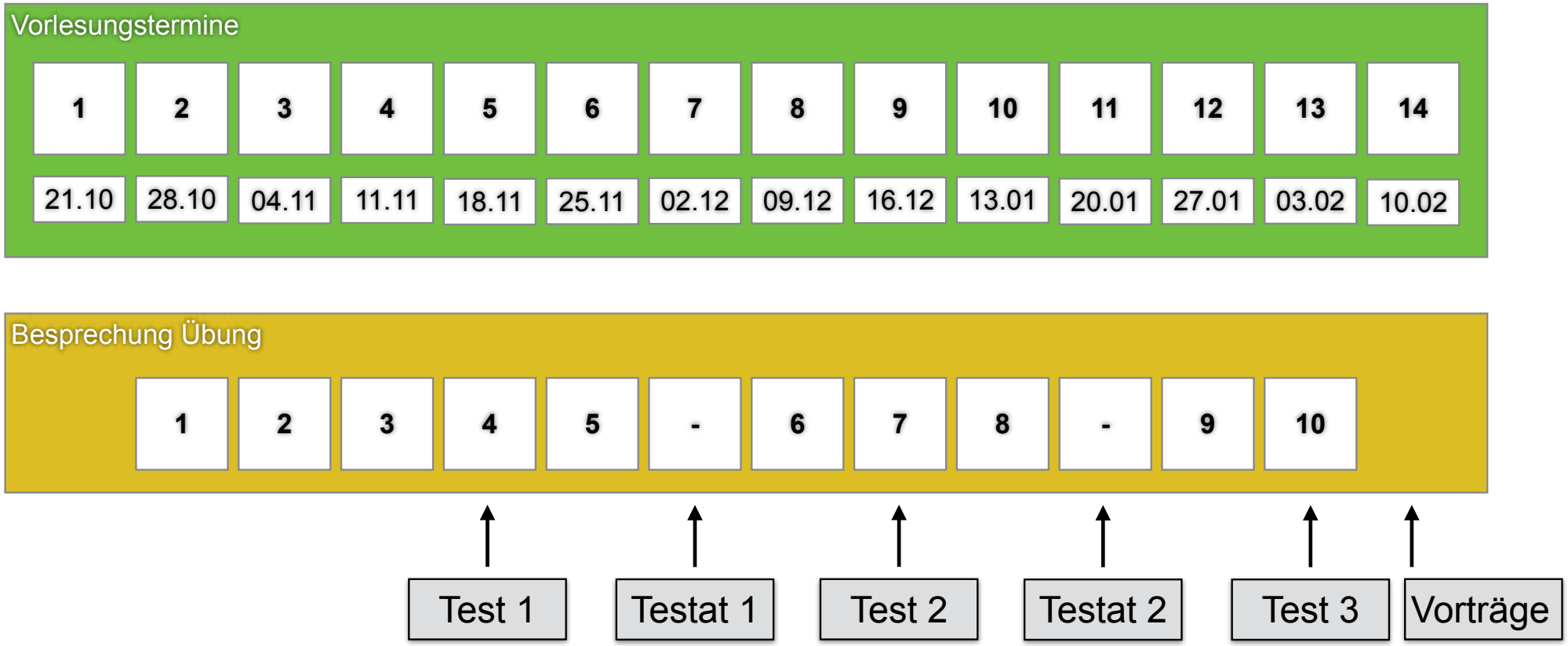
Algorithmen und Datenstrukturen

Prof. Dr. Thomas Wiemann - FB AI

Hochschule Fulda
University of Applied Sciences

Ablaufplan Vorlesung und Portfolioleistungen



Inhalt

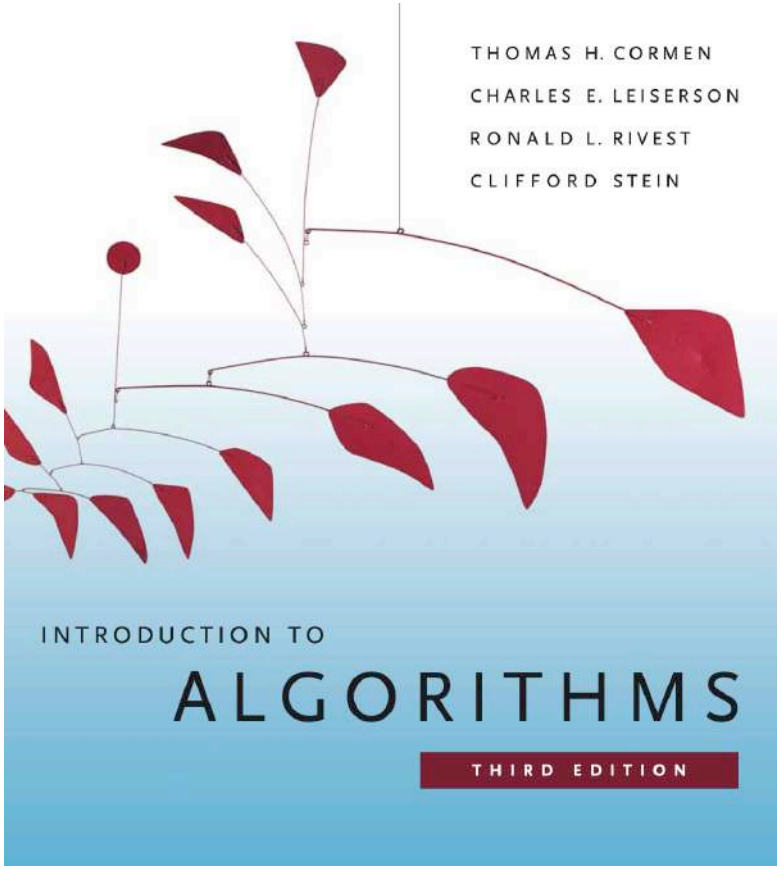
Gliederung

1. Laufzeit und Komplexität
2. **Sortieren**
3. Abstrakte Datentypen
4. Hashing
5. Suchbäume
6. Graphen- und Graphenalgorithmen
7. Ausblick

2. Sortieren

1. Simple Sortierverfahren
2. Komplexe Sortierverfahren
3. Untere Schranke für vergleichsbasiertes Sortieren
4. **Sortieren in linearer Zeit**

Inhalt



Gliederung

1. Laufzeit und Komplexität
2. Sortieren
3. **Abstrakte Datentypen**
4. Hashing
5. Suchbäume
6. Graphen- und Graphenalgorithmen
7. Ausblick

https://edutechlearners.com/download/Introduction_to_algorithms-3rd Edition.pdf

Abstrakte Datentypen

- Viele Programmiersprachen haben verschiedene “Container”, Daten zu speichern
- Ein Abstrakter Datentyp (**ADT**) besteht aus einer Datenstruktur und darauf definierten Operationen
- Die Menge der zur Verfügung gestellten Operationen bildet die **Schnittstelle** (Interface)
- Eine Datenstruktur ist die **Implementierung** eines ADT
- Es gibt in nahezu allen Objektorientierten Programmiersprachen Standart-ADTs in unterschiedlichen Ausprägungen
- Hier die wichtigsten anhand vereinfachter Beispiele
- Beispiel

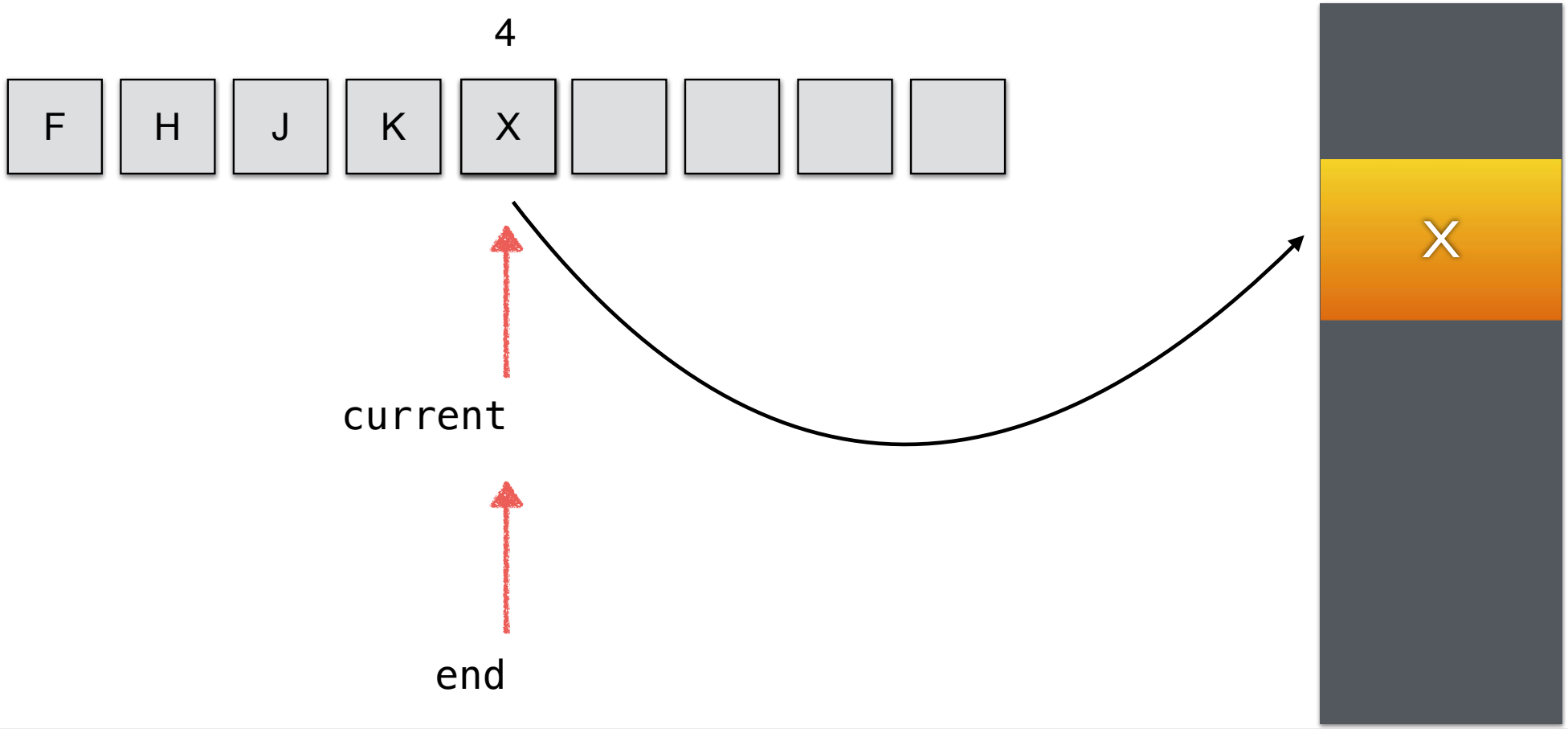
ADT Liste

- Lineare Liste von Objekten, die hintereinander angeordnet sind
- Ein Zeiger `begin` auf den Beginn der Liste, ein Zeiger `current` auf eine Position in der Liste
- Elemente werden bei `current` eingefügt und gelöscht
- Man kann an den Anfang der Liste springen und jeweils ein Element weiter gehen

```
public interface List {
    public boolean empty();          // Checks if list is empty
    public boolean end();            // Checks if the end of the list is reached
    public void reset();             // Resets the iterator
    public void advance();           // Advances the iterator
    public Object current();         // Returns the object stored at the current position
    public Object at(int i);         // Returns the object at the i-th position
    public void insert(Object x);    // Inserts x into the list at the current position
    public void delete();            // Delete the element at the current position
    public void delete(Object x);    // Delete object x
    public boolean find(Object x);   // Check if x is in the list
}
```

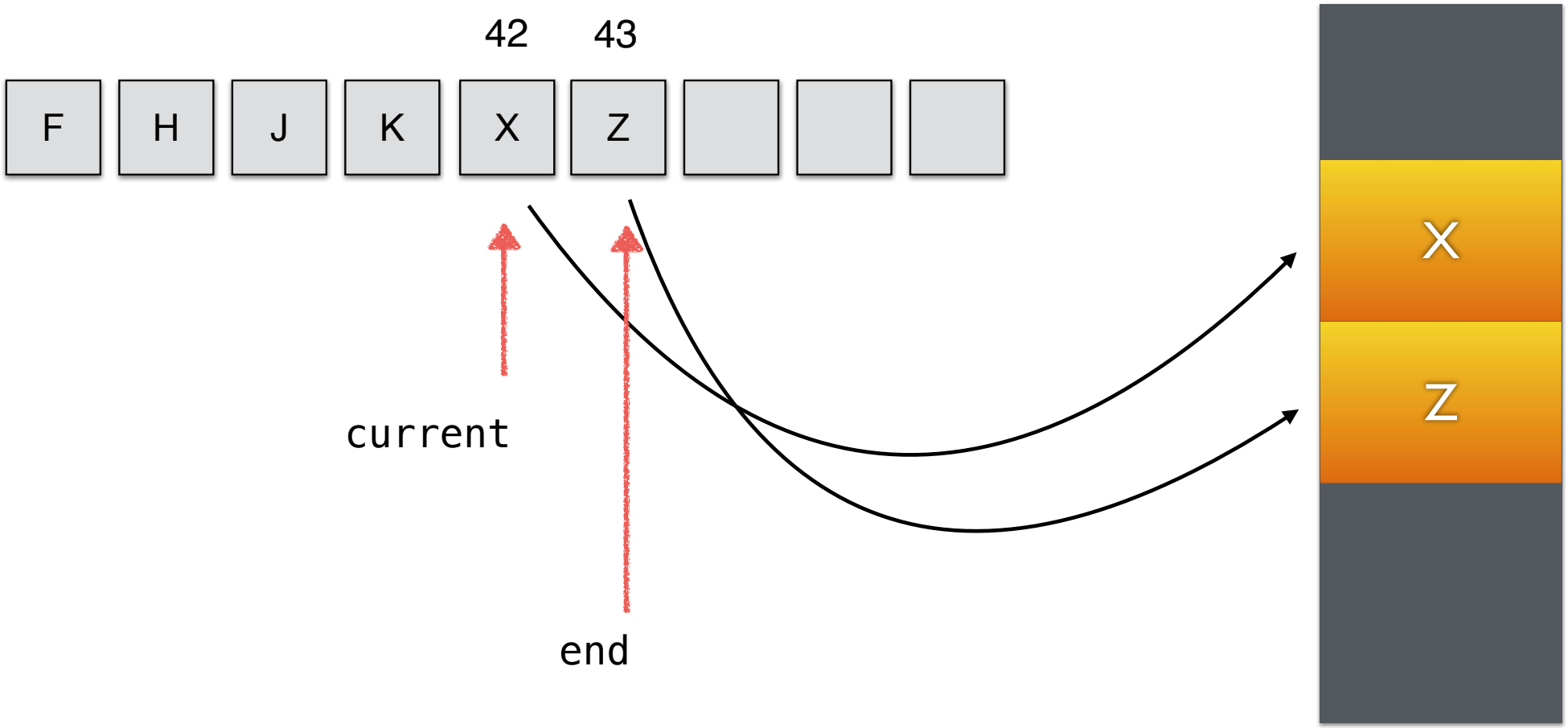
Liste mit Array

- Lässt sich eine Liste mit einem Array als Datenstruktur implementieren?



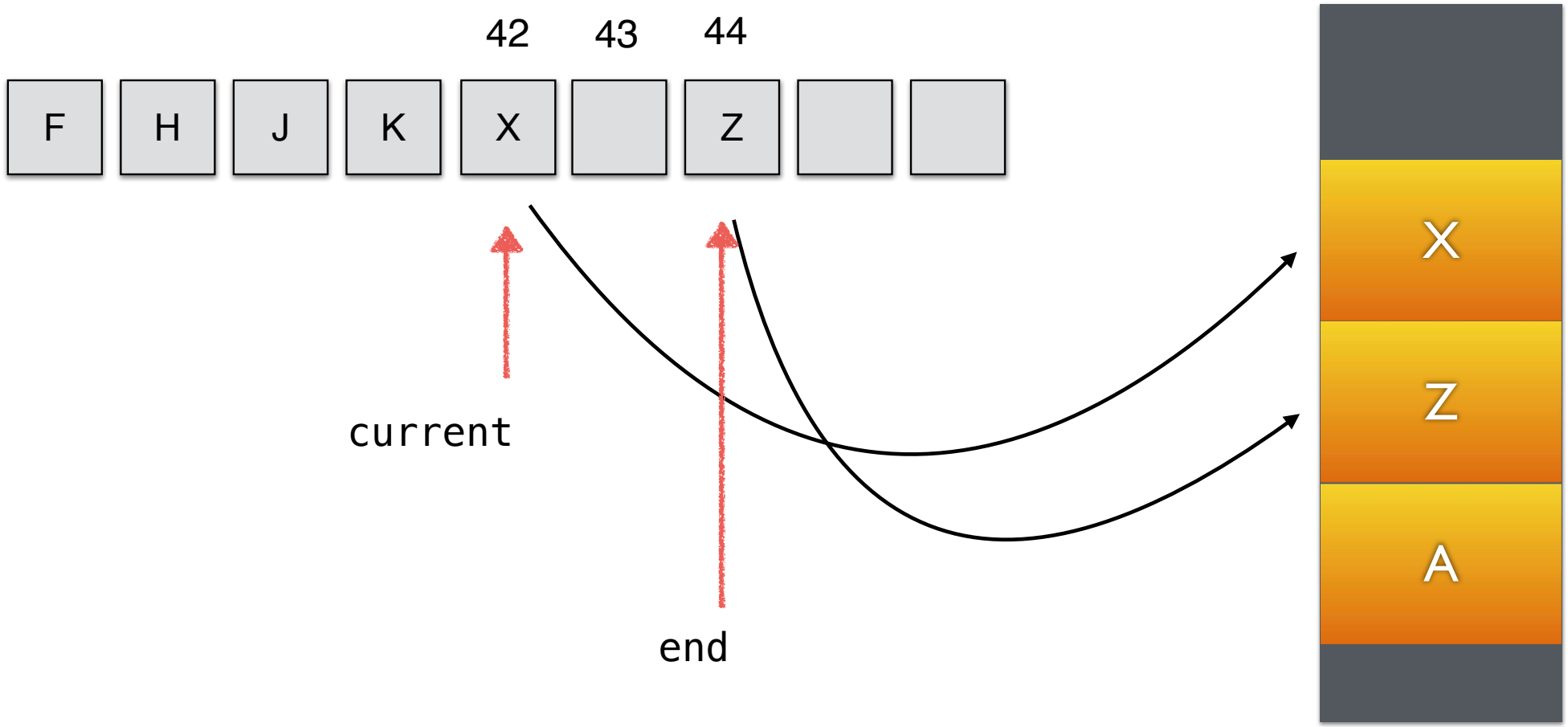
Liste mit Array

- `insert("Z")` 👌



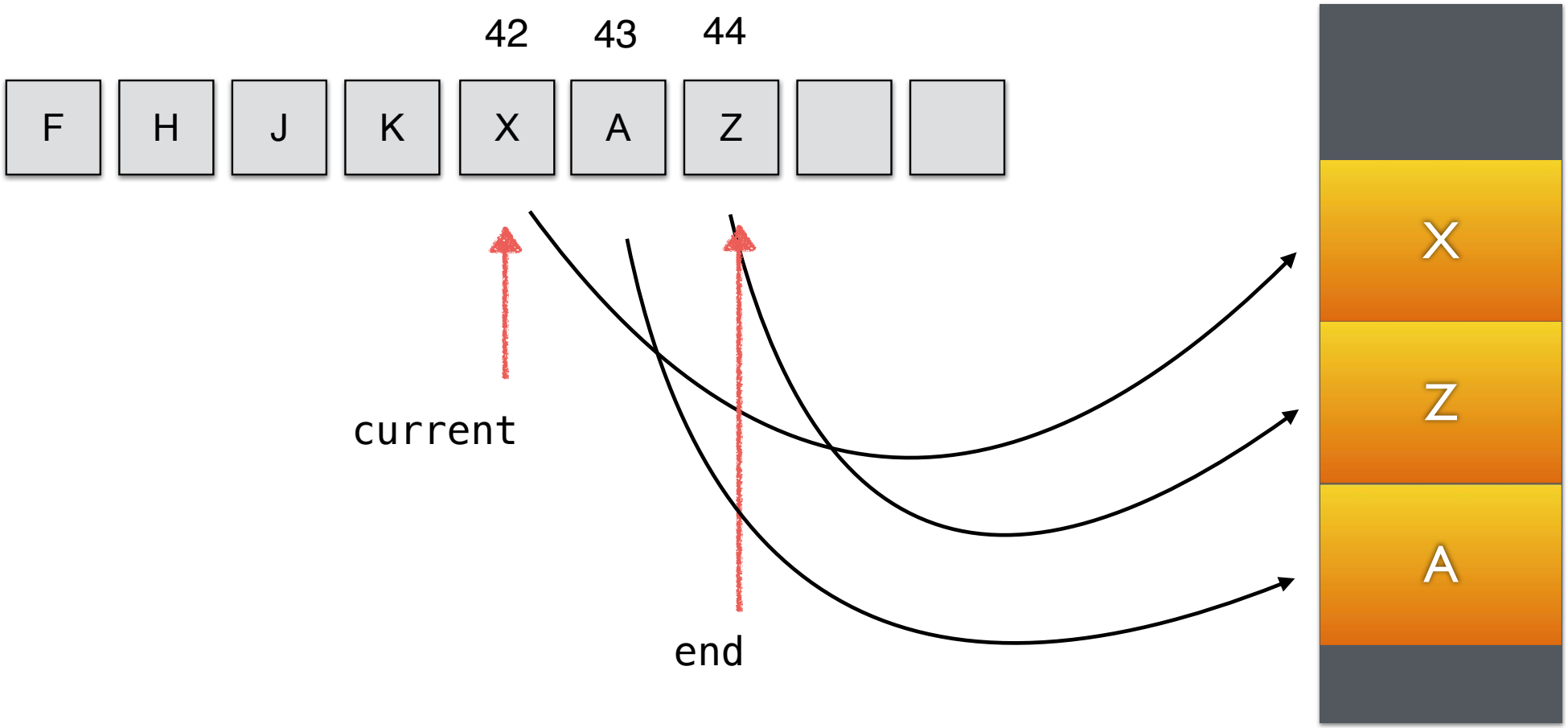
Liste mit Array

- `insert("A")`



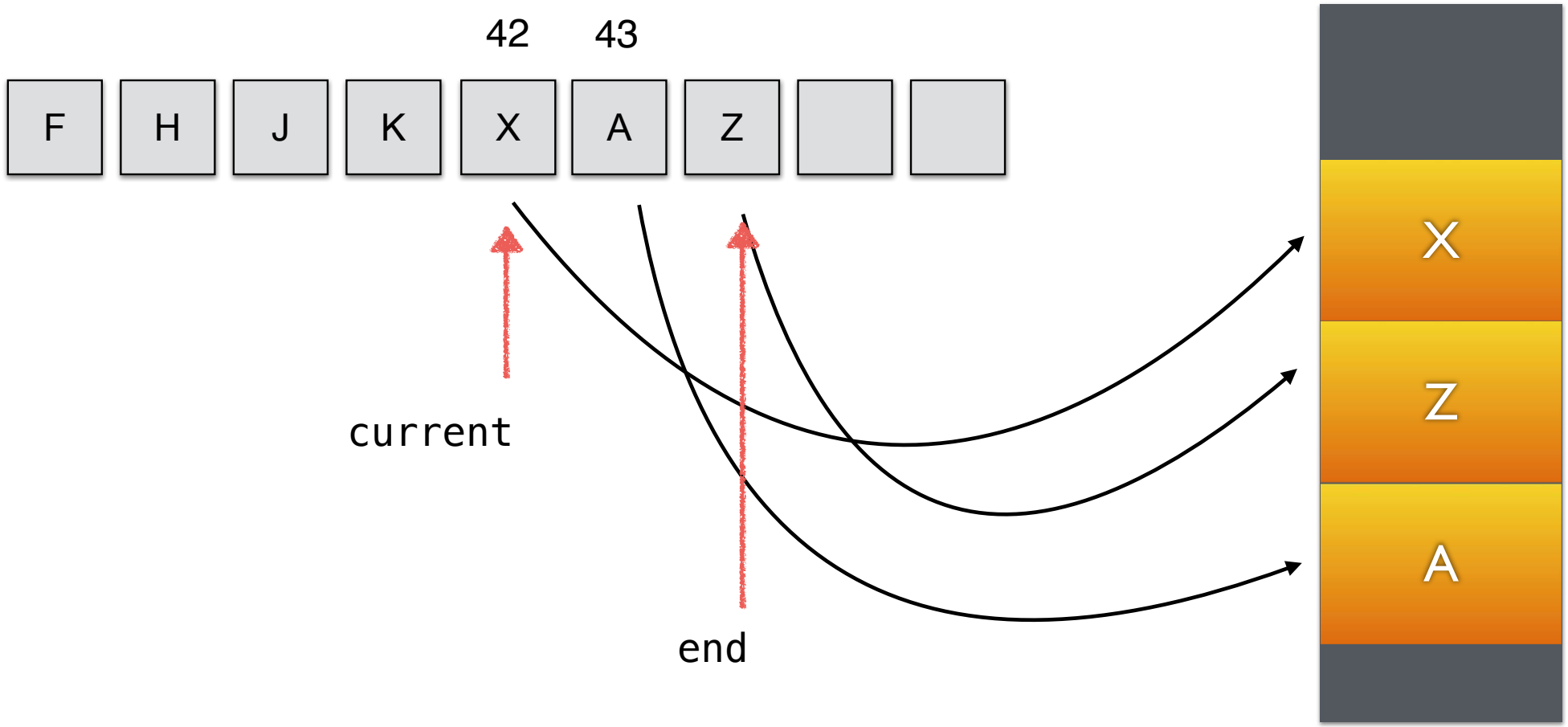
Liste mit Array

- `insert ("A")` 💩



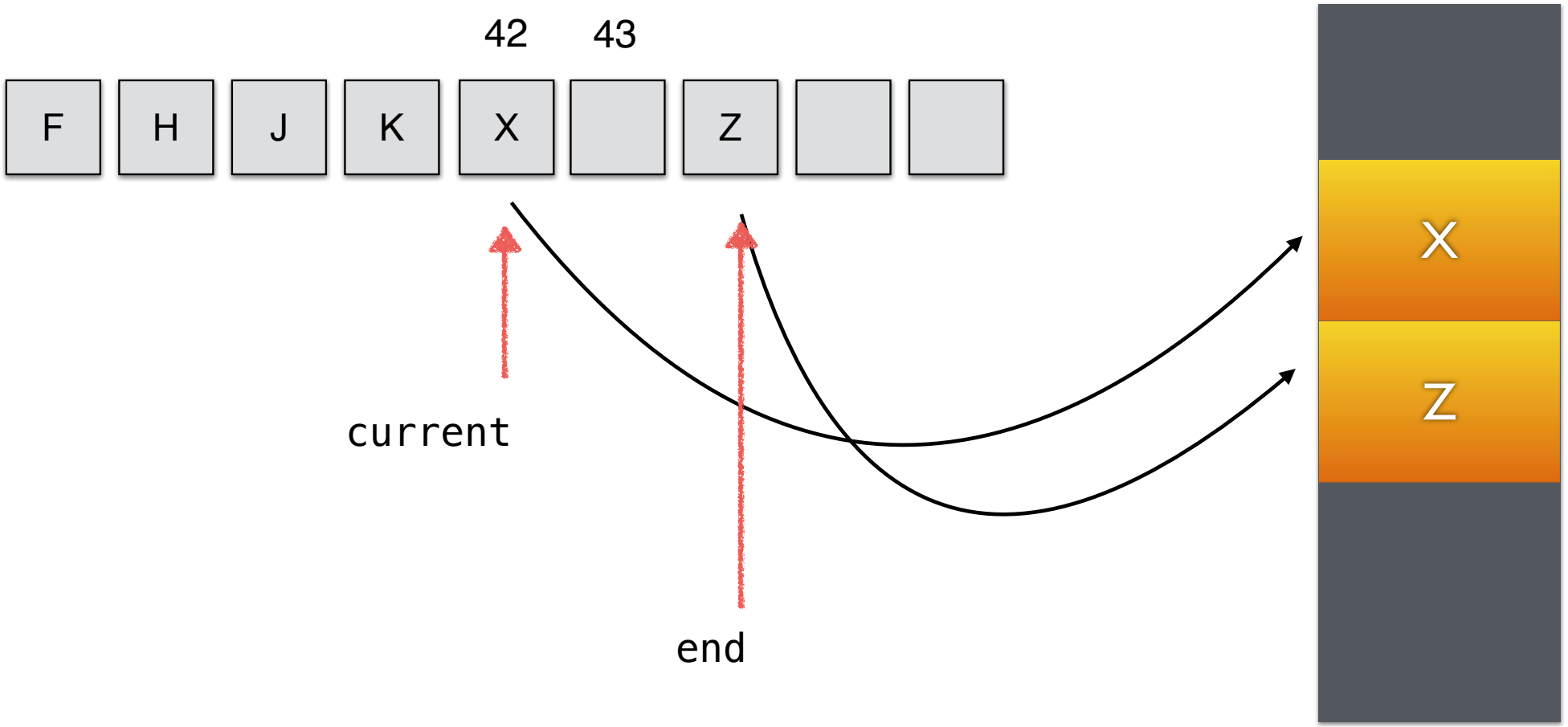
Liste mit Array

- `delete()` ?



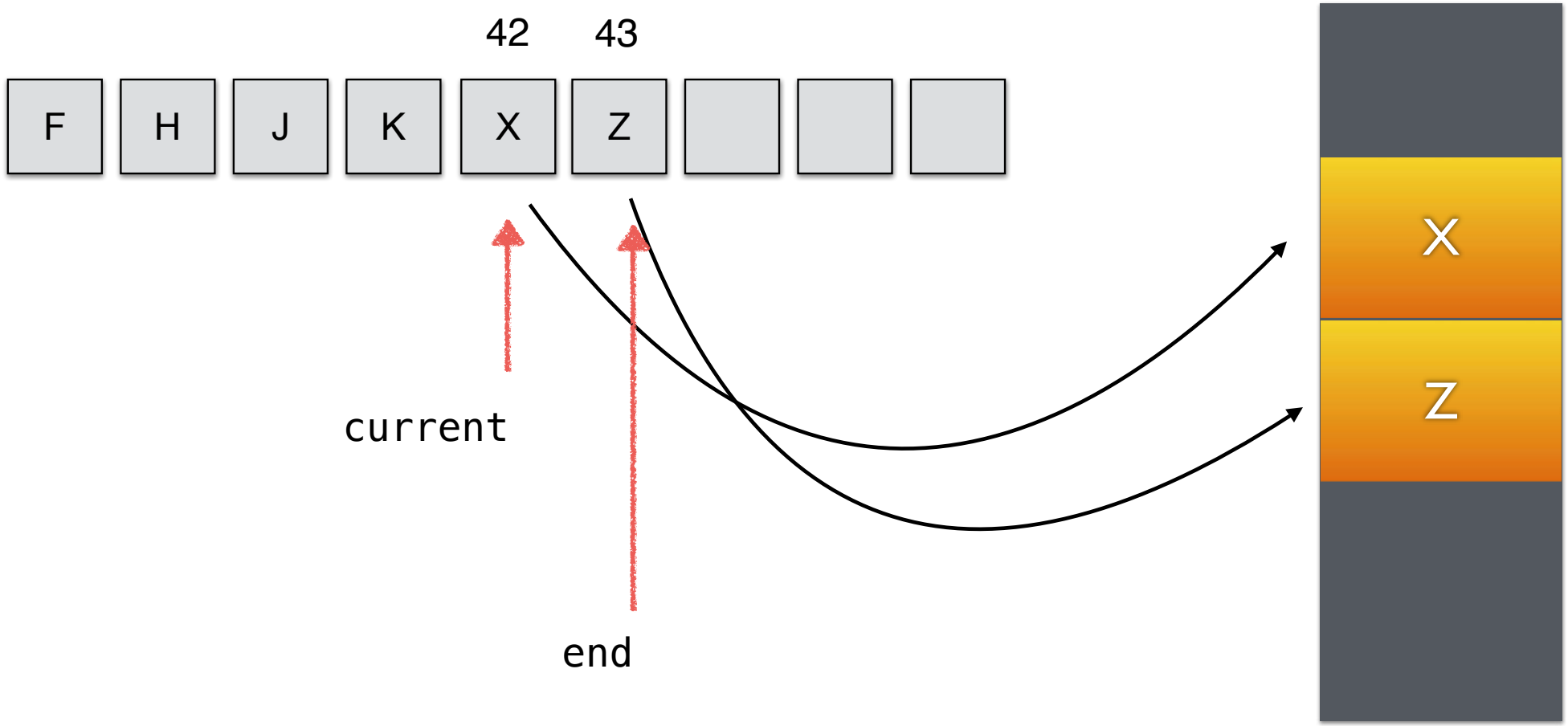
Liste mit Array

- `delete()` ?



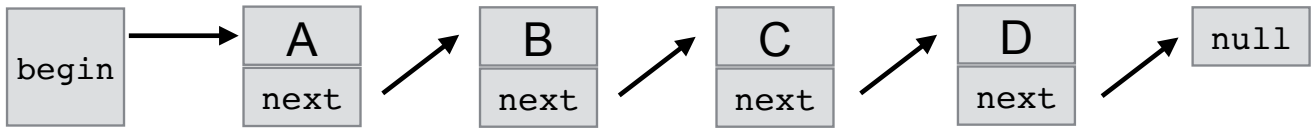
Liste mit Array

- `delete()` 💩



Einfach verkettete Liste (1)

- Prinzip



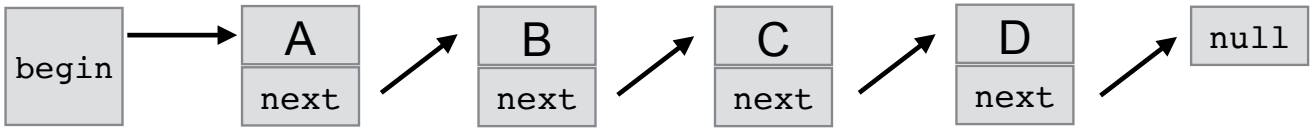

```
public class Node{
    public Object data;
    public Node next;
}
```

Einfach verkettete Liste - Beispiel (1)

- Grundlegende Datenstruktur (neben Arrays) zum Implementierung eines ADTs

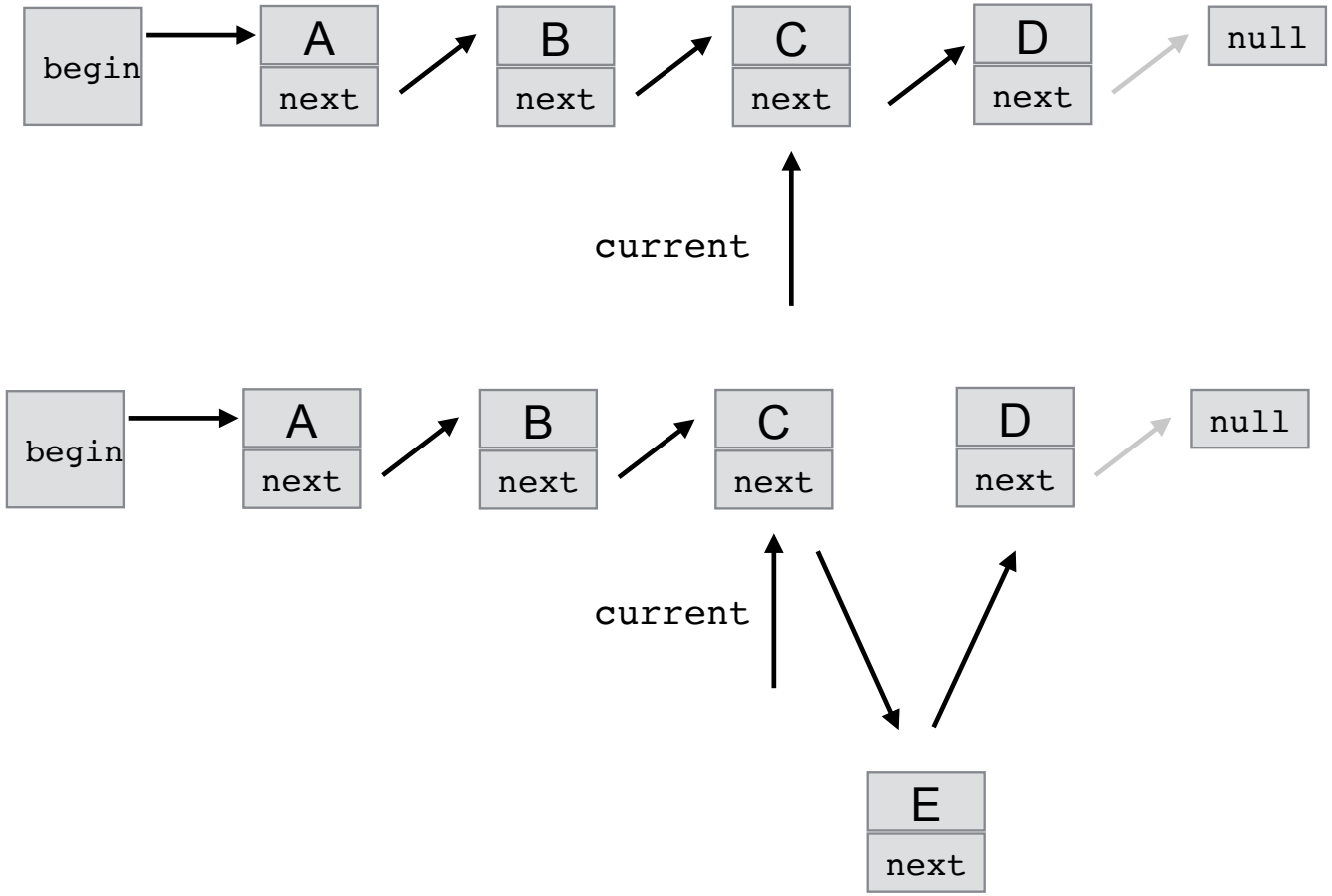
```
public class LinkedList implements List {

    private Node begin;
    private Node current;

    public LinkedList() {
        begin = current = new Node;
        current.next = null;
    }

    public boolean empty() {
        return begin.next == null;
    }

    public boolean end() {
        return current.next == null;
    }
```

```
public void reset() {
    current = begin;
}

public void advance() {
    if(end()) throw new RuntimeException("…");
    current = current.next;
}

public Object current() {
    if(end()) throw new RuntimeException("...");
    return current.next.data;
}
```



Einfach verkettete Liste - Beispiel (2)

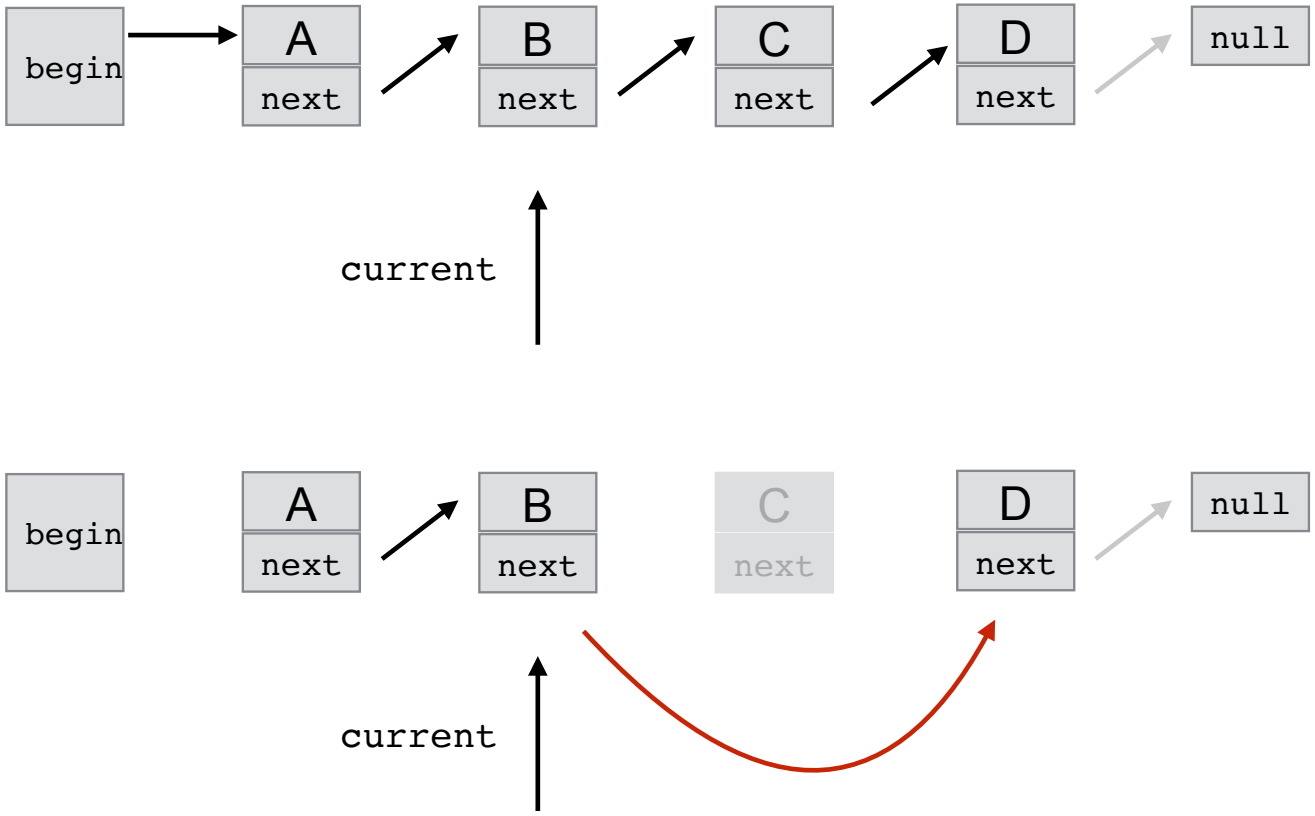
- insert("E")




```
public void insert(Object x) {
    Node tmp = new Node();
    tmp.data = x;
    tmp.next = current.next;
    current.next = tmp;
}
```

Einfach verkettete Liste - Beispiel (3)

- delete()




```
public void delete() {
    if(endpos()) throw new RuntimeException("...");
    current.next = current.next.next;
}
```

Einfach verkettete Liste - Beispiel (4)

- find()

```
public boolean find(Object x) {
    // Go to begin of list
    reset();

    // Advance until Object is found
    // or end of list is reached
    while(!end() && current.next.data != x) {
        current = current.next;
    }

    return !end();
}
```

Beispiel - Liste von Studierenden (1)

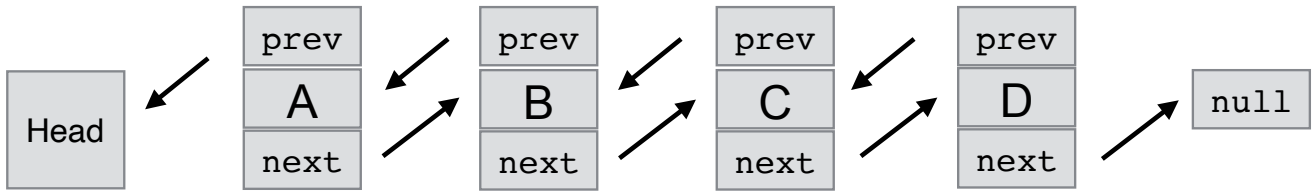
```
public class Person {
     String lastName;
     String firstName;
     Date birthDate;

     public Person(String fn, String ln, int d, int m, int y) {
         fistName = fn; lastName = ln; birthDate = new Date(d, m, y);
     }
 }

 public class Student extends Person {
     int studentNumber;
     int beginOfStudy;
     static int nextStudentNumber = 100000;
     String subject;

     public Student (String fn, String ln, int d, int m, int y, String s, int b) {
         super(fn, ln, d, m, y);
         subject = s;
         studentNumber = nextStudentNumber++;
     }
 }
```

Beispiel - Liste von Studierenden (2)

```
// Create two students
LinkedList l = new LinkedList();
Student s = new Student("Susi", "Sorglos", 1, 10, 2001, "Bio", 2020);
Student w = new Student("Willi", "Wacker", 29, 3, 1970, "Germanistik", 1992);

// Insert them into the list
l.insert(s);
l.insert(w);
l.reset();

// Print all subjects
while(!l.end()) {
    System.out.println (((Student)l.current()).subject);
    l.advance();
}
```

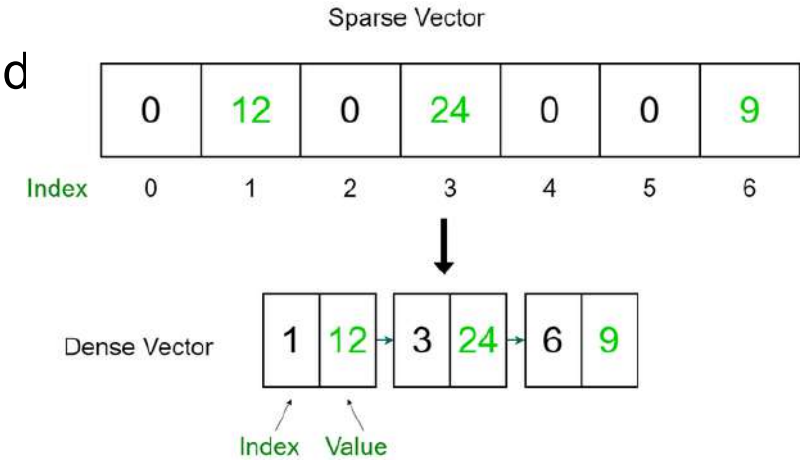
Doppelt verkettete Listen

- Wie einfache Listen, erlauben es aber in beide Richtungen zu traversieren



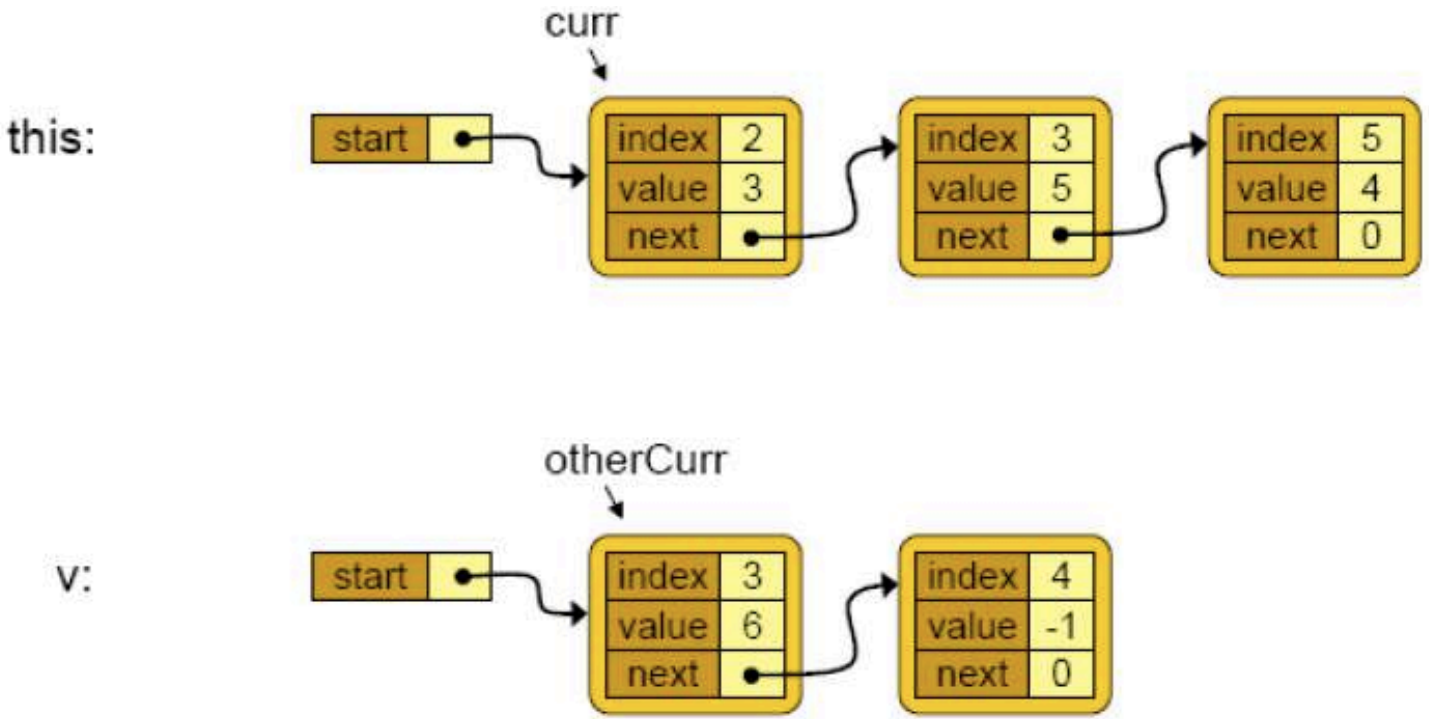
Beispiel: Sparse Vector Arithmetics (1)

- Addition und Subtraktion von hochdimensionalen Vektoren
- Annahme: viele Einträge sind Null
- Anforderungen:
  - Effizient
  - Zeit sollte proportional zu den nicht-Null-Werten sein
- Konzept:
  - Implementation als a += b
  - Traversiere beide verkettete Listen der Vektoren, und berechne während dessen das arithmetische Ergebnis
- Funktioniert nur, wenn die Listen nach Index sortiert sind



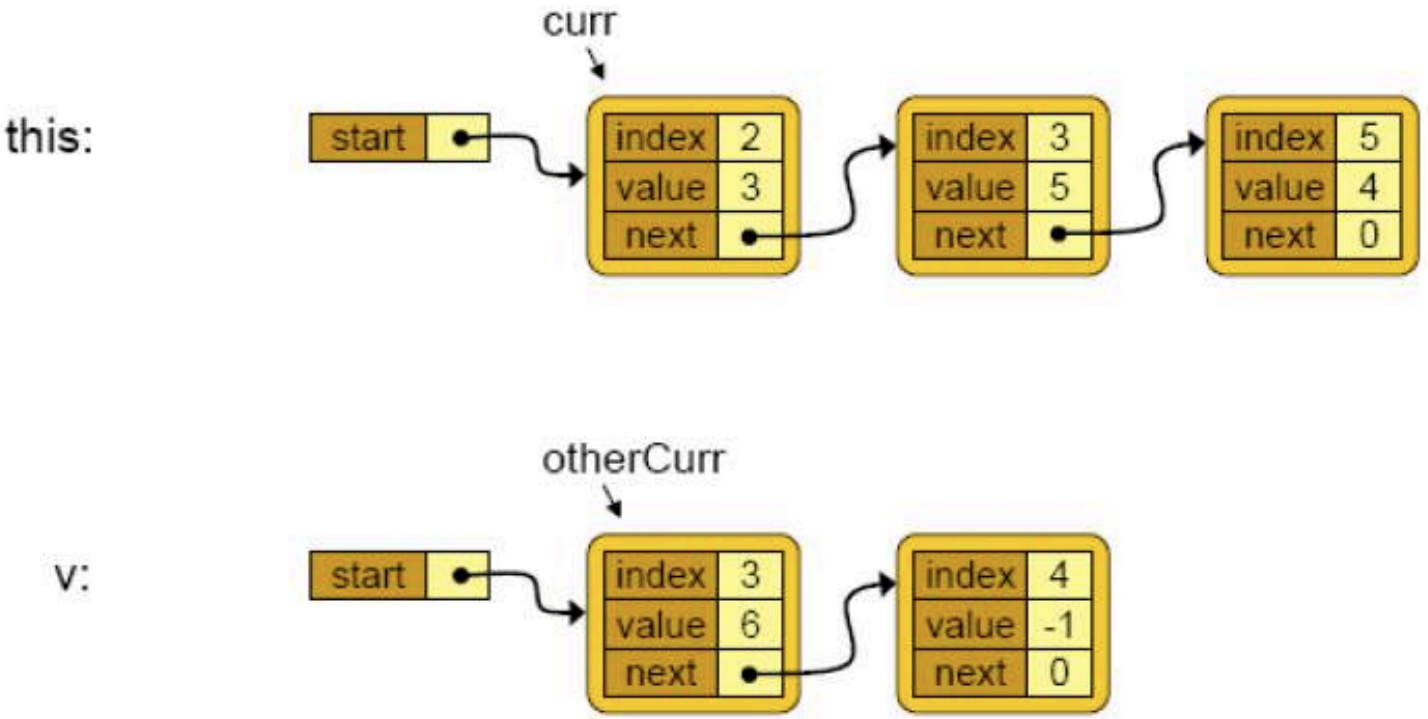
Beispiel: Sparse Vector Arithmetics (2)

- Start der Iteration



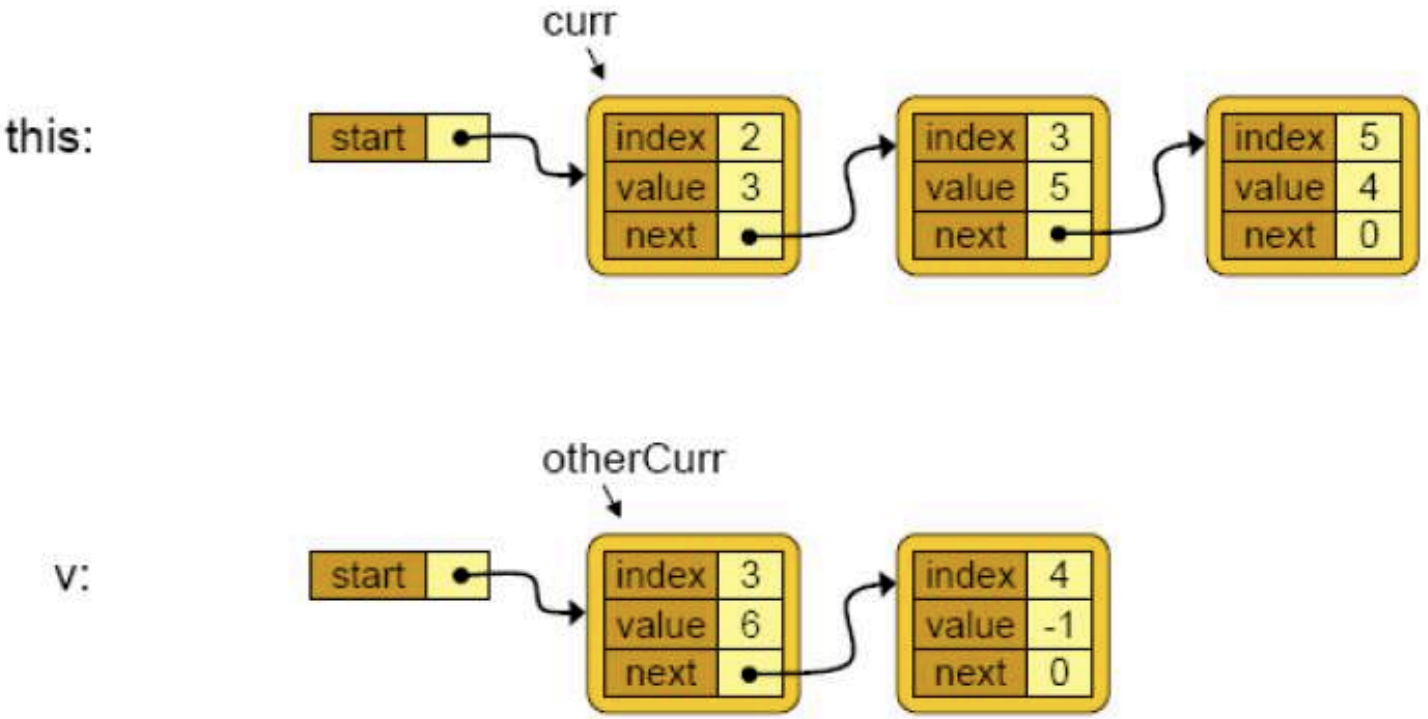
Beispiel: Sparse Vector Arithmetics (3)

- `v` hat keinen Eintrag für `2`, also ändere den Wert in `this` nicht
- setzte `curr` weiter



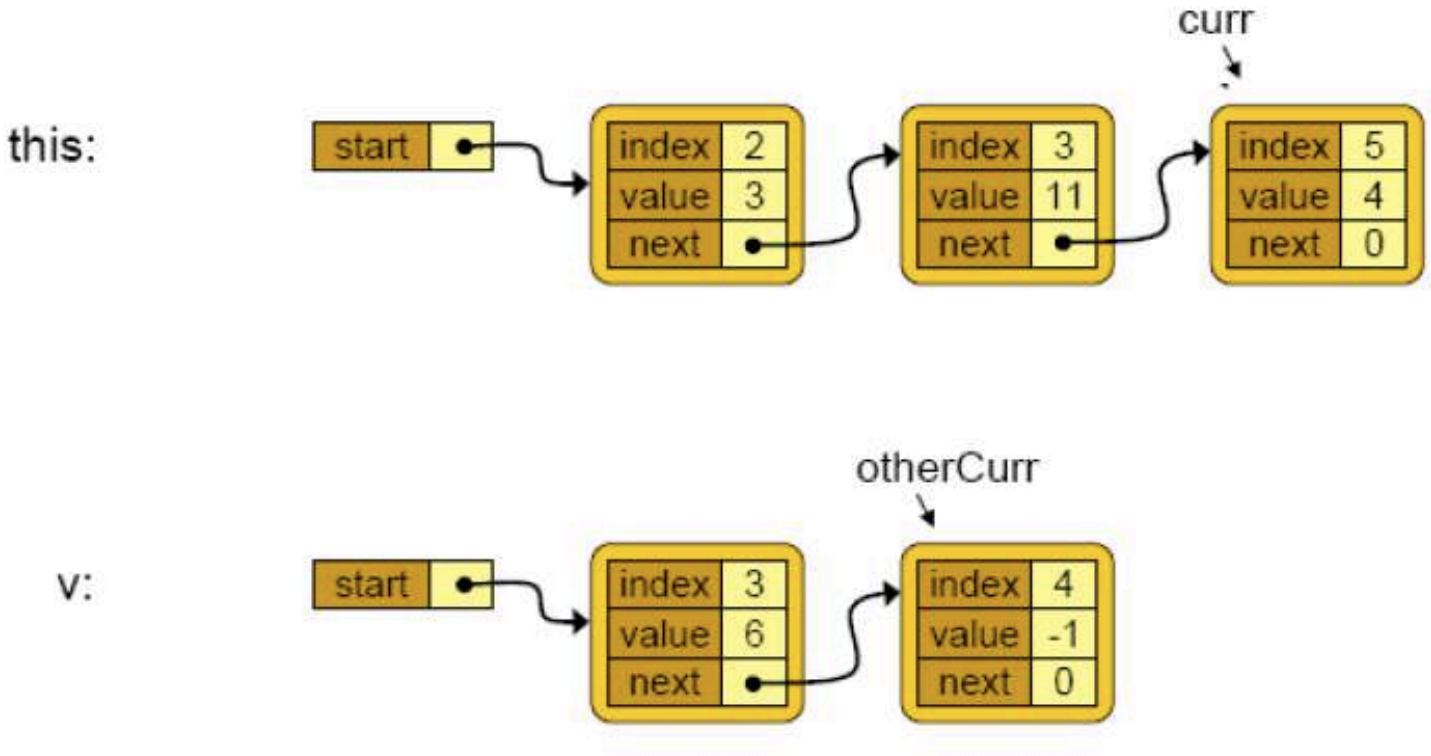
Beispiel: Sparse Vector Arithmetics (4)

- Beide haben einen Eintrag für Index 3
- Addiere die Werte und setze beide Zeiger weiter



Beispiel: Sparse Vector Arithmetics (5)

- Beide haben einen Eintrag für Index 3
- Addiere die Werte und setze beide Zeiger weiter



Beispiel: Sparse Vector Arithmetics (6)

- Nach Einfügen des Knotens setze `otherCurr` weiter
- hier sind wir jetzt fertig