Advanced Data Structures

MSc-Vorlesung
Wintersemester 2012/13
KIT
Preliminaries

• 5 ECTS

• lectures in German, slides etc. in English

• prerequisites:
  • Algorithmen II
  • interest in discrete, combinatorial problems

• ~14 lectures (NOT 27.12.12/03.01.13)

• oral exam (20-25 mins)
Preliminaries

• course homepage:
  http://algo2.iti.kit.edu/2056.php
  ‣ slides & script
  ‣ additional course information

• Johannes.Fischer@kit.edu (room 207)

• office hours: Thursday 14-15
• write a **paper** on a data structure **not** covered in the lecture
  ▶ list of topics on course website
  ▶ can be strengthened by experiments
• use **LaTeX** ⇒ learn to write scientifically
  ▶ vector graphics: ipe, xfig, ... (no bitmap!)
• website will provide style files, etc.
The Process

- as author:
  - write paper (5-10 pages)
  - submit to conference management system

- as member of program committee:
  - blind peer reviews & ranking (~2 weeks)

- back to author role:
  - submit final version ⇒ proceedings
  - symposium: 20 min oral presentation
What is a data structure?
What is a Data Structure?

A **Data Structure** specifies how to encode data from some Data Type so as to support the operators specified by a given Abstract Data Type.
Example: Permutations

• data type: **permutation** \( \pi \) of \([1, n]\)

• ADS operations:
  • \( \text{access}_\pi(i) \): return \( \pi[i] \)
  • \( \text{inverse}_\pi(j) \): return \( i \) such that \( \pi[i] = j \)

• data structure: 2 arrays \( A[1, n], A^{-1}[1, n] \)
  • \( \text{access}_\pi(i) = A[i] \)
  • \( \text{inverse}_\pi(j) = A^{-1}[j] \)

• might be OK, might be not (e.g. dynamic??)
Extending Functionality

- have: DS $D$ for ADT $T$
  - e.g. permutations with access/inverse
- want: DS $D'$ for ADT $T'$ with $T' \supseteq T$
  - e.g. perms with access/inverse plus inversions $\pi(i) = |\{j < i : \pi[j] > \pi[i]\}|$
- use $D$ as black box: $D'$ is called index
  - sublinear space possible: $|D'| = o(|D|)$
Implicit DS

• clever storage
  ▶ functionality "for free"

• e.g. heap:

  parent\( (x) = \left\lfloor \frac{x}{2} \right\rfloor \)
Course Contents

• hashing
• predecessor data structures
• integer sorting/searching
• distance oracles
• tree labelings
• lowest common/level ancestors
• range minimum queries
• succinct trees
• text indexing
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: $O(\log \log |U|)$

\[
\text{predecessor}(x)
\]
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
  ▶ $O(n \ lglg \ n)$ for all $w$
Distance Oracles

distance to C?
Tree Labelings: Ancestors
Lowest Common Ancestors
Level Ancestors

2\textsuperscript{nd} ancestor?
Range Minimum Queries

\[
\begin{array}{cccccccc}
5 & 1 & 6 & 3 & 8 & 4 & 2 & 7 & 1 & 4 & 2 & 3 & 7 & 4 & 2 & 1 & 6
\end{array}
\]
Succinct Trees

\[( (()()())((()())(()()))) \]

\[2n \text{ bits}\]

\[n \lg n \text{ bits}\]
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 \(\Rightarrow\) 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>
Time vs Space

e.g. tree + LCA