



DISCIPLINE ACADEMIC SHEET
ACADEMIC YEAR 2025 – 2026

1. PROGRAMME DATA

1.1. Higher Education Institution	UNIVERSITY OF CRAIOVA
1.2. Faculty	Automation, Computers and Electronics
1.3. Department	Computers and Information Technology
1.4. Field of Study	Computers and Information Technology
1.5. Study level	L(License / Undergraduate)
1.6. Organizational form	IF
1.7. Study Program (name/code) /Qualification	Computers / L2060101010

2. DISCIPLINE DATA

2.1 Discipline Name	Algorithms Design						
2.2 Course Activities Holder	Bădică Costin						
2.3. Practical Activities Holder							
2.4. Study Year	1	2.5. Semester	2	2.6. Evaluation type	E	2.7. Discipline regime	DOB

3. ESTIMATED TOTAL TIME (hours per semester of teaching activities)

3.1. Number of hours per week	5	in which: 3.2 course	2	3.3. sem./lab./pr.	3
3.4. Total hours of curriculum	70	in which: 3.5 course	28	3.6. sem./lab./pr.	42
3.7. Time distribution - hours/week					hours
Study after manual, course support, bibliography and notes					20
Additional documentation in library, on specialized electronic platforms and field					10
Training seminars / labs, homework, portfolios and essays					15
Tutorial					5
Examinations					5
Other activities.....					-
Total hours per individual activities					55
3.8. Total hours per semester					125
3.9. Number of credits					5

4. PRECONDITIONS (where appropriate)

4.1. of curriculum	• Computer Programming and Programming Languages, Mathematical Analysis
4.2. of competence	• n/a

5. CONDITIONS (where appropriate)

5.1. of the course	• The course is taught using interactive face-to-face (f2f) and/or online lectures, distribution of course materials in
--------------------	---

	digital format and interaction with the students. The teaching process is structured as follows: - 70% theoretical presentations, using the course support (slides), as well as examples of theoretical exercises and practical demonstrations and tutorials.
5.2. of seminar/ laboratory/project	•

6. COURSE OBJECTIVES – expected learning outcomes that are supported by completing and passing the course

Knowledge	The student/graduate: 1. Identifies and describes basic concepts, principles and methods from computer science. 2. Explains and interprets theoretical and experimental results from computer science. 3. Describes, identifies, summarizes elementary concepts and methods regarding programming languages, programming environments, programming techniques and the way they are applied to concrete problems. 1. Will know fundamental algorithms, data structures, graph theory and dynamic programming in order to efficiently implement algorithmic solutions to well-defined problems.
Skills (Abilities)	The student/graduate selects and explains concepts specific to algorithm design, object-oriented programming, logic and functional programming. The student/graduate specifies requirements, analyzes, elaborates, develops and tests software systems in general-purpose (C etc.) and/or object-oriented (C++, Java etc.) programming languages, applying the elements specific to software engineering. The student/graduate designs, implements, develops and/or manages applications that include various types of databases. The student understands the main concepts underlying translation systems and source code compilation, and contributes to the development and improvement of tools specific to programming languages.
Responsibility and autonomy	The student/graduate behaves honorably, responsibly and ethically, in the spirit of the law, to ensure the reputation of the profession.

7. CONTENT

7.1. Course	The method of implementation	Teaching methods	Allocated time (hours)
1. Introduction to algorithms. Analysis of algorithm complexity. Asymptotic notation.	face to face	Lectures. Theoretical examples. Practical demonstrations. Interaction through questions and answers.	2
2. Analysis of algorithms using recurrences. The Master theorem.			2
3. Correctness and testing of algorithms. Induction and invariants.			2
5. Divide-and-conquer algorithms. Merge sort. The stockbroker problem. The convex hull.			2
25. Quicksort. Randomized algorithms. Selection.			2

Counting sort. The universe of sorting algorithms. The lower bound of comparison-based sorting.			
6. Abstract data types. Lists. Modular programming.			2
7. Stacks and queues. Dynamic memory allocation.			2
8. Graphs and trees: basic elements. Representation and traversal of trees.			2
9. Representation and traversal of graphs. Depth-first traversal and level-order traversal. Topological sorting.			2
10. Dynamic programming. Optimal matrix chain multiplication. The longest common subsequence problem. The domino problem.			2
11. Exhaustive search and backtracking. Combinatorial algorithms. The knapsack problem - the discrete version. The queens problem.			2
12. Greedy algorithms. Activity scheduling (or Activity planning). The knapsack problem - the continuous version. Graph coloring.			2
13. Introduction to NP-completeness. Classes of problems. NP-hard and NP-complete problems. Polynomial reduction. Classic NP-complete problems.			2
14. Evaluation and make-up laboratories			2
Bibliography:			
1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein, Introduction to Algorithms (4th edition), MIT Press and McGraw-Hill, 2022 2. Steven S. Skiena, The Algorithm Design Manual (2nd ed.), Springer, 2011 3. Antti Laaksonen, Guide to Competitive Programming. Learning and Improving Algorithms Through Contests. Springer, 2017 4. Steven S. Skiena, Miguel A. Revilla, Programming Challenges: The Programming Contest Training Manual, Springer, 2003			

7.2. Seminar/laboratory/project	The method of implementation	Teaching methods	Allocated time (hours)
1. Exercises covering the correctness of algorithms and the analysis of algorithm complexity.	face to face	Tutorials. Solved problems. Interaction through questions and answers.	2
2. Exercises with divide-and-conquer algorithms.			2
3. Exercises with algorithms for lists, stacks and queues.			2
4. Exercises with algorithms on graphs and trees.			2
5. Exercises with dynamic programming algorithms.			2
6. Exercises with combinatorial and backtracking algorithms.			2
7. Exercises with greedy algorithms.			2
Laboratory			
1. Introduction to the laboratory working procedure. Experiments with insertion sort.	face to face	Tutorials. Practical demonstrations and hands-on experience. Question-and-answer based interaction. Competitive programming methods, environments and practical experience.	2
2. Binary search. Merging. Merge sort. Selection sort.			2
3. Testing algorithms. Implementing algorithms in competitive programming environments.			2
4. Implementing divide-and-conquer algorithms.			2
5. Quicksort and selection. Randomized algorithms.			2
6. Implementing lists. Algorithms using stacks and queues.			2
7. Representation and traversal of trees.			2
8. Graph algorithms.			2
9. Dynamic programming algorithms.			2
10. Combinatorial algorithms.			2
11. Backtracking algorithms.			2
12. Greedy algorithms.			2
13. Evaluation and make-up sessions.			2
14. Evaluation and make-up sessions.			2
Bibliography:			

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein, Introduction to Algorithms (4th edition), MIT Press and McGraw-Hill, 2022
2. Steven S. Skiena, The Algorithm Design Manual (2nd ed.), Springer, 2011
3. Antti Laaksonen, Guide to Competitive Programming. Learning and Improving Algorithms Through Contests. Springer, 2017
4. Steven S. Skiena, Miguel A. Revilla, Programming Challenges: The Programming Contest Training Manual, Springer, 2003

8. Correlating the content of the discipline with the expectations of representatives from the epistemic community, professional associations and representative employers in the field related to the program

The content of the lectures and applications is aligned with that of similar courses at other national and international universities. The learning outcomes and acquired competences correspond to the requirements of major employers. These have been continuously developed and improved based on the input received from the representatives of local IT companies.

9. EVALUATION

Activity Type	9.1. Evaluation criteria	9.2. Evaluation methods	9.3. Final mark weight
9.4. Course	Solving exercises and problems involving algorithm design and analysis methods and techniques.	Written exam.	70 %
9.5. Seminar/laboratory/project	Practical results obtained during the laboratory sessions. Practical experience in algorithm design and analysis during the seminar sessions.	The acquired practical knowledge is continuously evaluated during the laboratory and seminar sessions, by assessing the practical results and the answers to questions.	30 %
9.6. Minimum standard of performance			
Minimum grade 5.			

Date of Completion	Signature of the Course Activities Holder,	Signature of the Practical Activities Holder,
23.08.2025	Bădică Costin	

Date of Approval in the Department	Signature of the Department Director,
25.08.2025	Ș.l.dr.ing. Nicolae ENESCU

Date of Approval in the Faculty	Signature of the Dean,
	Prof.dr.ing. Dorin ȘENDRESCU