270012 - EDA - Data Structures and Algorithmics

Coordinating unit: 270 - FIB - Barcelona School of Informatics
Teaching unit: 723 - CS - Department of Computer Science
Academic year: 2019
Degree: BACHELOR'S DEGREE IN INFORMATICS ENGINEERING (Syllabus 2010). (Teaching unit Compulsory)
BACHELOR'S DEGREE IN ENGINEERING PHYSICS (Syllabus 2011). (Teaching unit Optional)
ECTS credits: 6
Teaching languages: Catalan, Spanish, English

Teaching staff

Coordinator: - Enric Rodriguez Carbonell (erodri@cs.upc.edu)
Others: - Albert Oliveras Llunell (oliveras@cs.upc.edu)
- Antoni Lozano Bojados (antoni@cs.upc.edu)
- Conrado Martinez Parra (conrado@cs.upc.edu)
- Pilar Nivela Alos (nivela@cs.upc.edu)
- Salvador Roura Ferret (roura@cs.upc.edu)

Prior skills

Students are expected to be familiar with imperative object-based programming techniques:
- parameter passing
- classes,
- objects,
- methods,
- pointers,
- dynamic memory,
- genericity,
- recurrence,
- standard classes usage,
- iterators

They are also expected to know at least one imperative object-oriented language, preferably C++.

Critical thinking capacity and mathematical maturity are required too.

Requirements

- Prerequisite PRO1
- Prerequisite PRO2

Degree competences to which the subject contributes

Specific:
CT2.3. To design, develop, select and evaluate computer applications, systems and services and, at the same time, ensure its reliability, security and quality in function of ethical principles and the current legislation and normative.
CT2.4. To demonstrate knowledge and capacity to apply the needed tools for storage, processing and access to the information system, even if they are web-based systems.
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.
CT4.3. To demonstrate knowledge and capacity to apply the fundamental principles and the basic techniques of the intelligent systems and its practical application.

CT5.1. To choose, combine and exploit different programming paradigms, at the moment of building software, taking into account criteria like ease of development, efficiency, portability and maintainability.

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.

CT5.5. To use the tools of a software development environment to create and develop applications.

CT8.6. To demonstrate the comprehension of the importance of the negotiation, effective working habits, leadership and communication skills in all the software development environments.

CT8.7. To control project versions and configurations.

Generical:

G6. SOLVENT USE OF THE INFORMATION RESOURCES: To manage the acquisition, structuring, analysis and visualization of data and information of the field of the informatics engineering, and value in a critical way the results of this management.

Teaching methodology

Topics are explained in a practical way through the use of numerous examples.

Theory classes cover the required concepts and techniques, which are put into practice in the problem-solving and laboratory classes by means of a collection of problems and exercises from an automatic judge.

The two-hour theory class will take place once a week. The two-hour laboratory class will take place once a fortnight. The two-hour problem-solving class will take place once a fortnight.

Programming for the game integrates knowledge and skills of the entire course.

The C++ programming language is used for this course.

Learning objectives of the subject

1. Understand the definitions of the Big-O, Omega and Theta asymptotic notations and their usefulness in characterising the efficiency of algorithms in time and space.
2. Calculate the efficiency of iterative algorithms using appropriate calculation rules.
3. Describe the efficiency of recursive algorithms using recurrence relations and understand and apply master theorems to solve them.
4. Design algorithms for solving various problems of medium difficulty with time and space constraints.
5. Compare the efficiency of different algorithms for solving the same problem and select the most appropriate one.
6. Understand, explain, design, analyse, compare and implement algorithms (such as mergesort, quicksort, Karatsuba, Strassen, etc.) using divide and conquer.
7. Understand, explain, design, analyse, compare and implement the main data structures that can be used to implement dictionaries (tables, sorted tables, lists, sorted lists, hash tables, binary search trees, AVL trees).
8. Understand, explain, design, analyse, compare and implement the main data structures that can be used to implement priority queues (trees, heaps).
9. Understand, explain, design, analyse, compare and implement algorithms that solve classic graph problems such as traversals, topological sorts, shortest paths, etc.
10. Understand, explain, design, analyse, compare and implement exhaustive search algorithms using the backtracking technique.

11. Identify computational limits: understand the implications of the question "P = NP?", understand the statement of Cook-Levin's Theorem, and recognise and identify several classic NP-complete problems.

12. Complete and modify C++ programming language implementations of several algorithms to solve medium-difficulty problems.

13. Identify and propose solutions to efficiency problems in algorithms and programs written in the C++ programming language.

14. Analyse a strategy game for designing and programming an effective, efficient, collaborative and competitive player that maximises the chances of winning the game and is capable of establishing partnerships and coordinating with other players.

15. Apply information search strategies (for bibliographic references, scientific articles, patents, credible web resources, etc.) and, making an ethical use of the compiled information and properly citing sources, produce a well-structured document describing a well-known algorithm that solves a given problem.

16. Compute the cost of an algorithm in the worst, best and average cases.

**Study load**

<table>
<thead>
<tr>
<th>Total learning time: 150h</th>
<th>Hours large group:</th>
<th>30h</th>
<th>20.00%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hours medium group:</td>
<td>15h</td>
<td>10.00%</td>
</tr>
<tr>
<td></td>
<td>Hours small group:</td>
<td>15h</td>
<td>10.00%</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>
## Content

### Analysis of Algorithms

<table>
<thead>
<tr>
<th>Degree competences to which the content contributes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Cost in time and space. Worst case, best case and average case. Asymptotic notation. Analysis of the cost of iterative and recursive algorithms.</td>
</tr>
</tbody>
</table>

### Divide and conquer

<table>
<thead>
<tr>
<th>Degree competences to which the content contributes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Principles: partition into subproblems, recombination of solutions. Examples: mergesort, quicksort, Karatsuba’s algorithm for multiplying large numbers, Strassen’s algorithm for matrix multiplication.</td>
</tr>
</tbody>
</table>

### Dictionaries

<table>
<thead>
<tr>
<th>Degree competences to which the content contributes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Operations of dictionaries and ordered dictionaries. Basic implementations: tables and lists. Advanced implementations: hash tables, binary search trees, AVL trees.</td>
</tr>
</tbody>
</table>

### Priority Queues

<table>
<thead>
<tr>
<th>Degree competences to which the content contributes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong> Operations of priority queues. Implementations with heaps. Heapsort.</td>
</tr>
</tbody>
</table>

### Graphs

<table>
<thead>
<tr>
<th>Degree competences to which the content contributes:</th>
</tr>
</thead>
</table>

### Exhaustive Search and Generation

| Degree competences to which the content contributes: |
### Notions of Intractability

**Description:**

**Degree competences to which the content contributes:**
Basic introduction to P and NP classes, Cook-Levin's Theorem, reductions and NP-completeness.
### Planning of activities

<table>
<thead>
<tr>
<th>Topic</th>
<th>Hours: 16h</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysis of Algorithms</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Hours:</strong></td>
<td></td>
</tr>
<tr>
<td>Theory classes: 4h</td>
<td></td>
</tr>
<tr>
<td>Practical classes: 2h</td>
<td></td>
</tr>
<tr>
<td>Laboratory classes: 2h</td>
<td></td>
</tr>
<tr>
<td>Guided activities: 0h</td>
<td></td>
</tr>
<tr>
<td>Self study: 8h</td>
<td></td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td>Topic 1 development.</td>
<td></td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td>1, 2, 3, 4, 5, 16</td>
</tr>
<tr>
<td><strong>Divide and conquer</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Hours:</strong></td>
<td></td>
</tr>
<tr>
<td>Theory classes: 4h</td>
<td></td>
</tr>
<tr>
<td>Practical classes: 2h</td>
<td></td>
</tr>
<tr>
<td>Laboratory classes: 2h</td>
<td></td>
</tr>
<tr>
<td>Guided activities: 0h</td>
<td></td>
</tr>
<tr>
<td>Self study: 8h</td>
<td></td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td>Topic 2 development.</td>
<td></td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td>3, 4, 5, 6, 12, 13</td>
</tr>
<tr>
<td><strong>Dictionaries</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Hours:</strong></td>
<td></td>
</tr>
<tr>
<td>Theory classes: 4h</td>
<td></td>
</tr>
<tr>
<td>Practical classes: 2h</td>
<td></td>
</tr>
<tr>
<td>Laboratory classes: 2h</td>
<td></td>
</tr>
<tr>
<td>Guided activities: 0h</td>
<td></td>
</tr>
<tr>
<td>Self study: 8h</td>
<td></td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td>Topic 3 development.</td>
<td></td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td>4, 5, 7, 12, 13</td>
</tr>
<tr>
<td><strong>Mid-semester written exam</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Hours:</strong></td>
<td>6h</td>
</tr>
<tr>
<td>Guided activities: 2h</td>
<td></td>
</tr>
<tr>
<td>Self study: 4h</td>
<td></td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td></td>
</tr>
<tr>
<td>Learning objectives corresponding to topics 1 and 2 will be assessed.</td>
<td></td>
</tr>
<tr>
<td><strong>Specific objectives:</strong></td>
<td>1, 2, 3, 4, 5, 6, 12</td>
</tr>
</tbody>
</table>
## Priority Queues

**Description:**
Topic 4 development.

**Specific objectives:**
4, 5, 8, 12, 13

<table>
<thead>
<tr>
<th>Hours</th>
<th>Theory classes: 2h</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Practical classes: 1h</td>
</tr>
<tr>
<td></td>
<td>Laboratory classes: 1h</td>
</tr>
<tr>
<td></td>
<td>Guided activities: 0h</td>
</tr>
<tr>
<td></td>
<td>Self study: 4h</td>
</tr>
</tbody>
</table>

## Graphs

**Description:**
Topic 5 development.

**Specific objectives:**
4, 5, 9, 12, 13

<table>
<thead>
<tr>
<th>Hours</th>
<th>Theory classes: 4h</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Practical classes: 2h</td>
</tr>
<tr>
<td></td>
<td>Laboratory classes: 2h</td>
</tr>
<tr>
<td></td>
<td>Guided activities: 0h</td>
</tr>
<tr>
<td></td>
<td>Self study: 8h</td>
</tr>
</tbody>
</table>

## Game

**Description:**
Goals corresponding to learning objective 15 will be assessed.

A statement describing a strategy game will be published. Students will have to program a player for this game (i.e. implement a strategy aimed at winning).

A competition will be carried out in which students will play against each other, from which a ranking will be obtained. To participate in the competition, the players of the students will have to pass a qualification test.

The grade corresponding to this part will be computed from the position in the ranking in a proportional way, ensuring that the winner gets a 10 and that all students with a qualified player get a minimum of 5. Those students who have not been able to qualify a player will get a 0.

**Specific objectives:**
14
### Exhaustive Search and Generation

| **Description:** | **Hours:** 18h  
Theory classes: 4h  
Practical classes: 2h  
Laboratory classes: 4h  
Guided activities: 0h  
Self study: 8h |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specific objectives:</strong></td>
<td></td>
</tr>
</tbody>
</table>
4, 5, 10, 12, 13 |

### Notions of Intractability and Undecidibility

| **Description:** | **Hours:** 10h  
Theory classes: 4h  
Practical classes: 2h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 4h |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specific objectives:</strong></td>
<td></td>
</tr>
</tbody>
</table>
11 |

### Computer-based mid-term exam

| **Description:** | **Hours:** 6h 30m  
Guided activities: 2h 30m  
Self study: 4h |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Specific objectives:</strong></td>
<td></td>
</tr>
</tbody>
</table>
7, 8, 9, 10 |

Laboratory aspects, i.e. implementation aspects, of the topics covered up to the date will be assessed.

Students will be issued two or three problems in front of their computer. These problems will have a statement, one or more public test suites and, possibly, an already implemented code. When students are ready to submit programs for particular problems, they upload them to an automatic judge which returns a verdict on program behaviour. Students can submit up to 10 solutions for the same problem. The lecturer will correct the last solution submitted for each problem.
## Consolidation.

**Hours:** 12h 30m  
Theory classes: 4h  
Practical classes: 2h  
Laboratory classes: 2h  
Guided activities: 0h  
Self study: 4h 30m

### Description:
Review.

### Specific objectives:
1, 2, 3, 4, 5, 12, 13

## RGF library science tutorials.

**Hours:** 2h 30m  
Theory classes: 0h  
Practical classes: 0h  
Laboratory classes: 0h  
Guided activities: 0h  
Self study: 2h 30m

### Description:
Self-learning through the BRGF library science tutorials on: intellectual property, the ethical use of information and the use of reference management software.

### Specific objectives:
15

## Practical on sound use of computer resources.

**Hours:** 3h  
Guided activities: 0h  
Self study: 3h

### Description:
Goals corresponding to learning objective 16 will be assessed.

A statement will be published consisting in the description of a computational problem and the name of an algorithm to solve it. Students will conduct research (in the library, on the web, etc.) into the problem and the algorithm and will write a brief, well-structured document that properly lists sources.

The document should be handed in on the day of the final exam.

Students' generic competences will be assessed on the basis of this document.

### Specific objectives:
15
270012 - EDA - Data Structures and Algorithmics

**Final Exam**

| Hours: 13h |
| Guided activities: 3h |
| Self study: 10h |

**Description:**
Learning objectives for content corresponding to topics 1 to 7 will be assessed.

**Specific objectives:**
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

**Qualification system**

NPP = written mid-semester exam grade (0 to 10)
NPO = computer-based mid-semester exam grade (0 to 10)
NF = final exam grade (0 to 10)
NJ = game grade (0 to 10)

GRADE = min(10, max (30%NPP + 30%NPO + 30%NF + 20%NJ, 30%NPO + 60%NF + 20%NJ))
# 270012 - EDA - Data Structures and Algorithmics

## Bibliography

### Basic:


### Complementary:


### Others resources:

- **Hyperlink**
  - [http://ocw.mit.edu/courses/#electrical-engineering-and-computer-science](http://ocw.mit.edu/courses/#electrical-engineering-and-computer-science)
  - [https://www.jutge.org/](https://www.jutge.org/)
  - [http://uva.onlinejudge.org/](http://uva.onlinejudge.org/)
  - [http://www.topcoder.com](http://www.topcoder.com)
http://www.cs.sunysb.edu/~algorith/