

SPT E2007: The Final Episode

Summarizing Performance & Test

The Final Episode

Ingredients

- Where are we in the course?
- An overview: theory, toolbox and skills
- What now?
- Rules of the Exam
- Fake Exam 1 (groups split and reunite)
- Fake Exam 2 (groups split and reunite)

Where Are We? What is Left?

Lectures (accomplished)



performance



test

Projects (just started)



Theory

- **Asymptotic complexity (big-O, about, proportional to):** how do algorithms **scale**?
- **Worst-case** running time: what time can be guaranteed?
- **Expected** running time: what time can be expected?
- **Randomized** algorithm/data structure: (here) dramatically decreasing probability of the worst case by using randomness.

Theory

- **Asymptotic complexity (big-O, about, proportional to):** how do algorithms **scale**?
- **Worst-case** running time: what time can be guaranteed?
- **Expected** running time: what time can be expected?
- **Randomized** algorithm/data structure: (here) dramatically decreasing probability of the worst case by using randomness.

Theory

- **Asymptotic complexity (big-O, about, proportional to):** how do algorithms **scale**?
- **Worst-case** running time: what time can be guaranteed?
- **Expected** running time: what time can be expected?
- **Randomized** algorithm/data structure: (here) dramatically decreasing probability of the worst case by using randomness.

Theory

- **Asymptotic complexity (big-O, about, proportional to):** how do algorithms **scale**?
- **Worst-case** running time: what time can be guaranteed?
- **Expected** running time: what time can be expected?
- **Randomized** algorithm/data structure: (here) dramatically decreasing probability of the worst case by using randomness.

Theory [continued]

- **Heuristic** algorithm: algorithm guided by an approximation goal function (a heuristics). Typically not optimal or optimal only under some conditions.
- **Asymptotic memory use, in place** algorithms.

Toolbox: Algorithms for Graphs

- Depth-First and Breadth-First search in unweighted graphs: how many edges between to vertices?
- Single Source Shortest Paths (Dijkstra): what is the distance between two points in a network?
- A*: a heuristic algorithm for SSSP.

Toolbox: Algorithms for Graphs

- Depth-First and Breadth-First search in unweighted graphs: how many edges between to vertices?
- Single Source Shortest Paths (Dijkstra): what is the distance between two points in a network?
- A*: a heuristic algorithm for SSSP.

Toolbox: Algorithms for Graphs

- Depth-First and Breadth-First search in unweighted graphs: how many edges between to vertices?
- Single Source Shortest Paths (Dijkstra): what is the distance between two points in a network?
- A^* : a heuristic algorithm for SSSP.

Toolbox: Algorithms for Search and Sort

- Sequential search: is an object in an array/list?
($O(n)$ worst and average)
- Binary search: is an object a sorted array?
($O(\lg(n))$ worst case)
- Sorting: quadratic and optimal ($n \lg n$. Counting based sorts.

Toolbox: Data Structures

- Heap / Priority Queue
- Union-Find
- Dictionaries:
 - Hash-tables
 - BSTs
 - 2-3-4 trees
- Graph representations: adjacency list and adjacency matrix

Skills

- Performance analysis,
- Scalability assessment.
- Implementing data structures as ADTs
- Combining algorithms into bigger programs.
- Choosing algorithms for an application.
- Limited training in design of new algorithms.

What now?

Studying Algorithms: Scalable Computing Specialization

Efficient AI Programming [F2008]

AI techniques for problem solving. Efficient algorithms for hard problems in modern IT applications such as ERP systems, decision support systems, configuration, optimization, computer games.

Advanced Algorithms [E2008]:

Greedy algorithms, divide and conquer, dynamic programming, network flow, reductions, approximation algorithms, randomized algorithms, parallel algorithms

What now?

Studying Development: Models and Programs Specialization

Advanced Object-Oriented Programming [E2007/E2008]

A natural second programming course at ITU. Especially if you are not an experienced programmer yet.

Advanced Models and Programs [F2008]

Advanced techniques of development (advanced language constructs, semantics, model-driven development, code generation—changes slightly over time).

What now?

Reading onwards

- Jon Kleinberg. Eva Tardos. *Algorithm Design*.
- Anany Levitin. *Introduction to The Design and Analysis of Algorithms*.
- Cormen. Leiserson. Rivest. Stein. *Introduction to Algorithms*.

- Eric Roberts. *Thinking recursively with Java*.
- Martin Fowler. *Refactoring*.
- Adam Barr. *Find the bug. A book of incorrect programs*.

What now?

Future thesis prospects

- My general topic is
 - (model-driven) component-based development
 - especially in the context of software product lines.
- My website lists sample project proposals:
 - Software engineering (incl. testing)
 - Open source
 - Formal semantics
 - Algorithms (incl AI)
- A thesis with me? Choose courses from:
 - Scalable Computing
 - Models and Programs
 - Advanced Mobile and Distributed Systems (useful for some topics).

Exam

- Two questions semi-drawn per student on algorithms
- 30 minutes to prepare answers
- up to 25 minutes to present your answers and talk about other aspects of the course (report, testing, also algorithms).
- After an hour you know your grade.

Exam

- No laptops.
- No mobile phones.
- No electronic translators / dictionaries.
- All printed material allowed.
- Have your own book and notes!
 - Impossible to share material with someone who has an exam within a 2 hour interval from you.
- Be at the exam at least 15 minutes earlier.

A Fake Exam: round 1

A Fake Exam: round 2