Preliminaries

- 5 ETCS
- Lectures in German, slides in English
- Prerequisites
  - Lecture Algorithmen (II)
  - Interest in discrete, combinatorial problems
- Oral exam (20-25 minutes)
- Course homepage: [http://algo2.iti.kit.edu/2472.php](http://algo2.iti.kit.edu/2472.php)
  - Will contain covered material
  - Hints for LaTeX and mathematical writing
  - List of topics for mini-seminar (will be held at the end of semester)
- gog@kit.edu (room 220)
- Previous edition of this lecture by Johannes Fischer
Content

- Hash functions (static/dynamic)
- Predecessor data structures
- Orthogonal range search structures
- Space-efficient structure for trees, graphs, and binary relations
- Text index structures
- Top-k query structures
Hashing

- Given a set \( S \) of \( n \) object from a large universe \( U \) of size \( u \)
- Map elements of \( S \) fast to natural numbers \( \{0, 1, \ldots, m\} \) with \( m \in O(n) \)
- Used to implement associative arrays
Hashing

- Known from Algo I: chaining/linear probing. Time complexity for lookup $O(1)$ expected time
- Perfect hashing (for a static set of keys): $O(1)$ worst case time
- Cuckoo hashing (dynamic set of keys): $O(1)$ worst case time
Predecessor Queries

- Given a set $S$ of objects from a sorted universe $U$ of size $u$
- For $x \in U$: $\text{predecessor}(x) = \max\{y \leq x | y \in S\}$
- Can be solved in $O(\log \log |U|)$ for integers

- van Emde Boas tree, $x$-fast-trie, $y$-fast-trie
Fast Operations on Trees
Lowest Common Ancestor Queries
Fast Operations on Trees
Level Ancestor Queries

level_ancestor(A, 1)
Succinct/Compressed Data Structures for ...

- ... bitvectors,
- sequences,
- trees,
- and other objects
Succinct/Compressed Data Structures

- space usage = “space for data“ + “space for index”
  
  ![](redundancy)

<table>
<thead>
<tr>
<th>Data Size</th>
<th>Redundancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Information-theoretic</td>
<td>lower-order</td>
</tr>
<tr>
<td>Entropy ($H_0$)</td>
<td>„lower-order“</td>
</tr>
<tr>
<td>Entropy ($H_k$)</td>
<td>„lower-order“</td>
</tr>
</tbody>
</table>

(implicit/in-place) (succinct) (density-sensitive) (compressed)
Succinct/Compressed Data Structures

- space usage = „space for data“ + „space for index“

Data Size | Redundancy
--- | ---
Naive | $O(1)$ (implicit/in-place)
Information-theoretic | lower-order (succinct)
Entropy ($H_0$) | „lower-order“ (density-sensitive)
Entropy ($H_k$) | „lower-order“ (compressed)

Example of a implicit data structure: array-based representation of binary heap

„Information-theoretic” count total number of instances of a given size; take log base 2

„Entropy” is usually an empirical version fo classical Shannon entropy
Range Minimum Queries

- Given an array $A$ of objects from an ordered universe; $n = |A|$
- What is the index of the minimum in an arbitrary range $[i, j]$
- $2n + o(n)$ bit structure solves problem in $O(1)$ time

$\text{RMQ}(2, 5) = 3$
Orthogonal Range Queries
Binary Relations
Some of the structures of the second part of course (structures for dictionaries, sequences, text-indexes, top-k retrieval) are available:

- https://github.com/simongog/sdsl-lite
- http://sux.di.unimi.it/
- https://github.com/ot