Epilog (cf. Introduction)

“At this course you will become familiar with a number of fundamental structures and principles underlying the design of efficient algorithms, and will learn to approach new algorithmic problems using these generic paradigms. You will also come to appreciate the possibilities and limitations of theoretical a priori analysis of algorithm efficiency, and learn to perform such analyses in simple cases.”
Why this course? *(cf. Introduction)*

- Efficient algorithms are a core technology for advanced computation.
- The choice of algorithm for a challenging computational task can make a difference of several orders of magnitude in performance.
- Designing efficient algorithms requires knowledge and skill. There is a large literature on algorithmics.
- This course introduces a number of common “design paradigms” that occur repeatedly as underlying principles of efficient algorithms. Important examples of the use of each paradigm are presented.
- These paradigms provide also helpful conceptual tools for approaching new algorithm design tasks.
### Lecture Schedule & Tutorial Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Lecture</th>
<th>Tutorial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 8</td>
<td>1. Introduction</td>
<td>–</td>
</tr>
<tr>
<td>Sep 10</td>
<td>2. Algorithms and their analysis</td>
<td>–</td>
</tr>
<tr>
<td>Sep 15</td>
<td>–</td>
<td>No tutorials</td>
</tr>
<tr>
<td>Sep 17</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Sep 24</td>
<td>4. Divide-and-conquer II</td>
<td>–</td>
</tr>
<tr>
<td>Sep 29</td>
<td>5. Graph algorithms I</td>
<td>3. &amp; 4.</td>
</tr>
<tr>
<td>Oct 1</td>
<td>6. Graph algorithms II</td>
<td>–</td>
</tr>
<tr>
<td>Oct 8</td>
<td>8. Greedy algorithms</td>
<td>–</td>
</tr>
<tr>
<td>Oct 15</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Oct 20</td>
<td>–</td>
<td>No tutorials</td>
</tr>
<tr>
<td>Oct 22</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Lecture Topic</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Oct 29</td>
<td>10. NP-completeness</td>
</tr>
<tr>
<td></td>
<td>Nov 5</td>
<td>12. Randomised algorithms</td>
</tr>
<tr>
<td>46</td>
<td>Nov 10</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Nov 12</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Nov 19</td>
<td>14. Algorithms with numbers II</td>
</tr>
<tr>
<td></td>
<td>Nov 26</td>
<td>16. Fundamental data structures</td>
</tr>
<tr>
<td>49</td>
<td>Dec 1</td>
<td>17. Epilog &amp; exam information</td>
</tr>
<tr>
<td></td>
<td>Dec 3</td>
<td>18. Tutorials Q &amp; A</td>
</tr>
</tbody>
</table>
Grading scheme

Exam: max 54 points
Tutorials: max 6 points
Programming: max 3 bonus points for timely completion
Feedback: max 1 bonus point
Total: 60 points + 4 bonus points

Note: All the programming assignments must be completed and validated as correct before attending the exam.
Feedback (+1p)

- Feedback is greatly appreciated and is given via the Aalto Course Feedback questionnaire.
- The electronic feedback form opens 26 Nov and closes 12 Dec.
- Completing the feedback form will award you one extra point in the course grading scheme.
- Feedback is anonymous except that the set of student IDs of the persons who have completed the form will be available to the course staff for grading purposes.
Default grading scale

- Grade = 0 (fail): 0p – 29.5p
- Grade = 1: 30p – 35.5p
- Grade = 2: 36p – 41.5p
- Grade = 3: 42p – 47.5p
- Grade = 4: 48p – 53.5p
- Grade = 5: 54p – 60p
The Exam (Date and Registration)

- The exam is held on Mon 14 Dec 2015, 13:00–16:00, T1.
- Registration to the exam is mandatory (seriously!), via Oodi (course T-79.4202) during the registration period which ends Mon 7 Dec 23:59.
- Direct link to Oodi:
  https://oodi.aalto.fi/a/opintjakstied.jsp?Kieli=6&Tunniste=T-79.4202&html=1
The Exam (Structure)

- Closed-book: only writing equipment is allowed in the exam. (*Note*: Not even calculator.)
- Three hours
- Four problems
- 12–14 pts per problem
- Total $12 + 14 + 14 + 14 = 54$ pts
The Exam (Philosophy)

- The exam will test your problem-solving skills, not your ability to memorize
- ... but you must know the central/prerequisite concepts and definitions.
- E.g. approximation algorithm, big-$\mathcal{O}$ notation, divide-and-conquer, dynamic programming, heap, graph, greedy algorithm, modular arithmetic, randomized algorithm, RSA, search problem, reduction, recurrence relation, acyclic, path, cycle, tree, spanning tree, flow, cut, matching, topological sort, ...
- One formula that you should understand and memorize, however, is the Master Theorem for the analysis of divide-and-conquer recurrences (Lecture 3). There may be such algorithm design & analysis problems on the exam.
You should develop/have developed a feeling for the growth rates of the usual function classes. Study in particular the examples on Lecture 2, slide 12.

Make sure you understand the basic principles of algorithm design: divide-and-conquer, dynamic programming, greedy algorithms. Dynamic programming in particular often seems to be difficult to get right.

Graph techniques are often useful. If the problem asks for a linear-time graph algorithm, the answer is almost surely based on either depth-first search or breadth-first search.

Review the tutorial problems. If you didn’t work through the basic problems earlier, now is a good time to study them, ideally with a friend or in a group. You can also use the MyCourses forum for open discussion.
Examples of Questions (1/6)

- Is $4^{2010} - 5^{2112}$ divisible by 21?
Examples of Questions (2/6)

▶ Give a linear-time algorithm that decides (outputs “yes” or “no”) whether a directed graph given as input has at least one strongly connected component with at least two vertices.
Algorithm $A$ solves problems by dividing into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time. What is the running time of Algorithm $A$ (in big-$\mathcal{O}$ notation) as a function of the input size $n$?
Examples of Questions (4/6)

Bob the Builder needs to build a house in a hurry. There are $n$ tasks in the building project, all of them mandatory, each taking one week to complete. The prerequisite graph $G$ for the project has a vertex for each task, and a directed edge from task $u$ to task $v$ if and only if $u$ is a prerequisite of $v$. For example, the foundation needs to be ready before the walls are built. (We shall assume that the graph $G$ contains no cycles.)

Give a linear-time algorithm that takes as input the graph $G$ and determines the minimum number of weeks in which the house can be completed, assuming that any number of tasks can take place in any given week. Justify the correctness and complexity of your algorithm.
Professor D. E. Thinker presents the following algorithm for finding a shortest path from $s$ to $t$ in a directed graph $G$ with positive and negative weights on the edges: first add a large constant to the weight of each edge so that all weights become positive, then run Dijkstra’s algorithm from vertex $s$, and return the shortest path found to vertex $t$.

Are you happy with this algorithm? Either prove that the algorithm is correct or give an example where the algorithm fails.
Mr. Joe Q. Profit is planning a sales tour to zero or more cities among $n$ cities labeled 1, 2, …, $n$. The expected revenue to be made at city $i$ is $r_i$, and the cost of travelling from city $i$ to city $j$ is $c_{ij}$. Joe’s home office is in city 0, where no revenue can be made. Joe wants to find the expected maximum profit (revenue minus costs) that can be made by a sales tour that starts and ends at the home office, with no city visited more than once.

Give an algorithm that solves Joe’s problem in time $O(2^n n^2)$. Hint: This is closely related to the Travelling Salesman Problem.
The Exam (Results & Marking)

▶ The exam results will be announced on MyCourses after the exam.
▶ An office hour will be provided for discussing the marking of the exam.
What Next (1/2)?

- Give feedback.
- Register for the exam.
- Take the exam.
What Next (2/2)?

- T-79.4101 Discrete Models and Search (Spring 2016)
- T-79.5207 Advanced Course in Algorithms P (Spring 2016)
- T-79.5501 Cryptology P (Spring 2016)
- T-79.5103 Computational Complexity Theory P (Spring 2016)
Thank You

Questions, Comments, Feedback?