



## TITLE

# Überblick

- 1 Rekursion
- 2 Listen
- 3 Hoare-Kalkül
- 4 Bäume
- 5 SOS-Kalkül

# TITLE

- 1 Rekursion
- 2 Listen
- 3 Hoare-Kalkül
- 4 Bäume
- 5 SOS-Kalkül

# LIVEDEMO

## Mergesort

- ▶ Zu sortierende Liste aufteilen
- ▶ Wenn Teil-Listen nur noch 1 oder 0 Elemente drin sind
- ▶ Listen beim Wiederaufstieg zusammensortieren

### Beispiel

Alphabetisch sortieren: INFORMATIK → AFIKMNORT

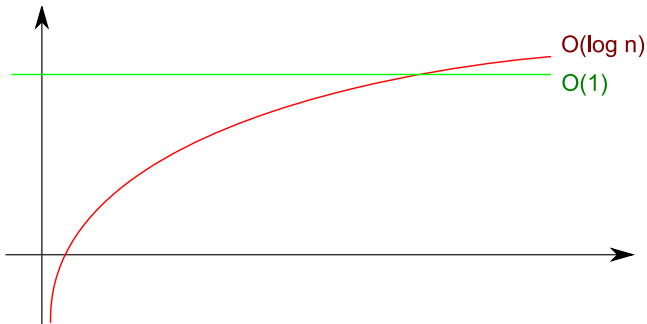
## O-Notation

- ▶ Beschreibt die Laufzeitkomplexität einer Funktion
- ▶ **nicht:** konkrete Laufzeit
- ▶ **sondern:** Wachstum der Funktion
- ▶ Faktoren, die direkt mit Hardware oder Rechenleistung zu tun haben

### Aufpassen bei O-Notation

- ▶ Geht von Unendlicher Eingabemenge aus
- ▶ Nur obere Schranke
  - ▶ Wird unter Umständen nie Erreicht

## Beispiel



- ▶ Konstanten werden Ignoriert
- ▶ Eigentlich  $O(1)$  besser

# TITLE

- 1 Rekursion
- 2 Listen
- 3 Hoare-Kalkül
- 4 Bäume
- 5 SOS-Kalkül

## Warum dieses relativ einfache Thema?

- ▶ Wird sicherlich in der Klausur nicht direkt nachgefragt
- ▶ Aber: Alle Primzahlen zwischen 1 und  $n$  berechnen und zurückliefern.
- ▶ Wohin mit den Ergebnissen?
- ▶ Man braucht eine Liste und die aus der Bibliothek darf man meistens nicht benutzen
- ▶ Also kurz selbst eine entsprechende Klasse definieren
- ▶ Kann man das ohne groß darüber nachzudenken spart man Zeit für die eigentliche Aufgabe

## Elementklasse

```
class Element {
    int Nutzlast;
    Element Naechstes;

    Element(int Nutzlast, Element Naechstes) {
        this.Nutzlast = Nutzlast;
        this.Naechstes = Naechstes;
    }
}
```

## Listenklasse (1)

```
class Liste {
    Element Anfang, Ende;

    Liste(int Zahl) {
        Anfang = Ende = new Element(Zahl, null);
    }

    ...
}
```

## Listenklasse (2)

```
class Liste {  
    ...  
  
    void add(int Zahl) {  
        Ende.Naechstes = new Element(Zahl, null);  
        Ende = Ende.Naechstes;  
    }  
  
    void zeig_dich() {  
        Element aktuell = Anfang;  
        while(aktuell != null) {  
            System.out.println(aktuell.Nutzlast);  
            aktuell = aktuell.Naechstes;  
        }  
    }  
  
    ...  
}
```

## Listenklasse (3)

```
class Liste {  
    ...  
  
    void zeig_dich_rek() {  
        zeig_dich_rek(Anfang);  
    }  
  
    void zeig_dich_rek(Element aktuell) {  
        if (aktuell == null) {  
            return;  
        }  
        System.out.println(aktuell.Nutzlast);  
        zeig_dich_rek(aktuell.Naechstes);  
    }  
  
    ...  
}
```

## Listenklasse (4)

```
class Liste {
    ...

    void sortiere_ein (int Zahl) {
        Anfang = sortiere_ein (Zahl, Anfang);
    }

    Element sortiere_ein (int Zahl, Element aktuell) {
        if (aktuell == null)
            return Ende = new Element (Zahl, null);
        else if (Zahl < aktuell.Nutzlast)
            return new Element (Zahl, aktuell);
        else {
            aktuell.Naechstes = sortiere_ein (Zahl, aktuell.Naechstes);
            return aktuell;
        }
    }
}
```

# TITLE

- 1 Rekursion
- 2 Listen
- 3 Hoare-Kalkül**
- 4 Bäume
- 5 SOS-Kalkül

## Hoare-Kalkül

# LIVEDEMO

- ▶ [http://wwswt.informatik.uni-rostock.de/deutsch/Mitarbeiter/michael/lehre/pt\\_2001/Hoare\\_akt\\_008.pdf](http://wwswt.informatik.uni-rostock.de/deutsch/Mitarbeiter/michael/lehre/pt_2001/Hoare_akt_008.pdf)

# TITLE

- 1 Rekursion
- 2 Listen
- 3 Hoare-Kalkül
- 4 Bäume**
- 5 SOS-Kalkül

## Wissenswertes zu Bäumen

- ▶ Durchlauf-Algorithmen verinnerlichen (Pre-, In-, Post-Order, Breitendurchlauf)
- ▶ linksseitig balancierte, vollständige Binärbäume können als Array gespeichert werden
- ▶ Binärbaum → binäre Suche

## Infos & Applets zu Bäumen

- ▶ <http://de.wikipedia.org/wiki/AVL-Baum>
- ▶ <http://fbim.fh-regensburg.de/~saj39122/bruhi/index.html>
- ▶ <http://de.wikipedia.org/wiki/B-Baum>
- ▶ <http://slady.net/java/bt/>

# TITLE

- 1 Rekursion
- 2 Listen
- 3 Hoare-Kalkül
- 4 Bäume
- 5 SOS-Kalkül

## SOS allgemein

- ▶ Hintereinanderausführung von Regeln
- ▶ Konstruktion von Ableitungen
  - ▶ Ziel als Ausgangspunkt
  - ▶ Rückwärtsanwendung von Regeln
- ▶ Vorgehensweise:
  - ▶ von unten nach oben
  - ▶ von links nach rechts
  - ▶ neuer Zustand  $\sigma$  bei Zuweisung (update!)

# SOS Beispiel

LIVEDEMO

$$\frac{\langle (y > 1), \sigma \rangle \rightarrow \mathbf{W}}{\langle \text{while}(y > 1) \text{ do } x = x - y; y = y - x, \sigma \rangle \rightarrow \sigma'}$$