UNIVERSITY OF GOTHENBURG

COMPUTER SCIENCE AND ENGINEERING

DIT600 Algorithms, 7.5 higher education credits

Algorithms, 7,5 högskolepoäng

Second Cycle

Confirmation

This course syllabus was confirmed by The IT Faculty Board on 2009-09-18 and was last revised on 2015-11-17 by Department of Computer Science and Engineering to be valid from 2016-08-29, autumn semester of 2016.

Field of education: Science 100%
Department: Computer Science and Engineering

Position in the educational system

This course is a compulsory course in the Computer Science Bachelor's programme and is offered within the framework of several degree programmes. The course is also a single subject course at the University of Gothenburg.

The course can be part of the following programmes: 1) Computer Science, Master's Programme (N2COS), 2) Bachelor's Programme in Mathematics (N1MAT) and 3) Computer Science, Bachelor´s Programme (N1COS)

Main field of studies	*Specialization*
Computer Science-Software Engineering and Tech	A1N, Second cycle, has only first-cycle course/s as entry requirements
Computer Science-Algorithms and Logic	A1N, Second cycle, has only first-cycle course/s as entry requirements

Entry requirements

To be eligible for the course the student should have successfully completed a 7.5 hec course in Data structures (DIT960 Data Structures or equivalent) and 15 hec in programming, including a 7.5 hec course in imperative or object oriented programming (DIT012 Imperative Programming with Basic Object-orientation, or equivalent).

In addition, the student should have successfully completed courses corresponding to 67.5 hec in the subject of Computer Science or Mathematics including knowledge in Discrete Mathematics (via DIT980 Discrete Mathematics for Computer Scientists or equivalent, or the sub-course Introductory Algebra of MMG200 Mathematics I or equivalent).

Applicants must prove knowledge of English: English 6/English B or the equivalent level of an internationally recognized test, for example TOEFL, IELTS.

Learning outcomes

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

Knowledge and understanding

- describe algorithms and their qualities: explain algorithms in writing, so that others can understand how they work, why they are correct and fast, and where they are useful;
- recognize and formalize non-trivial computational problems that appear in various real-world computer applications and which need to be solved by algorithms;
- intractability: recognize intractable problems and other classes of problems like P, NP, NPC;
- prove the correctness of algorithms.

Skills and abilities

- design: apply the main design techniques for efficient algorithms (for instance greedy, dynamic programming, divide-and-conquer, backtracking, heuristics) to problems which are similar to the textbook examples but new;
- 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;
- perform simple reductions between problems, explain NP completeness, recognize various computationally hard problems which tend to appear over and over again in different applications, cope, at least in principle, with computationally hard problems, using heuristics, refinements of exhaustive search, approximative solutions, etc;
- implement algorithms properly and evaluate them in theory and experiment.

Judgement and approach

- critically assess algorithmic ideas and demonstrate the ability to see through obvious and seemingly plausible algorithms that often turn out to be incorrect;
- explain why the seemingly plausible algorithms are incorrect;
- analyse: 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.) i.e. perform an objective evaluation of the performance and be able to compare it to other algorithms' performance.

Course content

The course topics are as follows:

- Introduction. What is an efficient algorithm?
- Tools for analysis of algorithms. O-notation. Analyzing loops and recursive calls. Solving recurrences;
- Data structures and algorithms. Review of basic data structures;
- Combining data structures. Merge-and-find;
- Graph algorithms;
- Greedy algorithms;
- Divide-and-conquer;
- Dynamic programming;
- Backtracking and Implicit search trees. Branch-and-bound;
- Short introduction to local search and approximation algorithms;
- Basic complexity theory. Complexity classes P, NP, and NPC, reductions. Examples of NP-complete problems. Coping with hard problems;
- Short introduction to other design techniques: local search, approximation algorithms, randomized algorithms, preprocessing, network flow.

Form of teaching

The course is given as lectures, combined with tutorial groups for problem solving.

Language of instruction: English

Assessment

The course is examined by an individual written exam carried out in an examination hall and both theoretical and practical programming assignments. The assignments are performed in groups usually consisting of two students.

If a student, who has failed the same examined component twice, wishes to change examiner before the next examination, a written application shall be sent to the

department responsible for the course and shall be granted unless there are special reasons to the contrary (Chapter 6, Section 22 of Higher Education Ordinance).

In cases where a course has been discontinued or has undergone major changes, the student shall normally be guaranteed at least three examination occasions (including the ordinary examination) during a period of at least one year from the last time the course was given.

Grades

The grading scale comprises: Pass with Distinction (VG), Pass (G) and Fail (U).
The grade G reflects fulfilment of learning outcomes for simple and highly structured problems, while VG reflects fulfilment of learning outcomes also for complex and unstructured problems. For VG the student has to demonstrate the ability to apply all thoeries, methods and techniques to the kind of problems discussed in the course.

The final grade is based on the result of the written exam.

Course evaluation

The course is evaluated through meetings both during and after the course between teachers and student representatives. Further, an anonymous questionnaire is 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. The results of and possible changes to the course will be shared with students who participated in the evaluation and students who are starting the course.

Additional information

The course is a joint course together with Chalmers.