COURSE DESCRIPTION CARD - SYLLABUS

Course name
Algorithmics in Practice [S1Inf1>ALG]

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 (e.g. online) 0
Tutorials 0	Projects/seminars 0	

Number of credit points

5,00

Coordinators	Lecturers
mgr inż. Artur Laskowski artur.laskowski@put.poznan.pl	mgr inż. Artur Laskowski artur.laskowski@put.poznan.pl

Prerequisites

A student starting this subject should have basic knowledge of implementation algorithms in C, C++ or Pascal and analysis of their complexity. Should have the ability to solve basic programming problems using data sorting techniques, heaps, greedy methods and programming dynamic. Additionally, the student should be familiar with the concept of team programming in accordance with the ACM ICPC formula and have the ability to read and understand simple texts English. The student should have the ability to obtain information from the indicated ones sources. He should also understand the need to expand his competences. In terms of social competences, the student must demonstrate attitudes such as honesty, responsibility, perseverance, ambition and cognitive curiosity, creativity, personal culture, respect for other people.

Course objective

1. Providing basic knowledge about computational complexity in terms of its analysis and operation deterministic and non-deterministic Turing machine, RAM machine, problem classification and algorithms, complexity classes P and NP, data structures including how trees work, BST trees, AVL, interval trees, graphs, including bipartite graphs, and analysis of their complexity and knowledge in algorithmics in the field of dynamic memory management, tree search algorithms and graphs, finding shortest paths, maximum flow in graphs, determining trees spanning, computational geometry and text algorithms, and advanced techniques programming to improve the process of implementing algorithms and optimizing them efficiency. 2. Developing the ability to prove NP-completeness of problems, programming implementation learned algorithms and data structures, selecting the appropriate algorithm and data structure for the problem being solved and the assessment of the computational and memory complexity of their implementation, team programming in accordance with the ACM ICPC formula, the use of external libraries with ready-made implementations of algorithms and data structures. 3. Preparing students to take part in team programming competitions.

Course-related learning outcomes

Knowledge:
Has structured, theoretically based general knowledge of algorithms and complexity.
(K1st_W4)
Knows the basic methods, techniques and tools used to solve IT tasks
scope of analysis of the computational complexity of algorithms and problems. (K1st_W7)

Skills:
Is able to plan and conduct experiments, including measuring the running time of algorithms,
interpret the obtained results and draw conclusions about the correctness of the selection and complexity of the algorithms.
(K1st_U3)
Is able to use analytical methods to formulate and solve IT tasks
experimental in order to select appropriate algorithms and data structures. (K1st_U4)
Is able to assess the computational complexity of elementary algorithms and problems. (K1st_U8)
Is able to obtain information about algorithms and data structures and how to use them in
specific problems from literature and the Internet (in the native language and English), integrate them,
interpret and critically evaluate them, draw conclusions and formulate them comprehensively
justify opinions. (K1st_U9)
Has the ability to formulate algorithms and program them using C++ and
use the advanced capabilities of this language. (K1st_U11)
Able to organize, cooperate and work in a group in order to develop and implement effective solutions
version of algorithms, assuming different roles in it, and is able to appropriately determine the priorities used
carrying out a task specified by yourself or others. (K1st_U18)

Social competence:
Understands based on the history of C++ standards ('11, '14, '17) that in computer science knowledge and skills
they become obsolete very quickly. (K1st_K1)
Is aware of the importance of knowledge in solving problems that require designing new ones
algorithms and knows examples and understands the causes of incorrectly implemented algorithms
led to serious financial losses. (K1st_K2)

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Formative assessment:
a) in the field of lectures, verification of the assumed learning outcomes is carried out by:
- monitoring and rewarding students' own work involving independent solving
algorithmic tasks
- rewarding students' activity during lectures
b) in the field of laboratories, verification of the assumed learning outcomes is carried out by:
- evaluation of solutions to tasks demonstrating how the algorithms presented by

students on the blackboard
- monitoring the student's progress in solving algorithmic problems during classes
laboratory and outside them, which are assessed by an automatic checking system
Summary rating:
a) in the field of lectures, verification of the assumed learning outcomes is carried out by:
- two colloquia requiring solving 4-10 practical problems in the field of algorithms,
- tasks are scored on a scale of 0-5 points (maximum number of points depends on the grade
difficulty), in increments of 0.5 points,
- points for tests are converted into percentages by dividing by the maximum number of points
to be obtained are included with a weight of 0.3 in the final grade of the subject
b) in the field of laboratories, verification of the assumed learning outcomes is carried out by:
- analysis of the results of tasks completed during the semester - approximately 30 compulsory tasks,
including:
3.0 you have to solve 50% of the tasks, for 3.5 60% of the tasks, etc. - the grade is included in the final
grade of the course
with a weight of 0.5
Additional assessment elements:
- for solving additional tasks selected by the instructor, the student can receive up to 1.5%
extra points for the task

Programme content

Program content included in this subject in addition to presenting algorithmic content
required of a graduate of first-cycle studies contain a wide range of topics that he has
prepare students to take part in algorithmic team programming competitions
compliant with the ACM ICPC formula. It is assumed that this subject has been chosen as an elective
will be mainly by students interested in this type of professions, who already have certain skills
basic algorithmic skills and knowledge. Therefore, many aspects are discussed at a high level
intermediate and focused on their practical use in solving problems
in algorithmic competitions.
The lecture is divided into two main parts. The first part is more theoretical and concerning
computational complexity. The division of problems into decision-making and optimization problems is
presented,
along with the characteristics of these classes and examples of problems belonging to them. Before starting
discussing the implementation of algorithms in modern programming languages is discussed
deterministic and non-deterministic Turing machine and RAM machine, as examples
an abstract model of a computer used to execute algorithms. Based on this material
the idea and definition of classes of decision-making problems P and NP are explained, along with
subclasses of problems
NP-complete and strongly NP-complete and presented methods of proving the affiliation of problems
to these classes. The second part is related to the presentation of specific algorithms and their applications
solving practical algorithmic problems (hence the name of the course). This part begins
from a discussion of dynamic data structures with a detailed discussion of how they work and
use of pointers and dynamic memory allocation in C++. They remain moved
problems of memory leaks, incorrect pointers, incorrect pointer arithmetic
(buffer overflow) and nested pointers (pointer to pointer, etc.).
Using dynamic memory allocation, an implementation of lists one and two is presented
directions and trees. The problem of balancing a tree is discussed and trees are presented
AVL, as an example of sustainable trees. The final tree topic is trees
compartmental, as an example of an effective data structure for storing data information
represented by ranges of numbers on the integer line. For each data structure
the algorithm for adding and removing elements and possible methods are presented
search. Next are graphs and the algorithms that use them. Their discussion begins with
presenting possible graph representations (incident matrix, adjacency list, incident list) and
their selection to the problem being solved.
The BFS and DFS search algorithms and their application to sorting are further discussed
topological, determines strongly coherent and bicoherent components, bridges and points of articulation.
The next ones are algorithms for finding the shortest paths using Floyd-Warshal algorithms,
Bellman-Ford, Dijsktra and Johnson,minimal spanning trees using Prima and algorithms

Kruskal and maximum flow using the Ford-Fulkerson and Dinic algorithm. Algorithms
flows are used to present the solution to the maximum matching problem in
bipartite graph. Then, the application of dynamic programming algorithms is discussed
to problems whose computer representation uses trees or graphs. There's another one
computational geometry and problems such as vector calculus, intersection of segments,
finding pairs of nearest and farthest points and convex hull. Overview of algorithms
text includes pattern search algorithms in one and two dimensions, finding
cycles in words and finding maximum palindromes. The topic of effectiveness is discussed
use of existing libraries with implementations of algorithms and data structures based on
STL libraries and advanced C++ language capabilities based on bit operations. The last one
the topic is the construction of algorithms with reversion and their optimization using the method
division and limitations.
Laboratory classes emphasize the practical application of algorithms and data structures
presented at the lecture by solving and implementing algorithmic problems on
computer in C++. In order to be well prepared to take part in programming competitions
and teaching how to solve specific practical problems, tasks and are used
problems illustrating the presented content, largely coming from historical competitions
and sets of tasks. About 10 tasks are assigned to each topic during the lecture
three different and announced levels of difficulty, illustrating the presented content.
The students' task is to solve as many problems as possible during laboratory classes,
when consultation and assistance from the teacher is possible, and then they can complete the work at home.
The beginning of the next classes serves to dispel doubts and explain solutions to these problems. Down
An automatic checking system is used to grade tasks and is completed within a minute
able to assess the correctness of the solution. Thanks to this, the student can immediately, without wasting time
start the process of finding the error and correcting the incorrect solution. Learning to solve
algorithmic problems on the computer is completed by solving tasks
presenting how algorithms work on the board. Part of the laboratory classes are intended
for students to solve problems in teams according to the ACM ICPC formula in order to obtain
experience working in a team on a set of problems. This is done in a mutual situation
competition between several groups of students in the same way as in real competitions.
Some of the above-mentioned program content is implemented as part of the student's own work.

Teaching methods

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

Bibliography

Basic:
1. Wprowadzenie do algorytmów, T.H. Cormen, Ch.E.Leiserson, R.L. Rivest, C. Stein, PWN, W -wa, 2012
2. Algorytmika praktyczna nie tylko dla mistrzów, P. Stańczyk, PWN, 2009
3. Kombinatoryka dla programistów, W. Lipski, WNT, 2007
4. Tablice matematyczne, W. Mizerski, Adamantan, 2008
5. Złożoność obliczeniowa problemów kombinatorycznych, J. Błażewicz, WNT, W-wa, 1988

Additional:
1. The CRC Concise Encyclopedia of Mathematics, E. W. Weisstein, CRC Press, 1998
2. C++ Biblioteka standardowa, Podręcznik programisty, N. M. Josuttis, Helion, 2003

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