Beschreibung des Verfahren:

2. Zu Beginn:

- Linker Teil ist bereits sortiert und besteht aus dem ersten Element des Arrays.
- Alle anderen Elemente gehören zum unsortierten rechten Teil.

Algorithmen: Sortierung

- Sortieren durch direktes Einfügen:

Beschreibung des Verfahren:

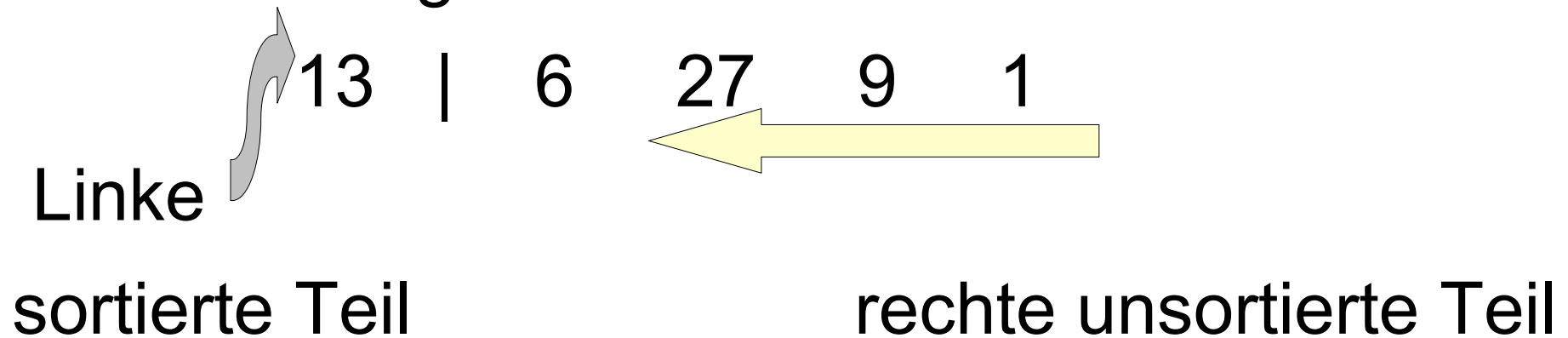
3. Solange der rechte Teil nicht leer ist wird das jeweils erste Element des rechten Teils in die sortierte Sequenz eingefügt:

Einfügen heißt dabei ,das Element solange mit seinem linken Nachbarn zu vertauschen bis dieser entweder kleiner ist oder der linke Rand des Arrays erreicht wird.

Algorithmen: Sortierung

- Sortieren durch direktes Einfügen:

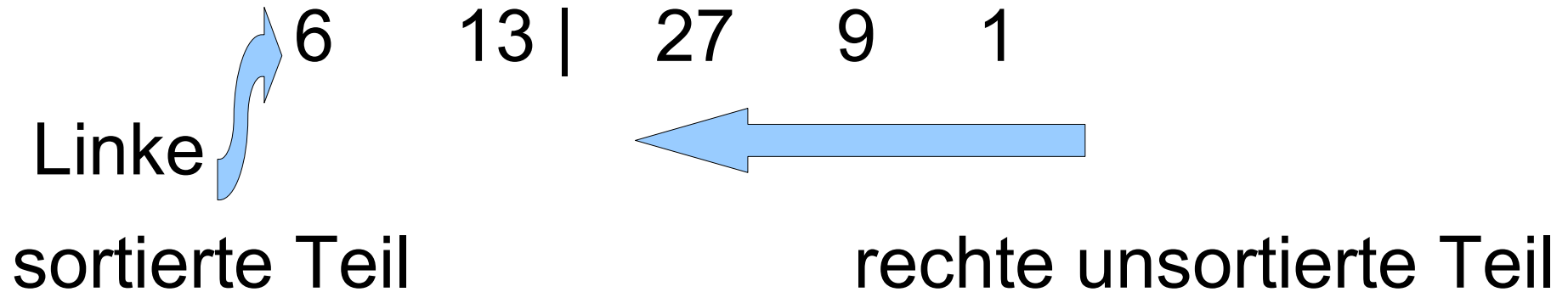
Beschreibung des Verfahren:



Algorithmen: Sortierung

- Sortieren durch direktes Einfügen:

Einfügen und Austauschen



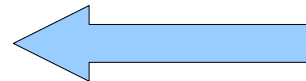
Algorithmen: Sortierung

- Sortieren durch direktes Einfügen:

Einfügen und Austauschen

6 13 27 | 9 1

Linke



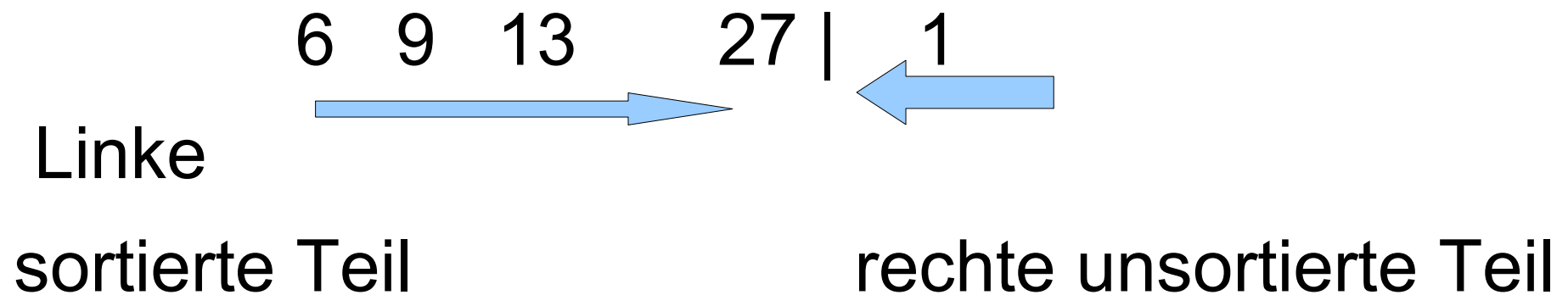
sortierte Teil

rechte unsortierte Teil

Algorithmen: Sortierung

- Sortieren durch direktes Einfügen:

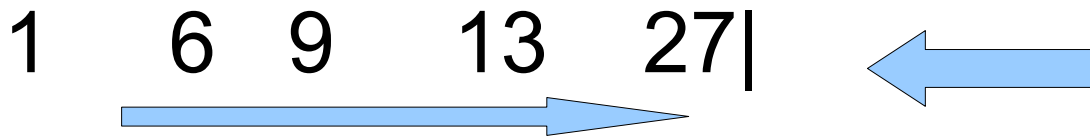
Einfügen und Austauschen



Algorithmen: Sortierung

- Sortieren durch direktes Einfügen:

Einfügen und Austauschen



Linke

sortierte Teil

rechte unsortierte Teil

vorlesung7_1.cpp

Binäre Suche

- Voraussetzung: die Liste ist bereits sortiert.

$$A[0] \leq A[1] \leq \dots \leq A[n-1]$$

Binäre Suche/Idee

- **Divide-and-Conquer:**

Vergleiche das Suchelement s mit dem Element in der Mitte m

- **Falls $A[m] == s$? Treffer \rightarrow STOP**
- **Falls s ist kleiner:**
 - **Durchsuche Elemente links von m .**
- **Falls s ist größer**
 - **Durchsuche Elemente rechts von m .**

Binäre Suche: Effizienz

- **Binäre Suche.**

Größe der Eingabe: Ein Array mit n Elementen.

Die Länge des Intervalls [unten, oben] wird nach jedem Durchlauf durch 2 geteilt. Deshalb endet die Schleife nach **$\log_2(n)$** .

Deshalb ist die Effizienz proportional zum Logarithmus der Anzahl der Menge. Das heißt **$O(\log(n))$**

Binäre Suche /Diskussion.

- Binäre Suche ist für Grosse Datenmenge sehr Empfehlenswert.
- Binäre Suche ist nur Sinnvoll, wenn sich die Daten nicht allzu oft ändern