Course guide
270021 - A - Algorithmics

Last modified: 30/01/2024

Unit in charge:	Barcelona School of Informatics
Teaching unit:	723 - CS - Department of Computer Science.
Degree:	BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2010). (Optional subject).

Academic year: 2023 **ECTS Credits:** 6.0 **Languages:** Catalan, Spanish, English

LECTURER

Coordinating lecturer: MARIA JOSE SERNA IGLESIAS

Others:

Primer quadrimestre:
MARIA JOSE BLESA AGUILERA - 11, 21
AMALIA DUCH BROWN - 12
SANTIAGO MARCO SOLA - 14, 22
CONRADO MARTÍNEZ PARRA - 21, 22
MARIA JOSE SERNA IGLESIAS - 11, 12, 13, 14

Segon quadrimestre:
MARIA JOSE BLESA AGUILERA - 12, 13
CONRADO MARTÍNEZ PARRA - 11, 12, 13
MARIA JOSE SERNA IGLESIAS - 11

PRIOR SKILLS

- Familiarity with the basic programming techniques and the programming language C + +: iterations, alternative, recursive functions, parameter passing, pointers, references, dynamic memory, classes, objects, methods, ...

- Knowledge of basic algorithmic concepts: efficiency of algorithms, asymptotic notation, graphs, graph traversals, data structures (lists, search trees, hashing, heaps, ...)

- Basic knowledge of discrete mathematics, linear algebra and calculus

- Basic knowledge of probability theory and statistics

- Basic knowledge of computer architecture and memory hierarchy

REQUIREMENTS

- Prerequisite EDA
- Pre-Corequisite PE
- Corequisite PROP

DEGREE COMPETENCES TO WHICH THE SUBJECT CONTRIBUTES

Specific:
CCO1.1. To evaluate the computational complexity of a problem, know the algorithmic strategies which can solve it and recommend, develop and implement the solution which guarantees the best performance according to the established requirements.
CCO2.5. To implement information retrieval software.
CCO3.1. To implement critical code following criteria like execution time, efficiency and security.
CCO3.2. To program taking into account the hardware architecture, using assembly language as well as high-level programming languages.
CT1.2C. To use properly theories, procedures and tools in the professional development of the informatics engineering in all its fields (specification, design, implementation, deployment and products evaluation) demonstrating the comprehension of the adopted compromises in the design decisions.
CT4.1. To identify the most adequate algorithmic solutions to solve medium difficulty problems.
CT4.2. To reason about the correction and efficiency of an algorithmic solution.
CT5.2. To know, design and use efficiently the most adequate data types and data structures to solve a problem.
CT5.3. To design, write, test, refine, document and maintain code in an high level programming language to solve programming problems applying algorithmic schemas and using data structures.
CT5.4. To design the programs¿ architecture using techniques of object orientation, modularization and specification and implementation of abstract data types.

Generical:
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

TEACHING METHODOLOGY

Theory lectures will be magistral, with theoretical explanations from the teacher, interspersed with numerous examples. Students should actively participate with their questions and comments along these classes. Each week there will be two hours of lectures and two hours devoted to problems. In problem sessions, there will be disucssion of the solutions proposed by the students to the exercises posed by the teacher in advance (more complex assignments, with which the student has been able to work during the week, autonomously) or of the short exercises posed during the class to be worked by teams of two students or individually. Occasionally, the students could be required to expose their solutions to the rest of their mates.

To complement personal study and practical problem solving in paper, a programming project will be proposed. In the project, the student must design and encode programs in C++ that solve the proposed problems, for instance, to implement two (or more) algorithms that solve the same problem and to carry out experiments that allow to compare the performance of the algorithms, and at the same time, to compare these performances with the predictions of the theoretical analysis. Each student will have to develop the project in a team of two or three people.
This project will be used to assess the autonomous learning skills, as it will require the study of a particular subject, related with those in the course, but no exposed in any lecture.

LEARNING OBJECTIVES OF THE SUBJECT

1.Knowing greedy algorithms, to identify when and how you can apply them, knowing the most common techniques to prove correctness and becoming familiar with some basic greedy algorithms, e.g, Dijkstra's algorithm, Kruskal's and Prim's algorithms.
2.Understanding the dynamic programming scheme, to identify when and how you can apply it and become familiar with some fundamental dynamic programming algorithms, eg, Floyd's algorithm or calculating the edit distance
3.Knowing the basic problem of optimal flows on networks, to become familiar with a basic algorithm (Ford-Fulkerson), to understand the maxflow-mincut theorem, to recognize when a problem can be formulated in terms of a flow problem
4.To understand the importance of randomization in the design of algorithms and data structures, to become familiar with some basic techniques of probabilistic analysis needed to study the efficiency of randomized algorithms and with some classic examples.
5.To know about some computational problems that arise in specific areas of CS as diverse as search in document databases, protein and genomic databases, geographic information systems, content-based information retrieval, data compression, etc. and to know some advanced data structures to respond to these needs
6.Becoming familiar with the use of algorithmic design principles for the design of data structures and to learn some essential techniques to obtain implementations which yield maximum efficiency and take advantage of the specific hardware features supporting the execution
7.To develop the necessary habits, attitudes and skills to be able to study, alone or in a team, a specific subject, making use of available sources of information (bibliography, web, ...) and to achieve the level of knowledge and compression of the subject which is enough to explain it to others, writing a summary and preparing a supplementary visual material
8.To understand some basic principles for the design of computational experiments and to learn basic techniques of data collection, validation and statistical analysis of the collected data, and how to draw conclusions, recognizing the need, usefulness and limitations of experimental studies in design and implementation of algorithms and data structures

STUDY LOAD

Type	Hours	Percentage
Hours large group	30,0	20.00
Guided activities	6,0	4.00
Hours medium group	30,0	20.00
Self study	84,0	56.00

Total learning time: 150 h

CONTENTS

Basic Algorithmic Concepts

Description:
Worst case analysis. Asymptotic Notation. Divide and conquer. Analysis of recursive algorithms. Linear sorting. Graph algorithms. Randomization.

Greedy algorithms

Description:
The scheme for greedy algorithms. Task scheduling. Bellman-Ford' and Johnson's algorithms for shortest paths. Kruskal's and Prim's algorithms for minimum spanning trees. Union-find. Huffman codes.

Dynamic Programming

Description:
Principle of optimality. Memoization. Floyd-Warshall algorithm for all-shortest paths. Traveling salesman problem. Knapsack problem. Other examples.

Network Flows

Description:
Basic concepts. Maxflow-mincut theorem. The Ford-Fulkerson algorithm. Applications: Matching and Edge disjoint paths. Duality.

Advanced Data Structures and Algorithms

Description:
A selection of some of the following algorithms and/or data structures (or others). Linear Programming. Fibonacci heaps. Hashing. Bloom Filters. Blockchains. Map Reduce. Random graphs. Page Rank.

ACTIVITIES

Basic Algorithmic Concepts

Description:
To remind the basic concepts learnt in previous courses, and to become familiar with the terminology and notation that will be used throughout the course. Learn other basic algorithmic techniques.

Full-or-part-time: 18h
Theory classes: 4h
Practical classes: 4h
Self study: 10h

Greedy algorithms

Description:
Attend lectures and problem sessions where this subject is exposed, do the exercises
proposed by the teacher to do at home or in class

Specific objectives:
1

Related competencies :
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

Full-or-part-time: 19h
Theory classes: 5h
Practical classes: 4h
Self study: 10h

Dynamic Programming

Description:
Attend lectures and problem sessions where this subject is exposed, do the exercises
proposed by the teacher to do at home or in class

Specific objectives:
2

Related competencies :
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

Full-or-part-time: 32h
Theory classes: 7h
Practical classes: 10h
Self study: 15h

Network Flows

Description:
Attend lectures and problem sessions where this subject is exposed, do the exercises
proposed by the teacher to do at home or in class

Specific objectives:
3

Related competencies :
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

Full-or-part-time: 32h
Theory classes: 7h
Practical classes: 10h
Self study: 15h

Advanced Data Structures

Description:
Attend lectures and problem sessions where this subject is exposed, do the exercises
proposed by the teacher to do at home or in class

Specific objectives:
5, 6

Related competencies :
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

Full-or-part-time: 8h
Theory classes: 2h
Practical classes: 2h
Self study: 4h

Autonomous Learning Project

Description:
The teacher makes a brief interview with each of the student teams to discuss the written documentation and audiovisual material delivered, find out how the activity has been developed over the course, finding out about planning issues, if there was any, finding out what mechanisms have been used to coordinate the parts of the work carried out by the team, etc.. The teacher assesses the degree of learning of the proposed subject by the students by means of short specific questions and the achievement of the activity goals

Specific objectives:
1, 2, 3, 4, 5, 6, 7, 8

Related competencies :
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

Full-or-part-time: 15h
Self study: 15h

Final Exam/Second partial

Specific objectives:
1, 2, 3, 4, 5, 7

Related competencies :
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

Full-or-part-time: 13h
Guided activities: 3h
Self study: 10h

Mid term exam

Description:
Partial examination outside teaching hours.

Specific objectives:
1, 2, 5, 6, 7

Related competencies :
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

Full-or-part-time: 7h
Guided activities: 2h
Self study: 5h

Project supervision

Description:
Solve doubts, follow up of the activity, etc.

Specific objectives:
6, 7, 8

Related competencies :
G7. AUTONOMOUS LEARNING: to detect deficiencies in the own knowledge and overcome them through critical reflection and choosing the best actuation to extend this knowledge. Capacity for learning new methods and technologies, and versatility to adapt oneself to new situations.

Full-or-part-time: 6h
Guided activities: 6h

GRADING SYSTEM

The final grade (NF) is calculated from the note of the resolution of algorithmic problems (A), that of the mid term exam (M) (subject corresponding to the 6 -7 first weeks of the course), the final exam (F) and the project note associated with autonomous learning (P).

The final mark is obtained by the formula

NF = 0.7 max(0.4 M + 0.6 F, F) + 0.2 P +0.1 A

The teacher will assess the degree of acquisition of the "Autonomous learning" skill from the score earned in a programming project that involves autonomous learning on the part of the students. The score P will be graded in a numerical scale from 0 to 10.

The qualitative grade for "Autonomous learning" is given according to the range in which the numerical grade falls: [0,5) => D, [5,6.5) => C, [6.5, 8.5) => B, [8.5, 10] => A

BIBLIOGRAPHY

Basic:
- Kleinberg, J.; Tardos, E. Algorithm design. Pearson, 2014. ISBN 9781292023946.
- Dasgupta, S.; Papadimitriou, C.; Vazirani, U. Algorithms. McGraw-Hill, 2008. ISBN 9780073523408.
- Moore, C.; Mertens, S. The Nature of computation. Oxford University press, 2011. ISBN 9780199233212.

Complementary:
- Brassard, G.; Bratley, P. Fundamentals of algorithmics. Prentice-Hall International, 1996. ISBN 013073487X.
- Skiena, S. The algorithm design manual. 2nd ed. Springer, 2008. ISBN 9781848000698.
- Skiena, S.S. The algorithm design manual. Third edition. Cham: Springer, 2020. ISBN 9783030542559.
- Mehta, D.P.; Sahni, S. (eds.). Handbook of data structures and applications. Chapman & Hall/CRC, 2005. ISBN 1584884355.
- Lamport, L. LaTeX: a document preparation system: user's guide and reference manual. 2nd ed. Addison-Wesley, 1994. ISBN 0201529831.
- Motwani, R.; Raghavan, P. Randomized Algorithms. Cambridge University Press, 1995. ISBN 0521474655.
- Mitzenmacher, M.; Upfal, E. Probability and computing: randomized algorithms and probabilistic analysis. Cambridge University Press, 2005. ISBN 0521835402.
- Easley, D.; Kleinberg, J. Networks, crowds, and markets: reasoning about a highly connected world. Cambridge University Press, 2010. ISBN 9780521195331.
- Cormen, T.H [et al.]. Introduction to algorithms. 4th ed. Cambridge: MIT Press, 2022. ISBN 9780262046305.
- Sedgewick, R.; Wayne, K. Algorithms. 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2011. ISBN 9780321573513.