Preliminaries

• 5 ECTS
• lectures in German, slides etc. in English
• prequisites:
  • Algorithmen II
  • interest in discrete, combinatorial problems
• 15 lectures (NOT on December 8th)
• oral exam (20 mins)
Preliminaries

- course homepage:
  http://algo2.itl.kit.edu/1909.php
  - slides (& script)
  - hints for LaTeX and mathematical writing
- Johannes.Fischer@kit.edu (room 206)
- office hours: Monday 14-15 (NOT Dec. 5th)
No „Übungen“! Scribing!

• write script for one lecture

• material:
  ▶ slides
  ▶ my notes
  ▶ research literature (NOT wikipedia etc.)

• use LaTeX ⇒ learn to write scientifically

• vector graphics: ipe, xfig, ...
Course Contents

- hashing
- predecessor data structures
- integer sorting/searching
- distance oracles
- tree labelings
- lowest common/level ancestors
- range (minimum) queries
- succinct trees
- text indexing (string B-trees)
Hashing

- set $S$ of $n$ objects from a LARGE universe $U$
- query for membership (+satellite info)
- Use space $O(n)$, not $O(|U|)$
Hashing: lookup time

- chaining/linear probing: \(O(1)\) expected time
- cuckoo hashing: \(O(1)\) worst case time
- other operations \(O(1)\) amortized & expected
Predecessor Queries

• \( S \): \( n \) objects from a SORTED universe \( U \)
• given \( x \in U \), return \( \max\{y \leq x : y \in S\} \)
• fast if elements are integers: \( O(l\log \log |U|) \)

predecessor(\( x \))
Integer Sorting

- sort $n$ elements from a universe $[0, 2^w - 1]$
- comparison based sorting: $\Theta(n \lg n)$
- counting sort: $O(n + 2^w)$
- with predecessor queries: $O(n \lg w)$
- **signature sort:**
  - $O(n)$ for $w$ sufficiently large
  - $O(n \lg \lg n)$ for all $w$
Distance Oracles

distance to C?
Tree Labelings: Ancestors

- L
  - H: (2,8)
  - K: (10,22)
  - I: (11,15)
  - J: (17,21)
  - A: (3,3)
  - B: (5,5)
  - C: (7,7)
  - D: (12,12)
  - E: (14,14)
  - F: (18,18)
  - G: (20,20)
Lowest Common Ancestors
Level Ancestors

2\textsuperscript{nd} ancestor?
Range Minimum Queries

\[ i \quad j \]

\[ 5 \quad 1 \quad 6 \quad 3 \quad 8 \quad 4 \quad 2 \quad 7 \quad 1 \quad 4 \quad 2 \quad 3 \quad 7 \quad 4 \quad 2 \quad 1 \quad 6 \]
2d Range Reporting
Succinct Trees

\[ n \lg n \text{ bits} \]

\[ 2n \text{ bits} \]
String B-Trees

• text indexing in **external** memory
• substring queries (cf suffix tree/array)
• new challenges (minimize IOs)
Theory vs. Practice

• focus on **theoretical** (=mathematical) analysis of data structures

• BUT: most methods highly **practical** (perhaps with some engineering effort)

  ▶ VL "Algorithm Engineering"

• every method better than naive approach (complex analysis $\not\Rightarrow$ slow running time)
## Classification of DSs

<table>
<thead>
<tr>
<th>object</th>
<th>type of DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>numbers</td>
<td>„normal“</td>
</tr>
<tr>
<td>point sets</td>
<td>integer</td>
</tr>
<tr>
<td>graphs</td>
<td>randomized</td>
</tr>
<tr>
<td>trees</td>
<td>distributed</td>
</tr>
<tr>
<td>arrays</td>
<td>succinct</td>
</tr>
<tr>
<td>strings</td>
<td>external</td>
</tr>
<tr>
<td>...</td>
<td>parallel</td>
</tr>
<tr>
<td></td>
<td>cache aware etc.</td>
</tr>
</tbody>
</table>
What is a DS?

- extend functionality
  - ADT + DS = ADT' with ADT' ⊇ ADT
- **tradeoff** time/space
- e.g. tree + LCA

Diagram:
- Query time: $O(n)$, $O(\lg n)$, $O(1)$
- Space: $n^2$
- 2n bits
- $O(n)$ words

Optimum:
- Naive
- Traditional
- Succinct
- Lazy
Implicit DS

• clever storage
  ▶ functionality "for free"
• e.g. heap:

$$\text{parent}(x) = \left\lfloor \frac{x}{2} \right\rfloor$$