Algorithmen und Datenstrukturen

B1. Einführung in Datenstrukturen

Marcel Lüthi and Gabriele Röger

Universität Basel

7. März 2019

Algorithmen und Datenstrukturen
7. März 2019 — B1. Einführung in Datenstrukturen

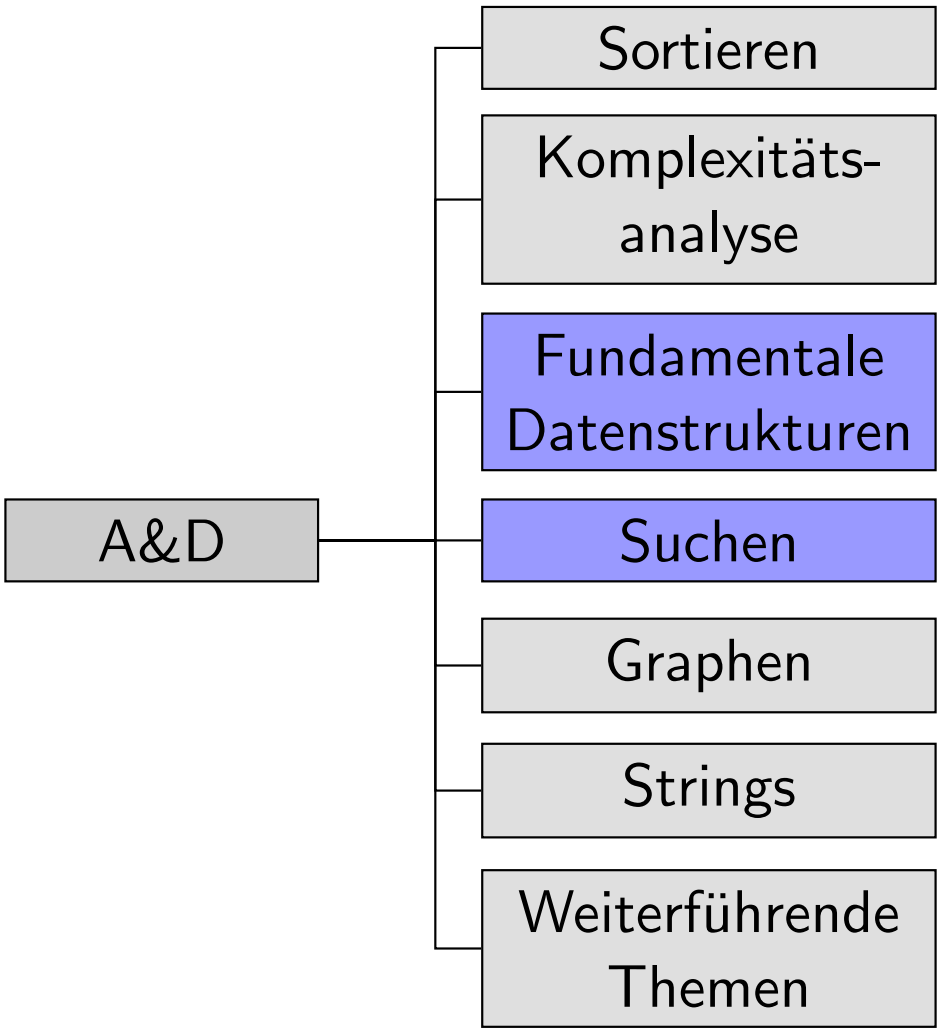
B1.1 Übersicht

B1.2 Datenstrukturen

B1.3 Sortieren mit Heaps

B1.1 Übersicht

Übersicht


A&D
Sortieren
Komplexitäts-
analyse
Fundamentale
Datenstrukturen
Suchen
Graphen
Strings
Weiterführende
Themen

Ausblick : Fundamentale Datenstrukturen

- Datenabstraktion (Abstrakte Datentypen)
  - Multimengen, Stapel, (Prioritäts-) Warteschlangen
- Datenstrukturen
  - Arrays, Verkettete Listen, Bäume, Heaps

Höhepunkt: Heapsort

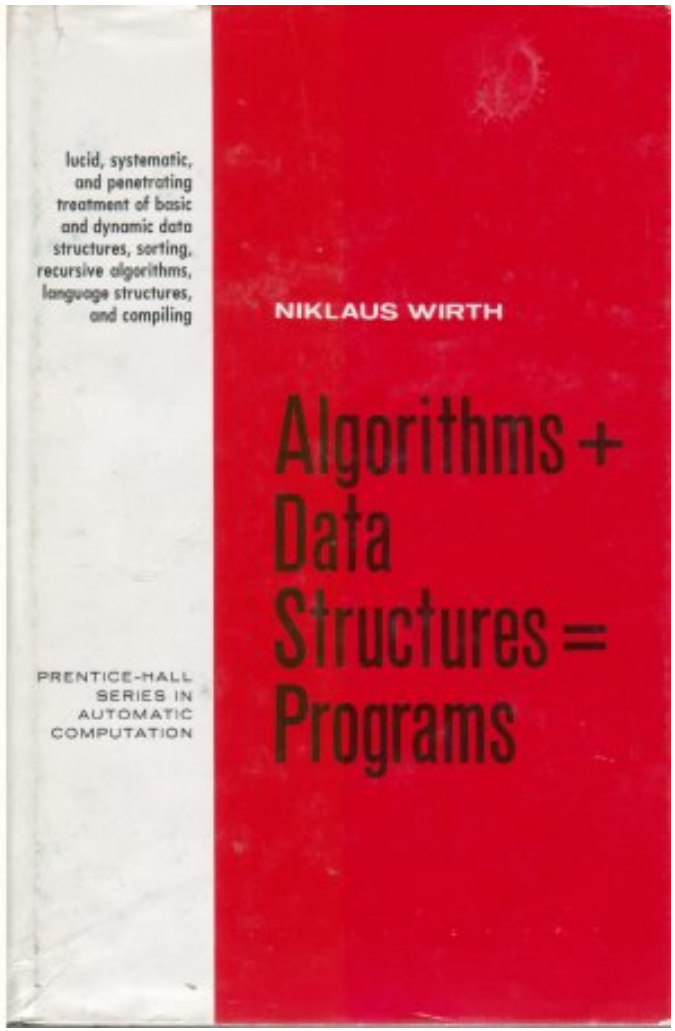
Elegantes Zusammenspiel Algorithmus und Datenstruktur.

- Clevere Datenstruktur - Simpler Algorithmus
- Garantiertes Laufzeitverhalten $O(n \log n)$

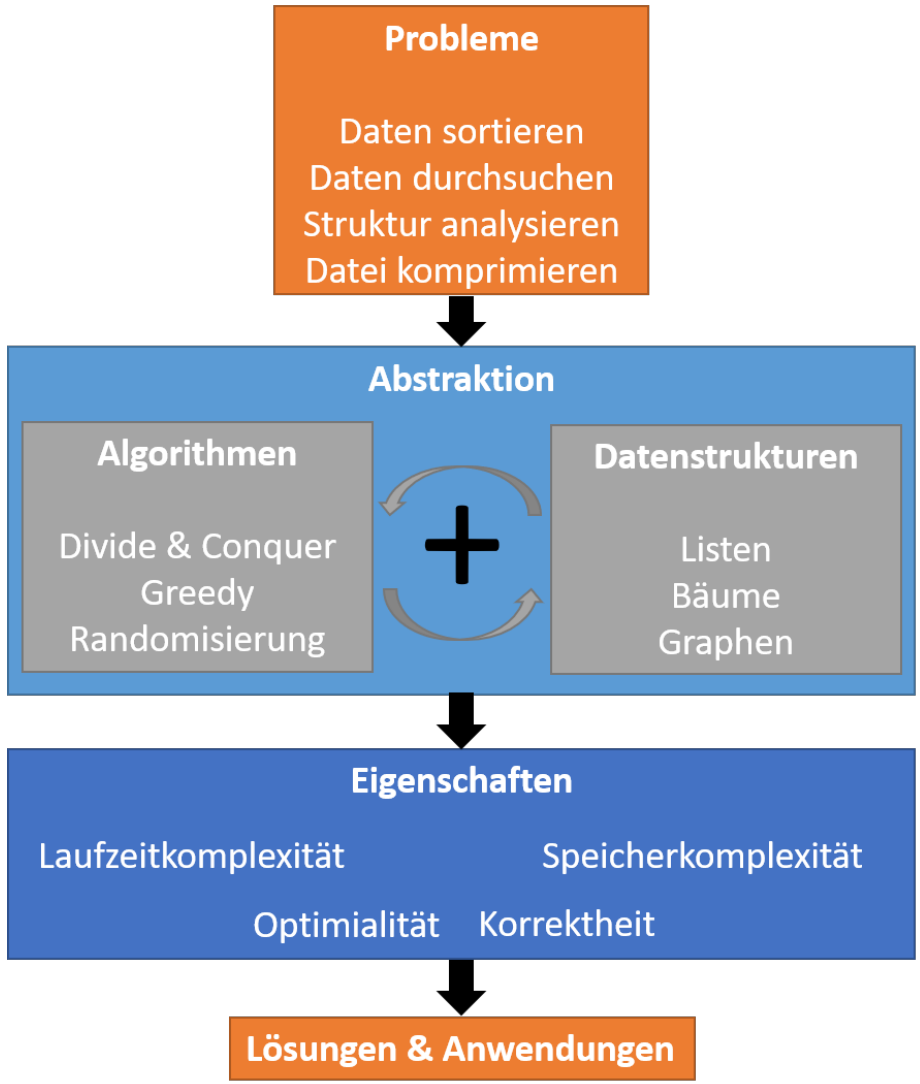
B1.2 Datenstrukturen

Datenstrukturen

- Programmieren ist mehr als Algorithmen schreiben
  - Datenorganisation ist zentral
- Elegante Datenstrukturen führen zu elegantem Code
- Programmierer
  - braucht Katalog von Datenstrukturen
  - muss Eigenschaften kennen



Übersicht


Probleme
Daten sortieren
Daten durchsuchen
Struktur analysieren
Datei komprimieren
Abstraktion
Algorithmen
Divide & Conquer
Greedy
Randomisierung
Datenstrukturen
Listen
Bäume
Graphen
Eigenschaften
Laufzeitkomplexität
Speicherkomplexität
Optimialität
Korrektheit
Lösungen & Anwendungen

Datenstrukturen

> Bad programmers worry about the code. Good programmers worry about data structures and their relationships.
>
> Linus Torvalds

Datenstrukturen

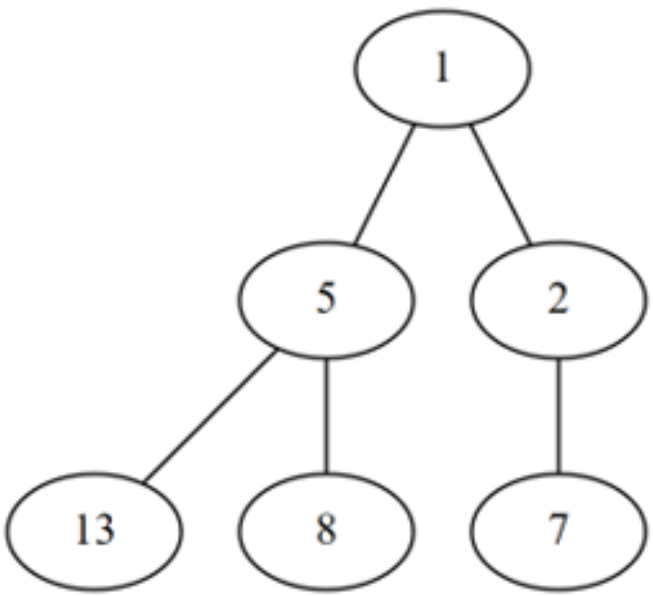
> Show me your flowcharts and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowcharts; they'll be obvious.
>
> Fred Brooks

B1.3 Sortieren mit Heaps

Beispiel: Sortieren mit Heaps (Ausblick)

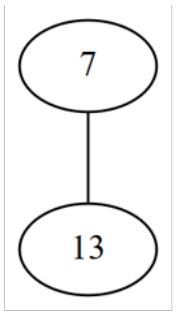
Datenstruktur: Heap

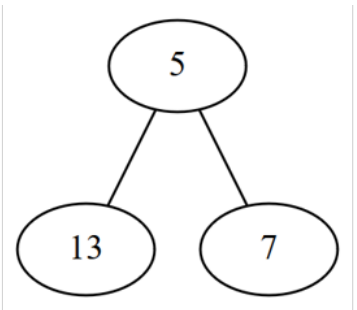
Ein (binärer) min-Heap ist ein vollständiger binärer Baum, bei dem gilt, dass der Wert in jedem Knoten kleiner gleich dem Wert seiner beiden Kindern (sofern vorhanden) ist.

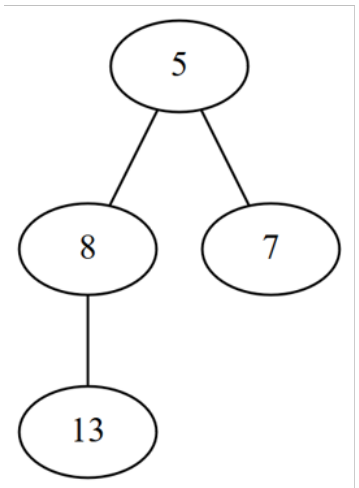


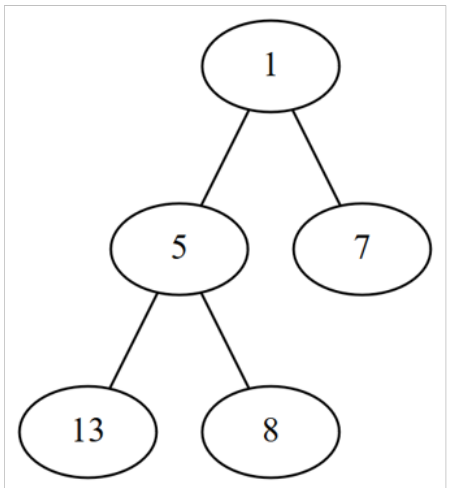
Aufbauen eines Heaps (Wandtafel)

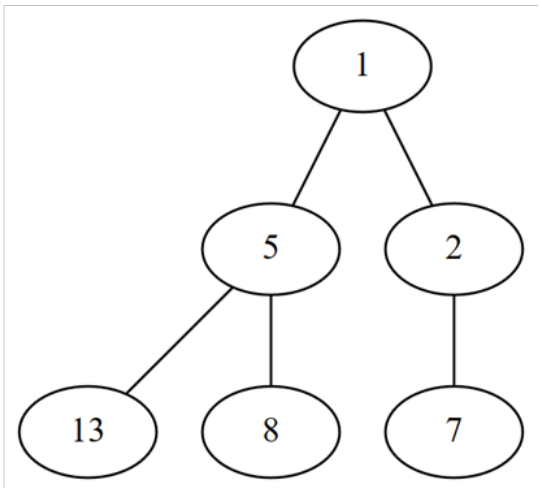
Elemente: 7, 13, 5, 8, 1, 2



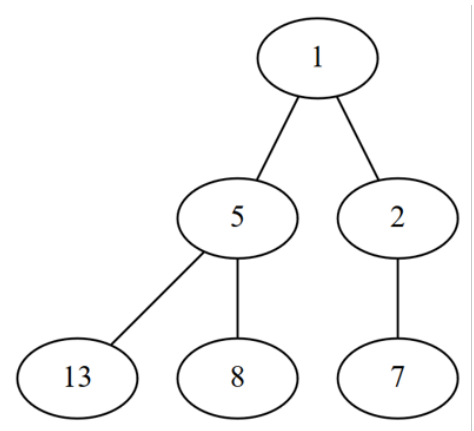


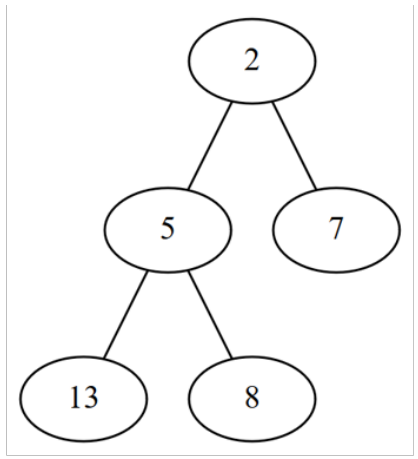


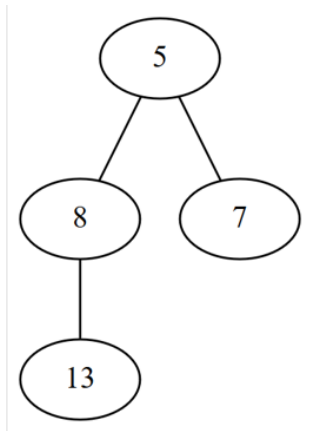


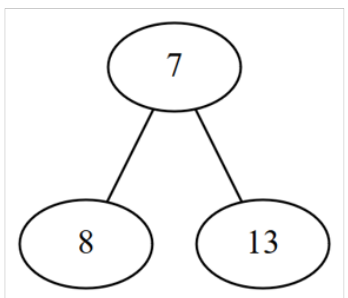


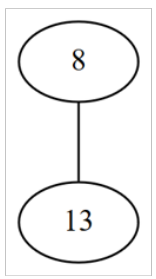
Entfernen des kleinsten Elements (Wandtafel)

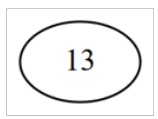






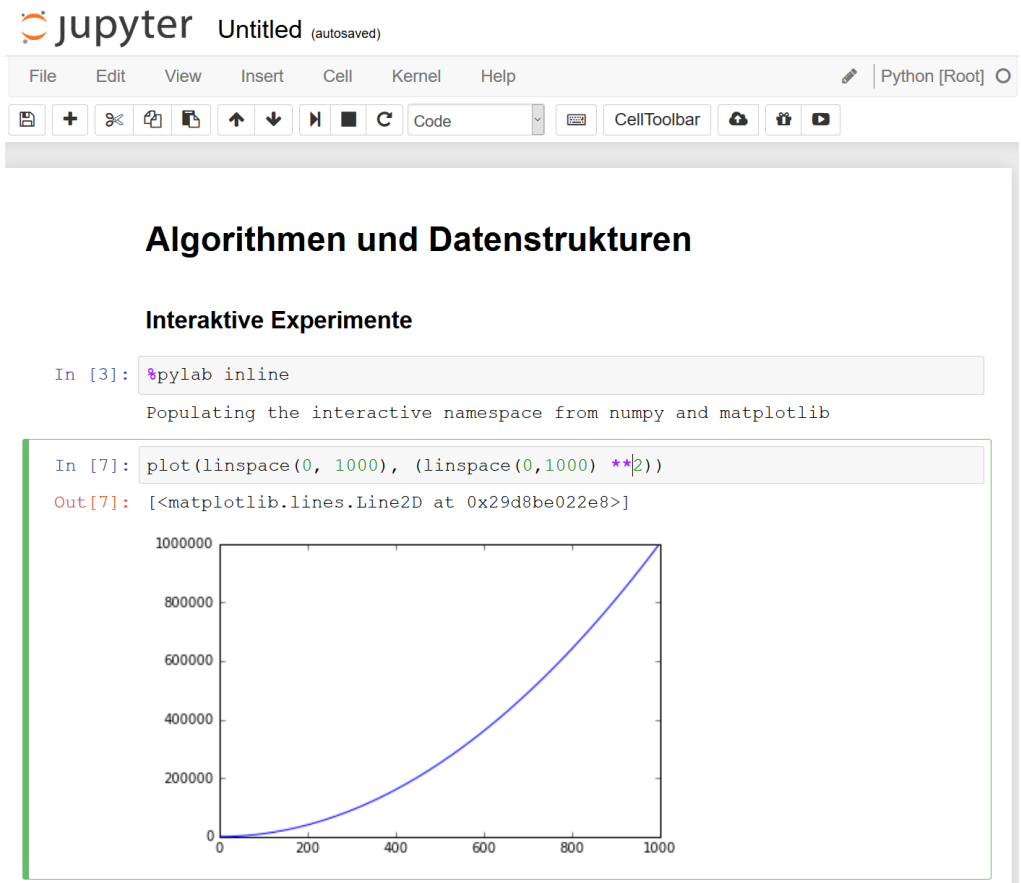






Jupyter notebook

Schreiben Sie einen Sortieralgorithmus



Jupyter Notebook: heapq.ipynb

Beispiel: Sortieren mit Heaps (Ausblick)

Idee des Algorithmus:

- Baue Heap aus unsortierter Liste
- Solange Elemente im Heap sind
  - Entferne kleinstes Element (Wurzel)
  - Schreibe Element in (neue) Liste
  - Stelle Heapbedingung wieder her
- Neue Liste enthält Elemente in sortierter Reihenfolge

Heapsort: Gleiche Idee, aber inplace.

Beispiel: Sortieren mit Heaps (Ausblick)

Offene Fragen:

- Wie schnell können wir Heap aus n unsortierten Elementen aufbauen?
- Antwort: Naiv: In $O(n \log_2 n)$ Operationen. Trickreich: In $O(n)$
- Wie schnell können wir Heapbedingung nach Entfernen wiederherstellen?
- Antwort: In $O(\log_2 n)$ Operationen
- Wie gross ist die gesamte Laufzeitkomplexität
- Antwort: In $O(n \log_2 n)$ Operationen

Komplexität verschoben von Algorithmus nach Datenstruktur

Zusammenfassung

- Algorithmen und Datenstrukturen arbeiten zusammen
  - (Teil der) Komplexität kann verschoben werden
- Datenstrukturen können meist visualisiert/graphisch verstanden werden
- Oft gilt: Gute Datenstrukturen $\Rightarrow$ Einfach(ere) Programme

Details von Heapsort folgen ...

Schöne Fasnachtsferien

Quelle: Basler-Fasnacht.info