



ALGORITHMICS 4

Course Code:

5RCX

Rationale:

Algorithms lie at the heart of computing science: regardless of the increasing processing power and storage capabilities of computers, there is always a need for programmers to employ efficient algorithms and data structures.

Over the last few decades, a wealth of general and specific algorithmic techniques have been developed for computational problems in many application domains as well as for core computing science problems.

Computing Science students should have the opportunity to both broaden their knowledge of general algorithmic techniques and paradigms, and to study in depth sophisticated algorithms and data structures for particular problems. As part of such a study, a student should develop further understanding of the nature and significance of important complexity classes, and should become familiar with a range of techniques for coping with complexity.

Aims:

The aims of the course are:

- to present a broad range of algorithm design methods, with examples chosen to reflect practical applications;
- to enable students to make educated choices between strategies for algorithmic problem-solving;
- to convey the significance of computational complexity, and to present a range of methods for dealing with it in practice.

Objectives:

On successful completion of this course, the student should be able to:

- describe a wide range of efficient algorithms for problems with important applications in domains such as computational geometry, string processing and graph theory
- understand why these algorithms are correct, and prove their correctness
- demonstrate the execution of such algorithms as applied to typical problem instances
- characterise and manipulate advanced data structures such as the suffix tree
- apply algorithmic techniques to specific problems motivated by practical applications
- analyse the worst-case complexity of algorithms using a variety of mathematical techniques
- understand the theory and practical implications of NP-completeness

- identify a variety of NP-complete problems
- explain techniques for coping with complexity, including constant-factor approximation algorithms and polynomial-time approximation schemes
- construct proofs of NP-completeness and inapproximability results

Credits:

10

Pre / Co - requisites:

Algorithmics 3 (4PTW)

Assessment Weightings:

Examination (80%); Practical Exercises: (20%)

Course Texts:

Recommended

- M.T. Goodrich and R. Tamassia, Algorithm Design: Foundations, Analysis and Internet Examples, Wiley, 2002.

•

Background

- G. Ausiello et al, Complexity and Approximation, Springer, 1999.
- T. Cormen et al., Introduction to Algorithms, MIT Press, 2001, 2nd edition.
- U. Manber, Introduction to Algorithms, Addison-Wesley, 1989.
- R. Sedgwick, Algorithms in C++, Addison-Wesley, 1992

Content:

The following list of topics, grouped under a number of headings, gives an indication (though not necessarily an exact breakdown) of the likely content of the course.

Geometric algorithms

- Intersection of two line segments
- Constructing a simple polygon
- Finding the convex hull of a set of points
- Finding a closest pair of a set of points
- Finding all intersections between horizontal and vertical lines

String and text algorithms

- Introduction to the suffix tree data structure
- Applications of suffix trees
- Matching regular expressions
- Longest Common Subsequence (LCS) using memoisation
- Local similarities: the Smith-Waterman algorithm

Matching problems and network flow

- Maximum cardinality matching in bipartite graphs
- Ford-Fulkerson algorithm for maximum flows in networks
- The stable marriage problem
- Applications of matching problems
- Floyd-Warshall algorithm for all-pairs shortest paths

Complexity and approximability

- Lower bounds for problems
- Pseudo-polynomial-time algorithms
- Constant-factor approximation algorithms
- Polynomial-time approximation schemes
- Limits to approximability

Please note that resit examinations in this course are only permitted for Masters students.