Midterm 2 Prep

- Midterm topics
- How to study
- How to approach an algorithms problem

Adam Smith
Design paradigms so far

• Basic iterative algorithms
  – Loop invariants
# Topics

<table>
<thead>
<tr>
<th>Topic</th>
<th>Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic iterative algorithms</td>
<td>Loop invariants; induction</td>
</tr>
<tr>
<td>Recursive algorithms</td>
<td>Analysis by strong induction Recurrences</td>
</tr>
<tr>
<td>Divide and Conquer</td>
<td></td>
</tr>
<tr>
<td>Backtracking</td>
<td></td>
</tr>
<tr>
<td>Dynamic programming</td>
<td>Memoization</td>
</tr>
<tr>
<td>Greedy algorithms</td>
<td>Exchange arguments, “greedy stays ahead”, structural arguments</td>
</tr>
<tr>
<td>Basic data structures</td>
<td>ADT’s vs data structures</td>
</tr>
<tr>
<td>Heaps</td>
<td>Array vs tree representation</td>
</tr>
<tr>
<td>Binary search trees</td>
<td></td>
</tr>
<tr>
<td>Balanced binary search trees</td>
<td></td>
</tr>
<tr>
<td>Augmenting data structures</td>
<td>Storing extra information Combining data structures</td>
</tr>
</tbody>
</table>

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova and K. Wayne

L1.3
How To Study

- **Review**
  - Lecture notes, book chapters
  - Homework
  - Exercises on homework
  - Problems in the book
- **For each algorithm / data structure**
  - Make sure you understand
    - What it does
    - How it works
  - Reproduce it
- **Try out your solution**
  - on a classmate
  - on a computer
  - (Best yet: write down pseudocode and see if a classmate can turn it into working code. This is a test of both clarity and correctness!)
- **Solve problems and exercises!**
How to Approach a Problem

- Make sure you understand the problem statement
  - Inputs, outputs
- Work through some simple examples
  - Start small. Work up to trickier examples.
- Write down your algorithm. Top-down:
  - Start with a high-level description (in English)
  - Work your way down to detailed pseudocode
- Try out your solution on examples
  - Play devil’s advocate: try to think of inputs that could cause your algorithm to fail
- Prove correctness
  - Try to convince yourself that your algorithm works for all inputs
- Test your solution
  - On a classmate
  - On a computer

At any point in the process, be prepared to return to an earlier step!