270021 - A - Algorithmics

Coordinating unit: 270 - FIB - Barcelona School of Informatics
Teaching unit: 723 - CS - Department of Computer Science
Academic year: 2018
Degree: BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2010). (Teaching unit Optional)
BACHELOR'S DEGREE IN ENGINEERING PHYSICS (Syllabus 2011). (Teaching unit Optional)
ECTS credits: 6

Teaching languages: Catalan, Spanish

Teaching staff
Coordinator: - Maria Jose Serna Iglesias (mjserna@cs.upc.edu)
Others: - Josep Diaz Cort (diaz@cs.upc.edu)
- M. Jose Blesa Aguilera (mjblesa@cs.upc.edu)

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
- Pre-Corequisite PE
- Corequisite PROP
- Prerequisite EDA

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.
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, e.g., 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 such as the randomized quicksort, the skip list data structure, Rabin primality test algorithm or the pattern matching algorithm for strings of Karp-Rabin.
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.

<table>
<thead>
<tr>
<th>Study load</th>
<th>Theory classes:</th>
<th>30h</th>
<th>20.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Practical classes:</td>
<td>30h</td>
<td>20.00%</td>
</tr>
<tr>
<td></td>
<td>Laboratory classes:</td>
<td>0h</td>
<td>0.00%</td>
</tr>
<tr>
<td></td>
<td>Guided activities:</td>
<td>6h</td>
<td>4.00%</td>
</tr>
<tr>
<td></td>
<td>Self study:</td>
<td>84h</td>
<td>56.00%</td>
</tr>
</tbody>
</table>
## Basic Algorithmic Concepts

**Degree competences to which the content contributes:**

**Description:**
- Worst case analysis.
- Asymptotic Notation.
- Divide and conquer.
- Analysis of recursive algorithms.
- Hashing.
- Graph algorithms.
- Randomization.

## Greedy algorithms

**Degree competences to which the content contributes:**

**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

**Degree competences to which the content contributes:**

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

## Network Flows

**Degree competences to which the content contributes:**

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

## Advanced Data Structures and Algorithms

**Degree competences to which the content contributes:**

**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.
## Planning of activities

<table>
<thead>
<tr>
<th>Basic Algorithmic Concepts</th>
<th>Hours: 22h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory classes: 6h</td>
<td></td>
</tr>
<tr>
<td>Practical classes: 6h</td>
<td></td>
</tr>
<tr>
<td>Laboratory classes: 0h</td>
<td></td>
</tr>
<tr>
<td>Guided activities: 0h</td>
<td></td>
</tr>
<tr>
<td>Self study: 10h</td>
<td></td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Greedy algorithms</th>
<th>Hours: 14h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory classes: 4h</td>
<td></td>
</tr>
<tr>
<td>Practical classes: 4h</td>
<td></td>
</tr>
<tr>
<td>Laboratory classes: 0h</td>
<td></td>
</tr>
<tr>
<td>Guided activities: 0h</td>
<td></td>
</tr>
<tr>
<td>Self study: 6h</td>
<td></td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Dynamic Programming</th>
<th>Hours: 20h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory classes: 4h</td>
<td></td>
</tr>
<tr>
<td>Practical classes: 6h</td>
<td></td>
</tr>
<tr>
<td>Laboratory classes: 0h</td>
<td></td>
</tr>
<tr>
<td>Guided activities: 0h</td>
<td></td>
</tr>
<tr>
<td>Self study: 10h</td>
<td></td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Network Flows</th>
<th>Hours: 26h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theory classes: 8h</td>
<td></td>
</tr>
<tr>
<td>Practical classes: 8h</td>
<td></td>
</tr>
<tr>
<td>Laboratory classes: 0h</td>
<td></td>
</tr>
<tr>
<td>Guided activities: 0h</td>
<td></td>
</tr>
<tr>
<td>Self study: 10h</td>
<td></td>
</tr>
</tbody>
</table>
### Advanced Data Structures

<table>
<thead>
<tr>
<th>Description:</th>
<th>Hours: 21h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attend lectures and problem sessions where this subject is exposed, do the exercises proposed by the teacher to do at home or in class</td>
<td>Theory classes: 8h</td>
</tr>
<tr>
<td></td>
<td>Practical classes: 6h</td>
</tr>
<tr>
<td></td>
<td>Laboratory classes: 0h</td>
</tr>
<tr>
<td></td>
<td>Guided activities: 0h</td>
</tr>
<tr>
<td></td>
<td>Self study: 7h</td>
</tr>
</tbody>
</table>

### Problem assignments

<table>
<thead>
<tr>
<th>Description:</th>
<th>Hours: 14h</th>
</tr>
</thead>
<tbody>
<tr>
<td>The goal is to solve problems proposed by the teacher. The assigments must be done individually. There will be at least 4 days between the date on which the assignment are publicized and the problems class in which the solution has to be presented and one additional week to hand out the written resolution. You can expect to get two or three problems along the course.</td>
<td>Theory classes: 0h</td>
</tr>
<tr>
<td></td>
<td>Practical classes: 0h</td>
</tr>
<tr>
<td></td>
<td>Laboratory classes: 0h</td>
</tr>
<tr>
<td></td>
<td>Guided activities: 3h</td>
</tr>
<tr>
<td></td>
<td>Self study: 11h</td>
</tr>
</tbody>
</table>

### Autonomous Learning Project

<table>
<thead>
<tr>
<th>Description:</th>
<th>Hours: 15h</th>
</tr>
</thead>
<tbody>
<tr>
<td>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</td>
<td>Guided activities: 0h</td>
</tr>
<tr>
<td></td>
<td>Self study: 15h</td>
</tr>
</tbody>
</table>
The final grade (NF) is calculated from the note of the resolution of algorithmic problems (A), that of the first partial written tests (M) (subject corresponding to the 6-7 first weeks of the course), second partial (E) (subject corresponding to the last 6-7 weeks of the course) and final exam (F) and the project note associated with autonomous learning (P) according to the formulas:

\[
NF1 = 0.7 \times 0.5 \times (M + E) + 0.1 \times A + 0.2 \times P \\
NF2 = 0.7 \times F + 0.1 \times A + 0.2 \times P
\]

One week before the end of the semester each student with grade \( M \geq 3 \) must choose to take the second partial or the final exam on the day assigned for the final exam in the academic calendar. In the first case \( NF = NF1 \) and in the second case \( NF = NF2 \). If \( M < 3 \), \( NF = NF2 \).

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) \( \Rightarrow \) D, [5,6.5) \( \Rightarrow \) C, [6.5, 8.5) \( \Rightarrow \) B, [8.5, 10] \( \Rightarrow \) A
Bibliography

Basic:


Complementary: