UNIVERSITÀ DEGLI STUDI DI PALERMO

DEPARTMENT	Matematica e Informatica
ACADEMIC YEAR	2019/2020
BACHELOR'S DEGREE (BSC)	COMPUTER SCIENCE
SUBJECT	ALGORITHMS AND DATA STRUCTURES
TYPE OF EDUCATIONAL ACTIVITY	B
AMBIT	50166-Discipline Informatiche
CODE	16670
SCIENTIFIC SECTOR(S)	INF/01
HEAD PROFESSOR(S)	GIANCARLO RAFFAELE Professore Ordinario Univ. di PALERMO
OTHER PROFESSOR(S)	
CREDITS	9
INDIVIDUAL STUDY (Hrs)	153
COURSE ACTIVITY (Hrs)	72
PROPAEDEUTICAL SUBJECTS	05880 - PROGRAMMING AND LABORATORY - INTEGRATED COURSE
MUTUALIZATION	
YEAR	2
TERM (SEMESTER)	2° semester
ATTENDANCE	Not mandatory
EVALUATION	Out of 30
TEACHER OFFICE HOURS	**GIANCARLO RAFFAELE** Monday 15:00 17:00 Stanza 106 Dipartimento di Matematica ed Informatica Thursday 15:00 17:00 Stanza 106 Dipartimento di Matematica ed Informatica

DOCENTE: Prof. RAFFAELE GIANCARLO

PREREQUISITES	Basic knowledge of programming and discrete mathematics, as specified by the sillabi of the courses propaedeutical to this one.
LEARNING OUTCOMES	Knowledge and Understanding Abilities Acquisition of the basic methods of the discipline for design and analysis of computer algorithms. Ability to use the specific language of the discipline Ability to apply knowledge and Understanding Ability to develop software based on efficient algorithms for basic problems. Judgment Autonomy To be able to evaluate the implications and the results of the study of algorithms and of the computational complexity associated to them. Communication Abilities Ability to convey the main results of algorithmic studies, even to a non-expert audience. Ability to put emphasis on the technological fall-out of algorithmic theory. Ability to Learn Ability to follow-up the advances of the field via the consultation of advanced texts and publications proper of the subject area. Ability to attend both master courses and LM courses.
ASSESSMENT METHODS	Oral Exam, Written Exam, Programming Exam The oral exam consists of a colloquium that, based on the two written exams, aims at evaluating the acquisition of the competences and disciplinary knowledge foreseen for this course; evaluation is in the interval 0-30. The questions of both the theory and programming written exam, both open or semi-structured, and designed to verify: a) the acquired knowledge; b) the elaborative abilities; c) the acquired ability to expose the themes of the course. The written exam, together with its discussion, will contribute 60% of the final vote. The programming exam, together with its discussion, will contribute 40% of the final vote. The final ranking is as follows: a) Basic knowledge of algorithmics , and limited ability to apply those techniques autonomously in new contexts, sufficient ability of exposition of the techniques and concepts associated to them (grade (18-21); b)Good knowledge of algorithmics, and good ability to apply those techniques autonomously in new contexts, good ability of exposition of the techniques and concepts associated to them (grade (22-25); c) Deep knowledge of algorithmics, and very good ability to apply those techniques autonomously in new contexts, very good ability of exposition of the techniques and concepts associated to them (grade (26-28); d) Excellent knowledge of algorithmics, and excellent ability to apply those techniques autonomously in new contexts, excellent ability of exposition of the techniques and concepts associated to them (grade (29-30L);
EDUCATIONAL OBJECTIVES	To expose the student to fundamental techniques for the design and analysis of algorithms, In particular, the entire spectrum of fundamental data structures and algorithmic paradigms will be covered, with notions of problem complexity and hardness. Fundamental engineering aspects for the efficient implementation of of algorithms will also be covered.
TEACHING METHODS	Classroom Lectures/ Laboratory programming
SUGGESTED BIBLIOGRAPHY	A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison Wesley. 1974-Libro di Testo Pincipale- Main Textbook Per i seguenti argomenti specifici, si utilizzeranno i seguenti testi (For the specific topics listed below, the textbook is as follows). T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein - Introduzione agli Algoritmi e strutture dati, McGraw Hill, 2010. Hashing C. Demetrescu, I. Finocchi, G.F. Italiano, Algoritmi e Strutture Dati, McGraw-Hill, 2008. Nozioni Introduttive. Tecniche Empiriche e Matematiche per l'Analisi di Algoritmi.

	Algoritmi di Ordinamento (Introductory Notions. Mathematical and Empirical Techniques for the Analysis of Algorithm. Sorting Algorithms). H. Horowitz, S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, 1984. Paradigma Greedy (Greedy Paradigm). Materiale distribuito dal docente (Handouts from the instructor). Tutta la parte di Lezioni di Programmazione Algoritmica. (All part regarding Algorithmic Programming).

SYLLABUS

Hrs	Frontal teaching
2	Introductory Notions Algorithms and Data Structures. Basic intuitive notions for the algorithmic solutions of computational problems, with different solutions for the same problem. Algorithmic efficiency.
6	Mathematical and Empirical Techniques for the Analysis of Algorithm. Analysis of algorithms. Growth rate of mathematical functions. Fundamental Recurrences. Solutions of fundamental recurrences. Iterative Method, Master Theorem.
4	Computational Models, Computational Complexity of Problems and Algorithms. Random Access Machines. RAM Computational Complexity. Turing Machines. Relation between RAM and Turing Machines (outline).
3	Sorting Algorithms Lower Bounds for Sorting Algorithms: worst case and average case. Fundamental sorting algorithms.
5	Algorithmic Paradigms for The Design of Efficient Algorithms: Divide and Conquer. Examples: MinMax; Integer Multiplication; Mergesort; Quicksort. Worst case and Average Case Analysis.
5	Algorithmic Paradigms for The Design of Efficient Algorithms: Dynamic Programming and Greedy Techniques. Examples: Dynamic Programming. Greedy Techniques. Matrix Chain Product. Longest Common Subsequence. Membership test for Context Free Languages. Greedy Algorithms: Optimal Storage on Tape, Bin Packing.
5	Advanced Data Structures and Set Operations. Fundamental Operations on Sets. Hash Tables. Union-Find.
5	ADVANCED DATA STRUCTURES: TREES Optimum binary search trees. Dynamic data structures: Dictionaries, Queues, Heaps.
5	Graph Algorithms: Representations and Connectivity. Representations of graphs. Visits. Biconnectivity and Strong Connectivity.
5	Graph Algorithms: Optimization Problems. Algorithms for the Determination of a Minimum Spanning Tree. Algorithms for shortest Paths.
3	NP-Completeness Theory. on-Deterministic Turing Machines. Outline of the P NP and NP-C classes.
1	Lecture in Algorithmic Programming.. Elementary data Structures in C (summary from the Programming Course). Arrays, Linked Lists, strings and their implementation in C.
3	Lectures in Algorithmic Programming. Abstract Data Structures in C. Queues and Stacks and their implementation in C via arrays and linked lists. Evaluation of an arithmetic expression in postfix form: C implementation.
5	Lectures in Algorithmic Programming. Sorting Algorithms in C based on Comparisons. Implementation examples: selectionsort; insertionsort; bubblesort; Mergesort, Heapsort, Quicksort.
4	Lectures in Algorithmic Programming. Sorting Algorithms in C: The special case of integers. Implementation examples: Bucketsort, Lex Sort.
6	Lectures in Algorithmic Programming. Algorithmic Paradigms in C. Recursion. Divide and Conquer: MinMax, Binary Search in C. Dynamic Programming: Chain Matrix Multiplication.
5	Lectures in Algorithmic Programming. Data Structures for the representation of trees and graphs in C. Tree Visits in C. DFS and BFS on a graph in C.