Computability and Complexity

This course consists of two parts: the computability part and the complexity part.

In the first part of the course we will discuss what it means for the function to be computable and for the problem to be decidable. We will consider examples of uncomputable functions and undecidable problems and study the properties of computability and decidability. In particular, we will discuss the following well known programming problem: write a program that prints its own text. We will see that this problem is a simple formulation of an important theoretical result and its solution corresponds to mathematical technique behind the first part of the course.

In the second part of the course we will discuss computations with bounded resources. We will consider two main types of resources: time and space. We will discuss complexity classes P and NP and relation between them. We will study the notion of NP-completeness and will practice in proving NP-completeness of various problems. We will discuss the complexity class PSPACE and the notion of PSPACE-competeness.

Both parts of the course can be taken separately, see prerequisites.

# Course Objective

The goal of the course is to provide an introduction to the theory of computation. It can serve as a theoretical basis for students interested in more practical areas of Computer Science as well as a starting course for further studies in Theoretical Computer Science.

# The position of the course in the structure of the educational program

The course has 40 academic hours. Half of them will be lectures and half – seminars. There will be homework assignments after each seminar, midterm written exam after the first part of the course and the final written exam after the second part of the course.

## Prerequisites of the course:

Basic calculus is required. Basic knowledge of graph theory might be helpful, but is not necessary. Basic programming experience would help to understand the intuition behind the material of the course, but is also not necessary.

In case the first part of the course is taken separately the prerequisites are the same.

In case the second part of the course is taken separately, basic knowledge on Turing machines is required additionally, including the universal Turing machine.

# Topic-Wise Curricula Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Topic name | Course hours, total | Audience hours | |
| Lectures | Practical studies |
|  | **Part 1: Computability (20 hrs.)** |  |  |  |
|  | Informal introduction to algorithms. Adaptive and non-adaptive algorithms. Binary search. Lower bounds: counting argument, adversary method. |  | 0 | 2 |
|  | Computable functions, decidable sets, enumerable sets. Post's theorem. Diagonal argument. Uncomputable functions and undecidable sets.  Universal computable function. Halting problem. |  | 2 | 2 |
|  | Godel universal function. Rice-Uspensky's theorem. |  | 2 | 2 |
|  | Fixed-point theorem. Program that prints its own text. $m$-reducibility. $m$-completeness. |  | 2 | 2 |
|  | Turing reducibility. Relativization. Turing machines. Church-Turing thesis. |  | 2 | 2 |
|  | Universal Turing machine. Undecidability of the word problem for finitely presented semi-groups. |  | 2 | 0 |
|  | **Part 1, totally:** | 20 | 10 | 10 |
|  | **Part 2 (20 hrs.)** |  |  |  |
|  | Time and space resources for Turing machines. Complexity class P. Time hierarchy theorem. Space hierarchy theorem. |  | 2 | 2 |
|  | Boolean circuits and formulae. Circuit complexity. Examples, upper and lower bounds. Counting argument. Complexity class P/poly. Relation between P and P/poly. |  | 2 | 2 |
|  | Complexity class NP. Reductions. NP-hard and NP-complete problems. SAT problem is NP-complete. |  | 2 | 2 |
|  | Other examples of NP-complete problems. Complexity class co-NP, co-NP-completeness. Relations between P, NP and co-NP. |  | 2 | 2 |
|  | Space complexity. Complexity class PSPACE. Savitch's theorem. TQBF is PSPACE-complete. Characterization of PSPACE in terms of games. |  | 2 | 2 |
|  | **Part 2, totally:** | 20 | 10 | 10 |
|  | **TOTAL:** | 40 | 20 | 20 |

# Bibliography

1. A.Shen, N.K. Vereshchagin, Computable functions, Providence, R.I.: Amer. Math. Soc., 2003.
2. Sanjeev Arora, Boaz Barak, Computational Complexity: A Modern Approach, 594 pages, Cambridge University Press; 1 edition (April 20, 2009).
3. M. Sipser, Introduction to the theory of computation, 2-nd ed., International edition.-N.-Y.: Thomson Course Technology, 2006.