Einführung in die Informatik

– Cheat Sheet Zine –

Datentypen / Arrays
Rekursion / OOP

Listen / Stacks / Queues
(Such) Bäume

Sortieralgorithmen

Datentypen

kleine Zahlen

byte → short → int → long → float → double

char: Symbole; ganze Zahlen; gleitkommazahlen

boolean { true / false

< > kleiner/größer — && UND
<= >= kl. gleich/gr. gleich — || ODER
== gleich — ^ XOR
!= ungleich — ! Nicht-

```
DatenTyp[] name = new DatenTyp[Länge];
int[] myArr = new int[10];
```

- Kann nur 1 Art Datentyp speichern
- Kann Mehrdimensional sein: `myArr[][]...`
- direkter Zugriff: `myArr[5]`
- beginnt bei 0 zu zählen

Arrays

OOP

- Methoden:
  `Modifikator Datentyp Name(Datentyp Parameter,...){}`
  `public static void main(String[] args){}`
  - Rückgabe: `return a;`
  - Ausgabe: `System.out.println(a);`
  - Aufruf mit `.` - Operator:
    `Auto.setGeschwindigkeit(180);`
  - Überladen: gleicher Name, andere Parameter
  - Überschreiben: gleiche Signatur in Unterklasse
- Klassen
  - Instanziieren: `Auto a1 = new Auto("red");`
  - Konstruktor: wie Klassenname; Parameter wählbar

Rekursion

- Fallunterscheidung zwischen Basisfall und Rekursionsschritt
- Termen: Linear, Kaskadenförmig, verschachtelt
- Head- vs Tail-Rekursion: Rekursiver Aufruf am Anfang bzw Ende der Methode

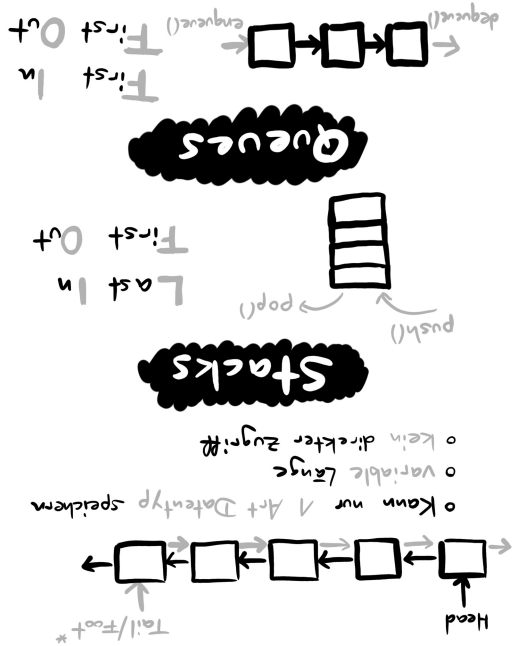
Tips & Tricks

CC-BY Bleeptrack

- O-Notation:
  $1 < \log n < \sqrt{n} < n < n\log n < n^2 < 2^n < n! < n^n$
- Lass beim Programmieren auf dem Blatt immer eine Zeile frei, falls du noch etwas einfügen willst
- Halte dich an die Algorithmen im Skript! Sie können sich von diesen hier unterscheiden!
- Üben! Üben! Üben!
- Schreibe dir einen Spickzettel (und lasse ihn bei der Klausur in der Tasche)
- Kreuze bei Multiple Choice im Zweifelsfall irgend etwas an.
- Vergiss nicht deinen Studierendenausweis zur Klausur mitzubringen
- Melde dich rechtzeitig zur Klausur an!

Du schaffst das!

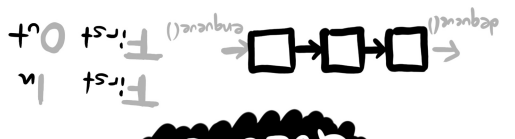
Listen



- Kann nur 1 Art Datentyp speichern
- Variable Länge
- Kein direkter Zugriff

Stacks

Queues

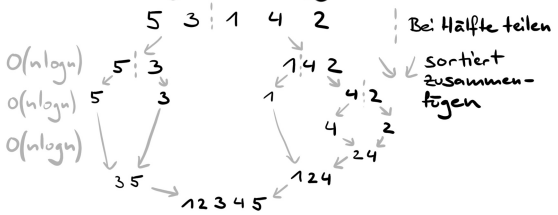


Quicksort

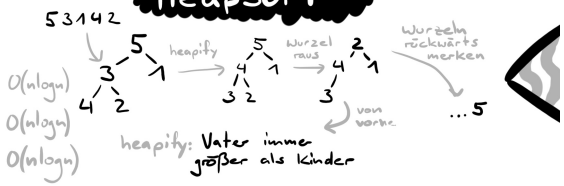
Best-Avg-Worst-Case

5 3 1 4 2 — O(nlogn) — Pivot Ein Element ist Pivot Elem. Andere Elemente nach größer o. kleiner als P sortieren. Rekursiv weiter.
1 2 5 3 4 — O(nlogn)
1 2 3 4 5 — O(n²)

Mergesort



Heapsort



heapify: Vater immer größer als Kinder

Insertionsort

sortiert

5 3 1 4 2 — O(n) — Erstes Element aus unsortiertem Bereich wird an passende Stelle im sortierten Bereich eingefügt.
3 5 1 4 2 — O(n²)
1 3 5 4 2 — O(n²)
1 3 4 5 2
1 3 4 5 2

Selectionsort

5 3 1 4 2 — Kleinstes Element aus uns. Bereich wird nach vorne im uns. Bereich getauscht und ist so sortiert.
1 3 5 4 2 — O(n²)
1 2 5 4 3 — O(n²)
1 2 3 4 5 — O(n²)
1 2 3 4 5

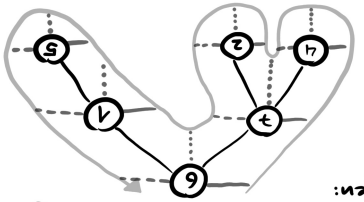
Bubblesort

5 3 1 4 2 — 2 Elemente werden verglichen und bei falscher Reihenfolge getauscht. Vergleich läuft weiter nach rechts bis zum sortierten Bereich.
3 1 4 2 5 — O(n)
1 3 2 4 5 — O(n²)
1 2 3 4 5 — O(n²)
1 2 3 4 5

Bäume

L: Links
R: rechts
A: Ausgabe

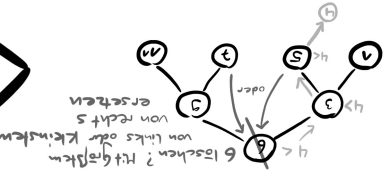
Traversierungen:
Pre: A L R
In: L A R
Post: L R A



Beispiel:

Pre:	6	3	4	2	1	5
In:	4	3	2	6	1	5
Post:	4	2	3	5	2	6

Tiefe Rekonstr. in Pfleinfachheit
Laufen und mit Inorder vergleichen

Suchbäume

kann auch anders sein!