GÖTEBORGS UNIVERSITET

The Board of the IT Faculty

DIT600, Algorithms, 7.5 higher education credits

Second Cycle/A1N

This syllabus in English is the binding document.

1. Confirmation

The Board of the IT Faculty established the course plan at 2006-11-17. It has been revised 2009-09-18 to be valid from spring term 2010.

Field of education: Science

Main field: Computer Science

Department: Computer Science and Engineering

2. Position in the educational system

The course is a part of the Computer Science Master's programme and an elective course at the University of Gothenburg.

The level for the course in relation to degree requirements is Master´s degree, code A1N. The course has course/courses at first cycle level as entry requirements.

3. General prerequisites

The requirement for the course is to have successfully completed a first year studies within the subject Computer Science or equivalent.

4. Course content

- Introduction

- O-notation
- Dynamic programming (Sequence alignment)
- Divide-and Conquer (Quicksort, Mergesort, Inversions, fast multiplication, closest pair)
- Greedy Algorithms (Spanning tress, shortest paths)
- Introduction to NP-completeness
- Introduction to coping with NP-completeness.

5. Learning outcomes

After completion the course the student is expected to be able to:

- recognize that nontrivial computational problems which need to be solved by algorithms appear in various real-world computer applications, and formalize them
- explain why the time efficiency of algorithms is crucial, express the time complexity in a rigorous and scientifically sound manner, analyze the time complexity of algorithms (sum up operations in nested loops, solve standard recurrences, etc.)
- apply the main design techniques for efficient algorithms (greedy, dynamic programming, divide-and-conquer) to problems which are similar to the textbook examples but new
- critically assess algorithmic ideas and resist the temptation to create obvious and seemingly plausible algorithms (which often turn out to be incorrect), actually prove the correctness of algorithms
- model problems with binary relations as graph problems and solve them, using the fundamental graph algorithms
- perform in simple cases the whole development cycle of algorithms: problem analysis, choosing, modifying and combining suitable techniques and data structures, analysis of correctness and complexity, filling in implementation details, looking for possible improvements, etc.
- explain algorithms in writing, so that others can understand how they work, why they are correct and fast, and where they are useful
- implement algorithms properly and evaluate them in theory and experiment
- compare the complexities of problems, perform simple reductions between problems,

explain NP-completeness, recognize various computationally hard problems

- cope, at least in principle, with computationally hard problems, using heuristics, refinements of exhaustive search, approximative solutions, etc.

6. Required reading

See separate literature list

7. Assessment

The course is examined by group exercises and written exam.

A student who has failed a test twice has the right to change examiner, unless weighty argument can be adduced. The written application should be sent to the Department.

8. Grading scale

The course is graded with the following marks: Fail (U), Pass (G), Pass with Distinction (VG).

9. Course evaluation

The course is evaluated through meetings both during and after the course between teachers and student representatives. Further, an anonymous questionnaire can be used to ensure written information. The outcome of the evaluations serves to improve the course by indicating which parts could be added, improved, changed or removed.

10. Additional information

The course is held in English.