HOGESCHOOL ROTTERDAM / CMI

Algorithms

INFDEV03-6A

Number of study points:	4 ects
Course owners:	G. Costantini

Module description

Module name:	Algorithms
Module code:	INFDEV03-6A
Study points and hours of effort for full-time students:	This module gives 4 ects, in correspondence with 112 hours: • 2 x 8 hours frontal lecture • the rest is self-study
Examination:	Written exam and practical assessment
Course structure:	Lectures
Prerequisite knowledge:	Object oriented programming
Learning tools:	• Book: *Introduction to Algorithms*; authors C.E. Leiserson, C. Stein, R. Rivest, and T.H. Cormen • Lesson slides (pdf): found on N@tschool • Multiple choice questionnaires published after each lesson • Visual Studio Community
Connected to competences:	• Realisation • Analysis
Learning objectives:	At the end of the course, the student can: • Analyse the performance of an algorithm [PERF] • Infer the behavior of algorithms involving basic data structures and analyse their performance [$\texttt{DS}^{\texttt{A}}$] • Implement sorting algorithms [$\texttt{SORT}^{\texttt{I}}$] • Infer the behavior of sorting algorithms and analyse their performance [$\texttt{SORT}^{\texttt{A}}$] • Implement recursive data structures [$\texttt{REC}^{\texttt{I}}$] • Infer the behavior of algorithms manipulating recursive data structures and analyse their performance [$\texttt{REC}^{\texttt{A}}$] • Implement graphs representation and algorithms [$\texttt{GRAPH}^{\texttt{I}}$] • Infer the behavior of algorithms manipulating graphs and analyse their performance [$\texttt{GRAPH}^{\texttt{A}}$]
Course owners:	G. Costantini
Date:	16 november 2018

1 General description

Designing and manipulating efficient data structures is at the foundation of computer programming. These data structures solve complex problems through well-known, highly difficult techniques that would simply take too long to rediscover for every application. When faced with certain classes of issues, lack of knowledge of algorithms and data structures might significantly impact a programmer's ability to tackle a given problem efficiently and effectively.
In this course we are going to explain some popular algorithms and data structures used in a variety of scenario's encountered in practice when dealing with structuring and traversal of data.

1.1 Relationship with other teaching units

This course builds upon the development courses of the first year.

Knowledge acquired through the algorithms course is also useful for some of the projects. A word of warning though: projects and development courses are largely independent, so some things that a student learns during the development courses are not used in the projects, some things that a student learns during the development courses are indeed used in the projects, but some things done in the projects are learned within the context of the project and not within the development courses. Moreover, the whole Informatica program is not based on the "workshop" philosophy but rather on building solid foundations preparing students for the future evolutions of computer science (see for instance the module description of Development 1).

2 Course program

In the following table you can see the program of the course, divided in lesson units. Each lesson unit is also associated with the corresponding book paragraphs. The last lesson unit of the course is reserved for a summary in preparation for the exam.
Note: Lesson units are intented as collections of topics and do not necessarily correspond to study weeks (for example, a lesson unit could span two study weeks).

Lesson unit	**Topics**	**Book chapters and paragraphs**
1	Introduction to algorithms Arrays Complexity of algorithms (empirical analysis, O notation)	3
2	Sorting algorithms - Insertion sort - Merge sort	2.1, 2.3
3	List Queue Stack Hash table	10.1, 10.2, 11.1 until 11.4
4	Trees - BST - k-d trees - 2-3 trees	12.1 until 12.3, 18
5	Graphs - undirected - directed - Dijkstra's shortest path	22.1 until 22.3, 24.3
6	Dynamic programming Floyd-Warshall	15.3, 25.2
7	Course review and preparation for the exam	

After each lesson unit, a multiple-choice questionnaire on the topics seen in class will be published. The questions are similar to those of the written exam.

3 Assessment

The course is tested with two exams: a practical assessment and a theoretical examination. The final grade is determined by the practical assessment. However, to pass the course, you **must** have a sufficient (i.e. ≥ 5.5) grade in the theoretical examination (as well as in the practical assessment).
The correspondence between learning goals and assessment parts is shown in the following table.

Learning goal	Theory	Practice
PERF	V	
DS^{A}	V	
$SORT^{I}$		V
$SORT^{A}$	V	
REC^{I}		V
REC^{A}	V	
$GRAPH^{I}$		V
$GRAPH^{A}$	V	

3.1 Theoretical examination

The theoretical examination consists of a **written exam** which covers the topics seen in class. The questions will be both theoretical and about code analysis. No help is allowed during the exam.

The exam consists of multiple-choice questions about:

- complexity of algorithms (i.e., "What is the complexity of the following code/algorithm?")
- inference of the behaviour of a given algorithm (i.e., "What does this algorithms do?", "Does this algorithm find the minimum element?", "What is the next step of this algorithm given the following state?", etc.)
- definitions given during the course

Each correct answer is worth 1 point. The grade of the written exam is computed as the percentage of correct answers in a scale from 0 to 10. For example, if the exam is made of 20 questions and you answer correctly 14 questions, then your grade is 14 * 10 / 20 = 7.

3.2 Practical assessment

The practical examination is a **practical assessment** during which the student is asked to implement (completely or in part) algorithms seen during the course or strictly connected to course topics. The practical assessment consists of a few assignments asking to fill in code of given partially-implemented algorithms. No help is allowed during the assessment.
The programming language used for the practical assessment will be C#. You need to install Visual Studio on your laptop before the practical assessment.
As preparation for the assessment, the students are strongly suggested to implement on their own every algorithm and data structure presented in the course (see also the "Homework" sections on the lesson slides).

3.3 Retake (herkansing)

If only one of the exam parts is passed (theory or practice), then only the other will need to be retaken.