Università degli Studi di L'Aquila - Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Course catalogue

Page compiled on 22/07/2019

Programme of Module "Lab. Of Algorithms And Data Structures"

- Code: F0132
- Type of course unit: Compulsory (Bachelor Degree in Computer Science curriculum General)
- Level of course unit: Undergraduate Degrees
- Semester: 1

Number of ects credits: (Bachelor Degree in Computer Science) 6 (workload 150 hours)

Teachers: Giovanna Melideo (Giovanna.Melideo@univaq.it)

1	**Course objectives**	This module invites students to study and implement the most important algorithms and data structures for information processing. The module focuses on techniques for the design and analysis of efficient algorithms, emphasizing methods useful in practice. Topics include lists, stacks, queues, trees, heaps and priority queues, binary search trees (including red-black), union-find for disjoint sets and graphs and evaluation of classic algorithms that use these structures for tasks such as sorting and searching. The module encourages students to develop implementations using the Java language, understand their performance characteristics, and estimate their potential effectiveness in applications using lectures and exercises.
2	**Course content and learning outcomes (dublin descriptors)**	Topics of the module include: • Java review. Interfaces and abstract classes. Generics. The Java Collection Framework Interfaces and classes. • Elementary data structures and their implementation in Java: lists, stacks and queues. Array and singly linked list implementations of lists, stacks and queues. Use of the standard classes from a client code perspective. • Searching and sorting algorithms and their implementation in Java: sequential and binary search; bubble-sort, insertion-sort, selection-sort, merge-sort, quick-sort. • Advanced data structures and their implementation in Java: binary trees, breadth-first search and depth-first search; binary search trees (BST), balanced BST, red-black trees; disjoint sets. • Priority queues and their implementation in Java: binary heaps and use of a heap to implement a priority queue; use of a priority queue to develop a sorting method. Java implementation of heap-sort. • Graphs: representations for both directed/undirected and weighted/unweighted models; Java implementation of breadth-first search and depth-first search; Java implementation of classical graph algorithms for solving the single-source shortest path problem and the minimum spanning tree problem. On successful completion of this module, the student should : • Know classical algorithms and data structures for information processing and their implementations, with particular focus on computational complexity aspects, and be aware of the effects of data organization and algorithms on program efficiency; Be familiar with standard techniques for designing algorithms, including the techniques of recursion, divide-and-conquer, and greedy and understand how apply them to design efficient algorithms for different kinds of problems. • Be able to apply their knowledge of data structures and algorithmic techniques to design efficient algorithms that correctly satisfy a given specification and write more efficient programs in Java; Be able to rigorously analyze the relative time and space efficiency of competing algorithms and carry out a comparative evaluation of merits in terms of efficiency and applicability of standard data structure. • Be able to identify efficient solutions to a given problem and choose appropriate data structures that effectively model the information in a problem; Judge efficiency tradeoffs among alternative data structure implementations or combinations. • Demonstrate the capability of formally presenting and modeling concrete problems and explain the most important characteristics concerning the standard data structures, their analyses, and their Java implementations. • Have an understanding of the role and characteristics of data structures and of the

		importance of time and space efficiency in designing algorithms and writing programs; To be able to recognize and look up variations of the material studied in the literature.
3	**Course prerequisites**	A basic understanding of programming in Java and a background in discrete mathematics are necessary prerequisites to this module.
4	**Teaching methodsand language**	Lectures and exercises will be held two times a week. The main goal is to present and discuss the content that is covered in this module, and illustrate the concepts using specific code examples. **Language**: English **Reference textbooks** • William J. Collins, ***Algoritmi e strutture dati in Java***. Ed. Maggioli, Apogeo Education. • C. Demetrescu, U. Petrillo, I. Finocchi, G. Italiano, ***Progetto di Algoritmi e Strutture Dati in Java***. McGraw-Hill.
5	**Assessment methods**	Pre-Assessment: There is no formal pre-assessment, but Course pre-requisites are clearly stated on the Module website. Formative Assessment: The formative assessment is performed via interaction between teacher and students during lectures. Students are involved in questioning and discussion, by means of open oral questions to the entire class. Summative Assessment: Written test followed by an optional oral exam. The oral exam can be required either by the student, to improve grades, or by the teacher, in presence of significant mistakes/misunderstandings in the written exam. The oral exam will occur within the same exam session of the written test and will typically cover the areas of the written answers that need clarification plus, possibly, additional subjects proposed by the teacher. An optional mid-term written test will also be provided, which is meant to cover the first part of the course, in order to help the students to split the workload. The written exam consists of a set of exercises to assess: (i) whether the student knows, can motivate and can compare the concepts, the implementation and the time and memory requirements of the data structures and algorithms; (ii) a student's general knowledge and understanding on the module material; (iii) a student's practical skills in selecting and applying suitable data structures and algorithms in software development and general problem solving. This involves writing program code with pen and paper. Criteria of evaluation will be the level of knowledge and understanding, and practical ability. The final marks of the Algorithms and Data Structures with Laboratory 12 CFU Course are roughly obtained as the average among the marks of the Algorithms and Data Structures and the Laboratory of Algorithms and Data Structures 6 CFU modules.