COURSE DESCRIPTION CARD - SYLLABUS

Course name
Algorithms and data structures [S1Inf1>AiSD]

Course

Field of study
Computing

Year/Semester
1/2

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
Polish

Form of study
full-time

Requirements
elective

Number of hours

Lecture
30

Laboratory classes
30

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

5,00

Coordinators

dr hab. inż. Aleksandra Świercz
aleksandra.swiercz@put.poznan.pl

prof. dr hab. inż. Marta Szachniuk
marta.szachniuk@put.poznan.pl

Lecturers

Prerequisites

A student starting this research topic should have a basic knowledge of the implementation of programs in C / C ++. It should also be able to find solutions for basic programming problems as well as to test and fix bugs in the programs he implemented himself. Additionally, the student should be able to obtain information from the indicated sources. He should also understand the necessity of expanding his competences in the domain of research. In terms of social competences, the student must present such attitudes as honesty, responsibility, perseverance, cognitive curiosity, creativity, personal culture and respect for other people.

Course objective

1. Provide students with basic knowledge of computational complexity in the field of analysis, operation of deterministic and nondeterministic Turing machine, RAM machine, classification of problems and algorithms as well as complexity classes P and NP. 2. Provide students with basic knowledge of algorithms in the field of sorting data strings with different computational complexity, greedy and dynamic programming, backtracking and basic graph algorithms such as BFS, DFS, finding the Euler and Hamilton cycle. 3. Provide students with basic knowledge of data structures, including the way trees, BST trees, heaps and graphs work, and the analysis of their complexity. 4. Developing students" skills to prove NP-completeness of problems. 5. Developing student"s ability of programming implementation of the known algorithms and data structures. 6. Developing students" skills to select the appropriate algorithm and data structure for the problem to be solved and to evaluate the computational and memory complexity of their implementation. 7. Developing students" skills in testing implemented algorithms and their evaluation.

Course-related learning outcomes

Knowledge:
Has extended and in-depth knowledge of mathematics useful for formulating and solving complex computer science tasks related to analysis and formal proofs of correctness and computational complexity of algorithms.
Has a structured, theoretically based general knowledge of algorithms and complexity. Has detailed knowledge of algorithmics, data structures, and computational and memory complexity analysis.
Knows the basic methods, techniques and tools used to solve simple IT tasks in the field of computational complexity analysis of algorithms and problems.

Skills:
Can plan and carry out experiments, including the measurement of algorithm operation time, interpret the obtained results and draw conclusions about the correctness of the selection and complexity of the algorithms.
Has the ability to formulate algorithms and program them using at least one of the basic high-level programming languages.

Social competences:
Is able to properly define priorities for the implementation of a task defined by himself or others by resolving the dilemma of whether the implementation of more efficient algorithms is worth the increased effort of their implementation.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Learning outcomes presented above are verified as follows:
Forming grade:
a) in the field of lectures, verification of the assumed learning outcomes is carried out by:
- assessment of the two tasks given to students during lectures: one concerns the implementation and analysis of the graph structure, the other concerns the analysis of computational complexity,
- rewarding student activity during lectures;
b) in the field of laboratory classes, verification of the assumed learning outcomes is carried out by:
- evaluation of reports with the results of projects aimed at the implementation and analysis of algorithms and data structures,
- evaluation of programming projects with implementation of algorithms using various data structures,
- assessment of the solutions of the tasks that demonstrate the operation of the algorithms presented by the students on the blackboard.
Summative grade:
Checking the assumed learning outcomes is carried out by:
- evaluation of reports with the results of projects aimed at the implementation and analysis of algorithms and data structures,
- evaluation of programming projects with implementation of algorithms using various data structures,
- assessment of knowledge and skills related to the implementation of laboratory tasks through 2 tests per semester,
- assessment of the knowledge and skills demonstrated in the problem-based written exam:
a. in the form of 4 closed tasks, which consist in entering the results of calculations and analyzes in the

empty spaces, checking the student"s skills in solving algorithmic problems,
b. 1 additional open task to check the ability to prove that algorithms belong to the NP class,
c. tasks are scored on a scale of 0-5 points, in increments of 0.25 points; 50% of the points are required to pass the exam.
Activity during classes is rewarded with additional points, which are taken into student"s individual account during the evaluation of his work in the semester.

Programme content

Lectures on the subject cover basic topics in the field of algorithmics:
* basic terms: problem and algorithm, data and data operations, instance
* correctness of algorithms, verification
* division of problems into decision-making and optimization problems
* deterministic and non-deterministic Turing machine and RAM machine, as examples of an abstract computer model used to execute
algorithms.
* definition of classes of decision problems P and NP, along with subclasses of NP-complete and strongly NP-complete problems and presented methods
proving that problems belong to these classes
* computational complexity of problems and time and memory complexity of algorithms along with methods of determining it and writing it down in O() notation
* methods of constructing algorithms such as top-down method, divide and conquer and reverse search
* comparison of the greedy method and dynamic programming with a discussion of pseudopolynomial complexity on the example of the knapsack problem
* computer representations of graphs, including a matrix and a list of incidents, a list of successors and a graph matrix

algorithm, data and data operations, instance, the concept of type. The topics of the correctness of algorithms, its definition and verification are discussed. The division of problems into decision-making and optimizating is presented, along with the characteristics of these two classes and examples of problems that belong to them. Before discussing the implementation of algorithms in modern programming languages, the deterministic and nondeterministic Turing machine and the RAM machine are discussed as examples of an abstract computer model used to execute algorithms. On the basis of this material, the idea and definition of the classes of decision problems P and NP are explained, along with the subclasses of NP-complete and strongly NP-complete problems, and the methods of proving that problems belong to these classes are presented. The computational complexity of problems as well as the time and memory complexity of algorithms are discussed, along with the methods of its determination and recording in the O () notation. The worst-case and best-case complexity and average complexity are discussed. During the lecture, general methods of constructing algorithms are presented in detail, such as the top-down method, divide and conquer and recurrence search. A comparison of the greedy method and dynamic programming is also presented, along with a discussion of the pseudo-polynomial complexity. For this purpose, a detailed analysis of the knapsack problem is used. The lecture also presents possible methods of computer representation of graphs, including the matrix and the list of incidents, the list of successors and the graph matrix along with a detailed analysis of their time and memory complexity depending on the number of vertices and edges in the graph and the performed operations.
Laboratory classes put a lot of emphasis on the practical application of algorithms and data structures presented at the lecture through the implementation of projects and solving tasks on the blackboard. The classes are divided into several thematic groups, each of which ends with the implementation of a project implementing the discussed algorithms. The first thematic group presents sorting algorithms ranging from the simplest, working with square complexity, such as bubble sort, through selection and insertion, through faster QuickSort sort, through merge and Shell, to linear time sort using the bucket algorithm and through counting. For each algorithm, its complexity is analyzed at best, medium, and worst case. Based on the sorting algorithms, the concept of recursion is also demonstrated. Another topic group covers complex data structures such as mono and bi-directional lists, trees including BST trees, and heaps. For each structure, an algorithm for adding and removing elements from them is presented, as well as possible ways of searching them. Their complexity is also analyzed as well as the problems in which they should be used. The third thematic group are graph algorithms including algorithms for directed and undirected graphs, such as BFS, DFS, topological sorting, spanning trees and

Euler and Hamilton cycle search, also presenting algorithms with conversion. Graphs, the subject of the implementation of graph representations presented in the lecture is discussed in detail. The last thematic group covers the implementation of the greedy and dynamic algorithm for the knapsack problem, their comparison and analysis.

Course topics

The lectures begin with the explanation of basic terms in the field of algorithmics, such as problem and algorithm, data and data operations, instance, the concept of type. The topics of the correctness of algorithms, its definition and verification are discussed. The division of problems into decision-making and optimizating is presented, along with the characteristics of these two classes and examples of problems that belong to them. Before discussing the implementation of algorithms in modern programming languages, the deterministic and nondeterministic Turing machine and the RAM machine are discussed as examples of an abstract computer model used to execute algorithms. On the basis of this material, the idea and definition of the classes of decision problems P and NP are explained, along with the subclasses of NP-complete and strongly NP-complete problems, and the methods of proving that problems belong to these classes are presented. The computational complexity of problems as well as the time and memory complexity of algorithms are discussed, along with the methods of its determination and recording in the O () notation. The worst-case and best-case complexity and average complexity are discussed. During the lecture, general methods of constructing algorithms are presented in detail, such as the top-down method, divide and conquer and recurrence search. A comparison of the greedy method and dynamic programming is also presented, along with a discussion of the pseudo-polynomial complexity. For this purpose, a detailed analysis of the knapsack problem is used. The lecture also presents possible methods of computer representation of graphs, including the matrix and the list of incidents, the list of successors and the graph matrix along with a detailed analysis of their time and memory complexity depending on the number of vertices and edges in the graph and the performed operations.
Laboratory classes put a lot of emphasis on the practical application of algorithms and data structures presented at the lecture through the implementation of projects and solving tasks on the blackboard. The classes are divided into several thematic groups, each of which ends with the implementation of a project implementing the discussed algorithms. The first thematic group presents sorting algorithms ranging from the simplest, working with square complexity, such as bubble sort, through selection and insertion, through faster QuickSort sort, through merge and Shell, to linear time sort using the bucket algorithm and through counting. For each algorithm, its complexity is analyzed at best, medium, and worst case. Based on the sorting algorithms, the concept of recursion is also demonstrated. Another topic group covers complex data structures such as mono and bi-directional lists, trees including BST trees, and heaps. For each structure, an algorithm for adding and removing elements from them is presented, as well as possible ways of searching them. Their complexity is also analyzed as well as the problems in which they should be used. The third thematic group are graph algorithms including algorithms for directed and undirected graphs, such as BFS, DFS, topological sorting, spanning trees and Euler and Hamilton cycle search, also presenting algorithms with conversion. Graphs, the subject of the implementation of graph representations presented in the lecture is discussed in detail. The last thematic group covers the implementation of the greedy and dynamic algorithm for the knapsack problem, their comparison and analysis.

Teaching methods

Lecture: multimedia presentation, illustrated with examples given on the board.
Laboratory classes: presentation illustrated with examples given on the blackboard and carrying out the tasks given by the teacher - practical exercises.

Bibliography

Basic
1. Elementy analizy algorytmów, L. Banachowski, A. Kreczmar, WNT, W-wa, 1982
2. Algorithms + Data Structures = Programs, N. Wirth, 2004
(http://www.ethoberon.ethz.ch/WirthPubl/AD.pdf)
3. Złożoność obliczeniowa problemów kombinatorycznych, J. Błażewicz, WNT, W-wa, 1988
4. Introduction to Algorithms, T.H. Cormen, Ch.E.Leiserson, R.L. Rivest, C. Stein, The MIT Press
(https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf)
Additional

1. Algorytmika praktyczna nie tylko dla mistrzów, P. Stańczyk, PWN, 2009

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,00
Classes requiring direct contact with the teacher	62	2,50
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	63	2,50