Government of Russian Federation

Federal State Autonomous Educational Institution of High Professional Education

“National Research University Higher School of Economics”

Syllabus for the course “Theory of computation”

bachelor program in 01.03.02.62 “Applied Mathematics and Information Science”

Authors:
Vladimir Podolskii, assistant professor, vpodolskii@hse.ru
Bruno Bauwens, assistant professor, brbauwens@gmail.com

Moscow - 2018

1. Course description

a) Title: theory of computation.
b) Pre-requisites:
- Discrete mathematics, linear algebra, probability theory.
- Basic English language, both oral and written.
c) Course type: elective, for 3th and 4th year Bach.
d) Abstract:

This course teaches a mathematical theory that helps to invent better algorithms. With "better" we mean that the algorithms use fewer resources such as time or memory. We also consider parallel computation, distributed systems and learning problems. In these settings we might also optimize other types of resources. For example, in distributed systems we might want to minimize the amount of communication. We focuss on worst case guarantees. A large part of our time is devoted to the study of what is not possible. In other words, we study fundamental barriers for the existence of programs that use fewer resources than a given bound.

This program establishes the minimal requirements to students' knowledge and skills and determines the content of the course and educational techniques used in teaching the course. The present syllabus is aimed at faculty teaching the course and undergraduate students studying 01.03.02.62 "Applied Mathematics and Information Science". This syllabus meets the standards required by:

- Educational standards of National Research University Higher School of Economics;
- Bachelor educational program for 01.03.02.62 "Applied Mathematics and Information Science".

2. Learning Objectives

After this course, students will understand the following concepts:

- The complexity classes P, NP, coNP, #P, EXP, NEXP, L, NL, PSPACE, EXPSPACE, BPP, RP; relations among these classes (including separations through the hierarchy theorems); famous open problems.
- Problems that are complete for NP and PSPACE.
- Algorithms for prime testing and cryptography.
- Circuit complexity, the classes NC_k and AC_k, and P-completeness.
- Several classes in communication complexity.
- Agreement in online distributed systems and blockchain.

3. Main Competencies Developed after Completing the Study of This Discipline

After completing the study of the discipline the student will:
• understand the concepts listed above,
• be able to critically analyse resources used by a program (and optimize them at a high level),
• be able to recognize intractable problems and categorize their difficulty,
• have deeper understanding and trained problem solving skills of known materials in: algebra, probability theory, discrete math, algorithms
• be trained to read, speak (and possibly write) technical and mathematical English.

After completing the study of the discipline the student should have developed the following competencies:

Competence	Code	Descriptors (indicators of achievement of the result)	Educative forms and methods aimed at generation and development of the competence
Capable to identify the scientific nature of problems in the professional field of activity	UC-2	Students obtain necessary knowledge to understand and formulate the theoretical difficulty of problems they are solving.	Lectures and exercise sessions
Capable to solve problems in the professional field of activity using analysis and synthetis	UC-3	Students will be able to solve problems in theory of computing	Lectures and exercise sessions
Capable to describe problems and situation of the professional field of activity using mathematical language and methods	PC-1	Students will be able to formulate and identify problems in theory of computing	Lectures and exercise sessions
Capable to comprehend, modify and apply contemporary mathematical methods	PC-3	Students will be able to apply, combine and modify standard methods in theory of computing	Lectures and exercise sessions
Capable to build a mathematical model and perform its analysis for the specified theoretical or applied problem	PC-8	Students will be able to apply standard models of theory of computing to problems in this area	Lectures and exercise sessions

4. Course plan

№	Topic	Total hours	Contact hours			Self-study
			Lectures	Seminars	Practice lessons	
1.	The complexity classes P and NP, hierarchy theorems, reductions, and NP-completeness.	44	7	7		30
2.	L, NL, PSPACE, EXPSPACE, and PSPACE-completeness of some games.	18	3	3		12
3.	The polynomial time hierarchy and solutions for the P vs NP problem relative to specific oracles.	12	2	2		8
4.	Randomized computation and prime testing.	12	2	2		8
5.	Communication complexity	38	6	6		26
6.	Cryptography	14	2	2		10

7.	Circuit complexity and parallel computation	14	2	2		10
8	The science behind blockchain	38	6	6		26
9	Optional: Computational learning theory	0	0	0		0
	Total	190	30	30		130

Topic 1. Complexity classes P and NP, reductions, NP-completeness and hierarchy theorems.
In this topic many fundamental concepts of the course are introduced. We focus mainly on decision problems of sets of bitstrings. For a fixed set, we consider the problem of deciding whether a string belongs to the set.

We start by defining the class P of all sets that can be decided in polynomial time. The class NP contains all sets for which membership can be verified in polynomial time using a certificate. This class coincides with languages that can be decided by a non-deterministic variant of Turing machines (and hence the name NP = **N**ondeterminstic **P**olynomial time decidable).

It is a big open problem whether P=NP. More generally: if a solution to a problem can be verified efficiently, can we also find the solution more efficiently?

We study NP-completeness. If a set is NP-complete, then there exists no solution for the problem unless P=NP (i.e., unless all problems in NP can be decided in polynomial time). For these problems it is currently unlikely that a polynomial time algorithm can be found. We prove the Cook-Levin theorem, in particular we show that the set circuit-SAT, (which is the set of Boolean circuits that have an input that evaluates to one) is NP-complete. We also show that the sets 3SAT, *clique, subsetsum* and *Hamiltonian-path* are NP-complete.

In this part we follow chapter 7 in [1] and some proofs about NP-completeness from [2].

Topic 2. L, NL, PSPACE, EXPSPACE, and PSPACE-completeness of some games
We define space complexity classes and show that a large group of games are in the class PSPACE. Space is a much stronger resource than time because it can be reused. We prove the space analogue of the P=NP problem, which is called Savitch theorem: PSPACE = NPSPACE. Again we define PSPACE completeness. The decision problem TQBF considers a fully quantified Boolean formula and asks whether this formula is true. We show that this problem is PSPACE-complete. Then we show that finding winning strategies in games corresponds to PSPACE complete problems. We follow chapter 8 in [1].

Topic 3. Oracle computations and oracles the P vs NP problem relative to specific oracles.
We define computation for devices that have access to a device that stores or solves the decision problem for another language. For example, the class P^{SAT} consists of all sets that can be decided in polynomial time by a program that can use solutions of the SAT problem. (Note that P=NP if and only if P^{SAT}=P.) We show that there exist oracles A and B such that $P^A = NP^A$ and $P^B \neq NP^B$. This is remarkable because all theorems we study in this course remain true when we formulate them with computations relative to an oracle. But a proof that P = NP or P != NP can impossibly hold relative to an oracle. Hence, a fundamentally different techniques are needed to resolve this problem. This topic contains parts of chapter 9 in [1].

Topic 4. Randomized computation and prime testing.

The complexity class BPP corresponds to sets that are decidable by a polynomial time algorithm that uses randomness and for each input gives the correct answer with probability 1/3. We show that changing the value 1/3 to any other positive number smaller than ½ does not change the definition. A famous example of such a problem is to decide whether two polynomials are symbolically the same (i.e. can be transformed into each other after applying commutatitive, associative and distributive laws). If two such polynomials have at most polynomial degree then this can be checked by a probabilistic polynomial time algorithm. This algorithm simply checks the values of these polynomials over a finite field of polynomial size. Another set in BPP is the set of prime numbers. Trying divisibility of all sufficiently smaller primes requires exponential time, but a combination of two tests gives an algorithm for prime testing. We follow Section 10.2 in [1].

Topic 5. Communication complexity, property testing, PCP-theorems and inapproximability of NP-hard problems.

Imagine Alice holds a string x and Bob holds a string y of equal length n. They want to figure out whether $x = y$. We show that in every communication protocol for this problem, there exist x and y for which Alice and Bob communicate at least n bits. The field of communication complexity gives many more results about such problems. We consider also non-deterministic classes. We follow chapters 1,2 and 3 from [4] and use [5] for the application to online algorithms.

Topic 6. Cryptography.

We define one-way function and RSA cryptography. Information security will be thought in another course. Our goal is to introduce necessary background for topics 8A and 8B.

Topic 7. Circuit complexity and parallel algorithms.

In computational complexity, there are several classes that represent problems for which highly efficient parallel algorithms exist. They are given by NC_1, NC_2, … and AC_1, AC_2, … Famous problems in this class are exponentiation and multiplication of matrices. If time permits, we study a few more problems. Finally we study problems that are P-complete, i.e., for which it seems currently unlikely that efficient parallel algorithms will be found.

Topic 8. The science behind blockchain.

We study distributed systems both for online and offline problems. In particular, we consider the problem of in which distributed computing devices need to find agreement on a sequence of commands that need to be executed. In this topic we study the main chapters of the book [8].

Topic 9. Computational learning theory.

The following topic will be thought if time permits and upon request of the students. Previously, we discussed 3 types of resources a computation process uses: time, space and communication. Now we consider the problem of learning a classifier and we consider the number of samples that are needed in order to learn such a classifier. We study several problems that can be learned in polynomial time (using polynomially many samples).

People in machinelearning sometimes claim that practical instances from problems in NP are easily solvable (possibly after inventing a clever heuristic algorithm). In practice, many neural networks can be trained surprisingly well. Industrial SAT solvers are used in more and more applications. However, using cryptography we can construct artificial learning problems for which no learning algorithm seem effective. If time permits, we show that can not PAC-learn an automaton with k states in time polynomial in k. See [7]. This part overlaps a bit with the course *Introduction to Statistical Learning theory*, but differs in the sense that we simplify the statistical framework, and pay more attention to the computational issues.

5. Reading list

a. Required

1. M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.
2. Lecture notes provided by the teachers, publically available on http://wiki.cs.hse.ru.
3. T. Roughgarden. *Communication complexity (for algorithm designers)*, 2015. publically available on http://theory.stanford.edu/~tim/w15/l/l1.pdf

b. Recommended

4. E. Kushilevitz and N. Nisan. *Communication Complexity*, 1997.
5. S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

c. Optional

6. J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC Press, 2014.
7. M. Kearns and U. Vazirani. *An introduction to computational learning theory*, 1994.
8. R. Wattenhofer. *The Science of the Blockchain*, 2016.
9. Dasgupta, C. H. Papadimitriou and U. V. Vazirani, *Algorithms*, 2006.

6. Grading system

Type	Form	1 year				Details
		1	2	3	4	
Intermediate	Colloquium		1			Oral discussion of the theoretical material. Near the and of the second module. Grade - O_{col}.
	Homework	1	1			Each homework is broken into two parts of written assignments. The grade for a homework is computed as an average. Grade O_{hw1}, O_{hw2}.
Final	Exam		1			Written exam, 2h40m, Grade – O_{exam} . There are also bonus homework problems that can give additional bonus points to the exam.

The assessment consists of homeworks.

Final assessment consists of an oral test (called colloquim) and a written exam.

The grade formula

$$O_{final} = 0{,}7 \cdot O_{accum} + 0{,}3 \cdot O_{exam},$$

Where $O_{accum} = (1/2)O_{col} + (1/4)(O_{hw1} + O_{hw2})$

In other words:

Homeworks count for 35% of the final grade in total.

The colloquim is worth 35% of the final grade.

The final exam is worth 30% of the final grade.

All intermediate grades are computed without rounding. Rounding is applied only to the final grade. The following rounding is applied. Grades between 1 and 5 are rounded down. Grades between 0 and 1 as well as grades between 6 and 10 are rounded up. Grades between 5 and 6 are rounded arithmetically.

Table of Grade Accordance

Ten-point Grading Scale	**Five-point Grading Scale**	
1 - very bad 2 – bad 3 – no pass	Unsatisfactory - 2	**FAIL**
4 – pass 5 – highly pass	Satisfactory – 3	**PASS**
6 – good 7 – very good	Good – 4	
8 – almost excellent 9 – excellent 10 – perfect	Excellent – 5	

7. Guidelines for Knowledge Assessment

The final exam consists of a selection of problems. Students are allowed to use textbooks and notes. Each question will require to solve mathematical problems using materials presented during the lectures. Questions will be asked in English and students can answer either in English or in Russian (also in Dutch and French).

To be prepared for the final exam, students will be given 6-10 exercises every week (during the seminar). They can ask for hints and feedback on solutions.

8. Methods of Instruction

There will be 15 *theory lectures*, of 2 accademic hours each, during which conceptual ideas are explained.

Each lecture is followed by a *seminar* of 2 accademic hours, in which students solve exercises that deepen the understanding of the materials and train problem solving skills. Each weak a list of 6-10 exercises are given.

Every week 2 exercises are given for homework. Around the end of the month, these homeworks need to be submitted.

The student can consult the professors and an assistent during the office hours. Intensive guidence can be given during the office hours of the teacher or the assistent.

Lecture notes, exercise lists, a summary of each lecture, and all practical information is maintained on the wikipage of the course: http://wiki.cs.hse.ru .