Computer Science and Artificial Intelligence

A new B.Sc. program at Neapolis University Pafos




bit.ly/csai2023

bit.ly/csai-nup

t.me/csainup

Table of contents

It’s clickable!

About. The B.Sc. program in Computer Science and Artificial Intelligence was launched at Neapolis University Pafos (Cyprus) in 2022 in collaboration with JetBrains, a company that has supported Computer Science education for over 20 years. Graduates from the B.Sc. and M.Sc. programs developed in partnership with JetBrains are now employed at leading tech companies, including Google, JetBrains, and Meta, and many continue their studies in Ph.D. programs at prestigious universities such as NYU, Princeton, and UCSD. Some graduates are even founding startups in Silicon Valley and securing funding from Y-Combinator. In Cyprus, JetBrains operates two major offices in Paphos and Limassol, employing over 300 people.

Curriculum. The curriculum emphasizes mathematics: you'll study algebra, discrete mathematics, mathematical analysis, probability, statistics, theoretical computer science, and other core subjects. This foundational knowledge is essential for future computer science professionals, especially in the era of large language models, enabling you not only to utilize current technologies but also to understand their limitations and innovate new solutions. You'll also dive into applied courses such as artificial intelligence, compilers, data science, databases, large language models, machine learning, programming languages, and robotics, with many taught by industry experts. You'll also gain extensive hands-on experience through labs, projects, and internships.

Classmates. You will be studying with highly motivated and well prepared students. Among our students there are those who participated as high-school students in the most prestigious international Olympiads: International Mathematical Olympiad (IMO), International Olympiad in Informatics (IOI), International Biology Olympiad (IBO), Open World Astronomy Olympiad (OWAO). Our students advanced to the Finals of the International Collegiate Programming Contest (ICPC), won gold medals in the International Mathematics Competition for University Students (IMC), and won the International Workshop on Logic and Synthesis Programming Contest.

Learning outcomes

You will learn how to:

1 **Explain**

Explain the fundamentals of mathematics and computer science, which form the basis of modern software, artificial intelligence, and robotics

2 **Design**

Design innovative hardware architectures for next-generation processors, computers, and sensors by applying processor design methods and tools

3 **Create**

Create general and special-purpose operating systems, along with performance analysis tools

4 **Develop**

Develop optimal high-load services and distributed systems for maximum efficiency

5 **Construct**

Construct programming languages, compilers, interpreters, virtual machines, and frameworks to facilitate software development

6 **Apply**

Apply artificial intelligence techniques to enhance search engines, social networks, and intelligent assistants

7 **Use**

Use robotics in autonomous robots, the internet of things, and other applications

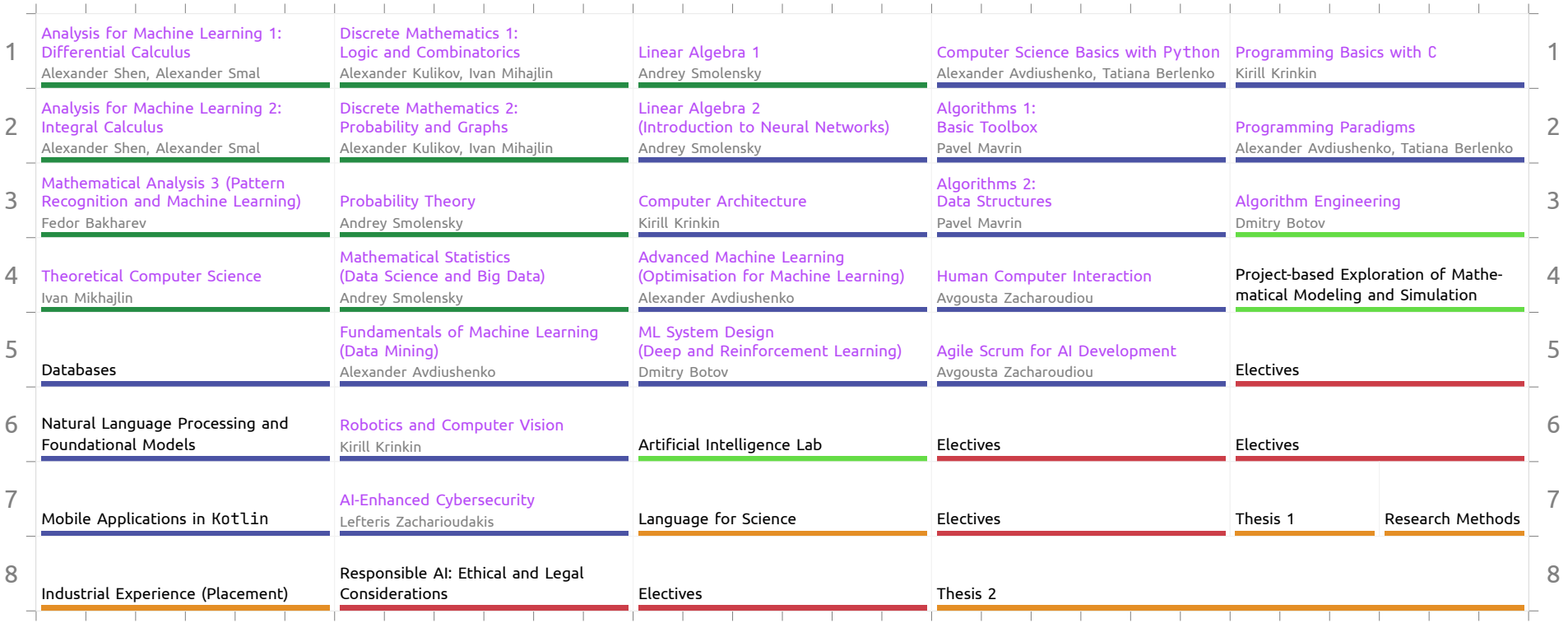
8 **Implement**

Implement machine learning and deep learning algorithms across various domains, such as computer vision, natural language processing, reinforcement learning, and recommendation systems

9 **Establish**

Establish and manage IT businesses, fostering effective team collaboration, process management, and customer and partner relations

Curriculum (2024–2025)



- Color codes: mathematical courses, programming courses, project-based courses, electives, miscellaneous
- Electives in Fall 2024: Advanced Graph Theory, Algorithms for NP-hard Problems, Complexity Theory, Cryptography, File Systems, Information Theory, Practical Formal Methods

Professors

The B.Sc. program is led by a team of seasoned professors and coordinators with over 10 years of expertise in Computer Science research and education. We provide you with an opportunity to familiarize yourself with the program's professors below, along with links to their teaching materials such as online courses, books, video recordings, and home assignments. We believe it's essential for you to investigate the professors of the program beforehand, regardless of which B.Sc. program you intend to pursue. We urge you to explore the professors' qualifications and professionalism on the internet to ensure that you receive an outstanding education.

Artem Anisimov holds a Ph.D. in mathematics and works on distributed storage systems. He is passionate about reliable and performant software, and enjoys sharing his experience. He has lectured at MIPT for more than 5 years on topics related to storage and operating systems.

Alexander Avdiushenko holds a Ph.D. degree in math modeling. He is a dedicated educator and researcher at JetBrains Education and Research team. With over three years of experience as a Data Scientist and over eight years of teaching at top universities, he has honed his skills in deep learning and data optimization. Passionate about both machine learning and human learning, Alexander has developed an engaging Machine Learning course that showcases his expertise and commitment to education. Sample video lecture: Developing Critical Thinking in the Era of ChatGPT.

Fedor Bakharev holds a PhD in Functional Analysis and doing research in Mathematical Physics especially in Spectral theory of differential operators. He has more than 15 years experience in teaching various analytical courses for university students. He has also been involved in several industrial IT projects as mathematician.

Tatiana Berlenko is a researcher in the Mobile Robot Algorithms Lab at JetBrains Research interested in Simultaneous Localization and Mapping in 3d. Tatiana has six years of experience teaching Computer Science and Software Engineering at universities, creating MOOCs, and conducting STEM-based robotics schools for students and schoolchildren.

Dmitry Botov holds a Ph.D. degree in Artificial Intelligence. He has over 12 years experience in Machine Learning & Data Engineering. He successfully created R&D departments in large IT companies from scratch. For more than 14 years he has been developing educational programs and teaching courses in Software Engineering and ML. His research interests focus on generative artificial intelligence, natural language processing techniques, and large language models. He designs and teaches original courses on Project Management in Data Science and ML System Design, and actively uses the practices of project-oriented and problem-based learning to develop the skills of machine learning engineers and researchers. Sample video lecture: Generative AI in Content Production: From Research to Application.

Edward A. Hirsch is a Professor of Computer Science at Ariel University. For many years, he held a permanent Research Lead position at the Steklov Institute of Mathematics at St.Petersburg and contributed to various topics in Algorithms and Complexity. Edward is an advisor to quite a number of PhD students who now work in academia and industry all over the world.

Savvas Chatzichristofis is the Vice-Rector of Research and Innovation and Professor of Artificial Intelligence. His research focuses on the intersection of AI, computer vision, and robotics, and he has been involved in multiple R&D projects funded by European and National organizations.

Kirill Krinkin holds a Ph.D. degree in Computer Science. Kirill is doing research in robotics autonomy and is an accomplished author or co-author of over 100 technical papers. For more than 20 years, Kirill is a seasoned organizer of STEM schools worldwide, has collaborated with top-tier universities and international companies in software engineering, operating systems, computer networks, autonomous mobile robots, and co-evolutionary hybrid intelligence. Under his expert guidance, student teams won the AI-DO twice, once at ICRA and once at NeurIPS. Sample video lecture: Linux Container Technologies and Their Applications.

Avgousta Kyriakidou is an Assistant Professor and the Programme Coordinator for the BSc in Computer Science and Artificial Intelligence at Neapolis University Pafos, with over 13 years of experience in Higher Education. She has served as an Associate Professor in Computing at the University of Greenwich, UK, since 2011. Dr. Kyriakidou holds a PhD in Information Systems from the London School of Economics (LSE), along with additional degrees from LSE, the University of Greenwich, and the University of Cyprus. Her current research focuses on digital innovation, business agility, digital systems development, and Artificial Intelligence within business. She has played key roles in programme development, quality assurance, and employability initiatives, and is certified in SCRUM, DevOps, and AgilePM. A Fellow of the Higher Education Academy (UK), she currently serves as an Associate Examiner for LSE's external programme and an External Examiner for the University of Hull.

Alexander S. Kulikov holds Ph.D. and Dr.Sci. degrees from Steklov Mathematical Institute. He serves as the head of the Laboratory of Algorithms and Complexity Theory at JetBrains Research. Alexander co-authored online courses Data Structures and Algorithms and Introduction to Discrete Mathematics for Computer Science that are available at Coursera and edX with over a million enrolled students. Sample video lecture: The Satisfiability Problem.

Pavel Mavrin is an expert in algorithms and data structures. He has published many educational videos on his YouTube channel. Pavel has participated in many programming competitions. His most prestigious title is ICPC World Champion 2004. Sample video lecture: Parallel algorithms.

Ivan Mihajlin is a theoretical computer scientist interested in computational complexity theory. He did his Ph.D. studies in the University of California, San Diego. He was a visiting graduate student at the Simons Institute for the Theory of Computing at Berkeley. Sample video lecture: Non-equivalence of Hardness Assumptions.

Niyaz Nigmatullin is a software engineer in Applied Program Analysis Laboratory at JetBrains. He used to research and develop compression algorithms and data storing techniques. He teaches algorithms and data structures, competitive programming, as well as organizes various competitions. He is a world champion in the most prestigious competitive programming competition ICPC in 2012 and 2013. Sample video lecture: How to Prepare for a Programming Contest?

Ilya Sergey is an Associate Professor at the School of Computing of National University of Singapore, where he leads the Verified Systems Engineering lab. Ilya got his PhD in Computer Science at KU Leuven. He does research in programming language design and implementation, distributed systems, software verification, and program synthesis.

Alexander Shen holds a Ph.D. degree in Computer Science. He is an Directeur de recherche 2 cl., LIRMM, CNRS, University of Montpellier 2. He is an author of many books on Mathematics and Computer Science.

Alexander Smal holds a Ph.D. from Steklov Mathematical Institute. He is a theoretical computer scientist interested in computational complexity, theory of information and algorithms. He has fifteen years of experience in teaching computer science courses to university students and several year of experience as a software engineer.

Andrei Smolensky holds a PhD in algebra and is doing research on the interfaces between algebra and machine learning. He has ten years of experience in teaching introductory and advanced courses in algebra (in particular, those geared towards applications) to university and high school students.

Lefteris Zacharioudakis is an Assistant Professor of Cybersecurity at Neapolis University Pafos in Cyprus. He holds a BSc, MSc, and PhD from the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute.” His expertise lies primarily in data, computer, and network security, with a particular focus on cryptography and systems security.

Research

We invite you to join us in conducting research across a wide range of Computer Science fields, including: Algorithms, Artificial Intelligence, Bioinformatics, Complexity Theory, Machine Learning, Programming Languages, and Robotics. Engaging in research is highly rewarding for several reasons:

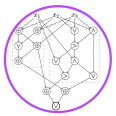
Personal satisfaction. You'll experience the joy of exploring uncharted territories, solving problems without existing solutions, and contributing to the advancement of the field. Your innovative ideas have the potential to make a real impact on everyday life.
Networking and travel. By attending research conferences, you'll have the opportunity to see the world and meet inspiring people who share your passion and work in similar areas.

We're eager to support you in publishing a paper in a top-tier conference or journal. Such publications can greatly benefit your future career:

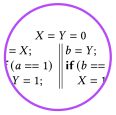
Further education. Strong publications will enhance your chances of being accepted into prestigious M.Sc. or Ph.D. programs. Many graduates of our B.Sc. program have continued their studies at top institutions such as EPFL, ETH, GWU, NYU, and UCSD. If you aim for a career in academia, a robust publication record is essential for securing positions as a professor, lecturer, or research scientist.
Industry opportunities. In the tech industry, especially in R&D departments, a track record of high-impact publications can set you apart and lead to more advanced job offers or career advancements.
Skill development. Publishing also hones your writing and communication skills, helping you articulate complex ideas effectively to a broader audience.

Here are some of our recent publications, many co-authored with students.

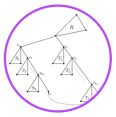
Nikolai Chukhin, Alexander S. Kulikov, Ivan Mihajlin.
Toward Better Depth Lower Bounds: Strong Composition of XOR and a Random Function.
Proceedings of the 42nd International Symposium on Theoretical Aspects of Computer Science (STACS 2025).



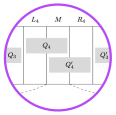
Daniil Averkov, Tatiana Belova, Gregory Emdin, Mikhail Goncharov, Viktoriia Krivogornitsyna, Alexander S. Kulikov, Fedor Kurmazov, Daniil Levtsov, Georgie Levtsov, Vsevolod Vaskin, Aleksey Vorobiev.
Cirbo: A New Tool for Boolean Circuit Analysis and Synthesis.
Proceedings of the 39th Annual AAAI Conference on Artificial Intelligence (AAAI 2025).



Evgenii Moiseenko, Matteo Meluzzi, Innokentii Meleshchenko, Ivan Kabashnyi, Anton Podkopaev, Soham Chakraborty.
Relaxed Memory Concurrency Re-executed.
Proceedings of the ACM on Programming Languages (POPL 2025).



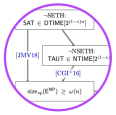
Evgenii Feder, Anton Paramonov, Pavel Mavrin, Iosif Salem, Stefan Schmid, Vitaly Aksenov.
Toward Self-Adjusting k-ary Search Tree Networks.
Proceedings of European Symposium on Algorithms (ESA 2024).



Tatiana Belova, Nikolai Chukhin, Alexander S. Kulikov, Ivan Mihajlin.
Improved Space Bounds for Subset Sum.
Proceedings of European Symposium on Algorithms (ESA 2024).

Iskander Azangulov, Andrei Smolensky, Alexander Terenin, Viacheslav Borovitskiy.
Stationary Kernels and Gaussian Processes on Lie Groups and their Homogeneous Spaces I: the compact case.
Journal of Machine Learning Research. 2024.

Vera Kuznetsova, Áine Coogan, Dmitry Botov, Yulia Gromova, Elena V. Ushakova, Yurii K. Gun'ko.
Expanding the Horizons of Machine Learning in Nanomaterials to Chiral Nanostructures.
Advanced Materials. 2024.



Tatiana Belova, Alexander S. Kulikov, Ivan Mihajlin, Olga Ratseeva, Grigory Reznikov, Denil Sharipov.
Computations with polynomial evaluation oracle: ruling out superlinear SETH-based lower bounds.
Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA 2024).

Yaroslav Alekseev, Yuval Filmus, Alexander Smal.
Lifting Dichotomies.
Proceedings of Computational Complexity Conference (CCC 2024).

Competitions

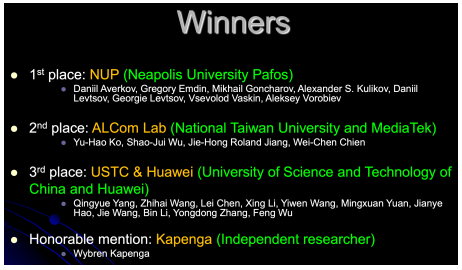
We would be happy to help you to prepare for various types of competitions. Participating in olympiads has many advantages: you will learn a lot and you will connect with like-minded peers, which can lead to future collaboration or job opportunities, as many companies prioritize candidates with olympiads experience.

Programming competitions

The International Collegiate Programming Contest (ICPC) is the largest algorithmic programming contest for college students. Of more than 3,000 participating universities, only about 130 teams advance to the world finals. NUP is the only university from Cyprus, whose team advanced to the ICPC World Finals. We have a community of studentes who are interested in ICPC, and they have ICPC trainings twice a week.

Mathematical competitions

International Mathematics Competition for University Students (IMC) is an annual competition for students under the age of 30. The competition is open to students from all over the world and is one of the most prestigious student competitions in mathematics.
At NUP, we conduct weekly training sessions and, based on their results, send a team to participate in IMC. In 2023, students brought home two gold medals and one bronze medal, whereas in 2024, they got five gold and two bronze medals!



Research competitions

In 2024, the NUP team won the IWLS 2024 Programming Contest. The goal in this competition is to synthesize efficient Boolean circuits for 100 Boolean functions given by their truth tables. In 2023, it was Google DeepMind who got the first place in the competition. In 2024, the NUP team reduced the size of the best circuits from 2023 by 12%. For some of the circuits, the size reduction was as dramatic as 83%!

Workshops

In addition to our main courses, we also offer intensive educational activities, such as student workshops and programming camps. These provide an excellent opportunity not only to learn new skills but also to connect with students from around the world who share your passions. Recently, we have organized the following workshops in Paphos, Cyprus.

The Pafos Programming Camp

Free, on-site boot camp for competitive programming hosted by Neapolis University Pafos (NUP). This boot camp is designed to help top-level EU teams prepare for the ICPC Regional and World Finals. The program spans 8-10 days of contests (Regionals final level), along with analyses, upsolving time, evening sports, and other leisure activities, as well as a day off with an organized excursion. See the webpage of the upcoming camp in 2024, as well as the past camp in 2023.

Joint Advanced Student School (JASS)

Joint Advanced Student School (JASS) is an annual international initiative to study emerging technologies using project-driven software development. The school is a place where students and professors from different countries gather for short-term, project-oriented, project-driven education in international teams. Over the course of 7-14 days, each team of students creates and delivers a finished project. We collaborate with world leading universities like CMU, TUM, ICL and, of course, with Cyprus universities. Information about past schools can be found at jass.school

School on the Practice and Theory of Distributed Computing

The school covers both fundamental and state-of-the-art topics on the theory and practice of distributed systems. Speakers: Gael Thomas, Dan Alistarh, Gregory Chockler, Alexey Gotsman, Pierre Sutra, Maurice Herlihy, Prasad Jayanti, Nikita Koval, Sergio Rajsbaum, and Panagiota Faturu. See the webpage.

Career

One can explore various career options with a bachelor's degree in Computer Science and Artificial Intelligence. These career paths include roles such as an Artificial Intelligence/Machine Learning Engineer, Data Scientist, Software Engineer, Robotics Engineer, Natural Language Processing Engineer, Technical Product Manager, Research Scientist. The demand for CSAI professionals is increasing, and there are numerous exciting career prospects in this area. By completing this B.Sc. program, you will gain practical skills that are highly valued in the job market, establish a strong foundation of knowledge for job interviews, and have the opportunity to engage with a supportive community of peers and industry professionals. The CSAI B.Sc. was launched in 2023 and does not have its own graduates yet. At the same time, the JetBrains company has supported various B.Sc. program for the last 20 years. Meet some of the graduates of such programs below.

Ilya Biryukov (2013)
Currently, a Team Leader at Google (Germany), working on C++ compiler. Before this, worked at JetBrains on Resharper C++ and CLion Nova as well as at Google on C++ Language Server.

Aleksei Artamonov (2014)
Currently, a Self Driving Car Developer at Avride (Serbia), working on implementing computer vision systems for business and innovate products.

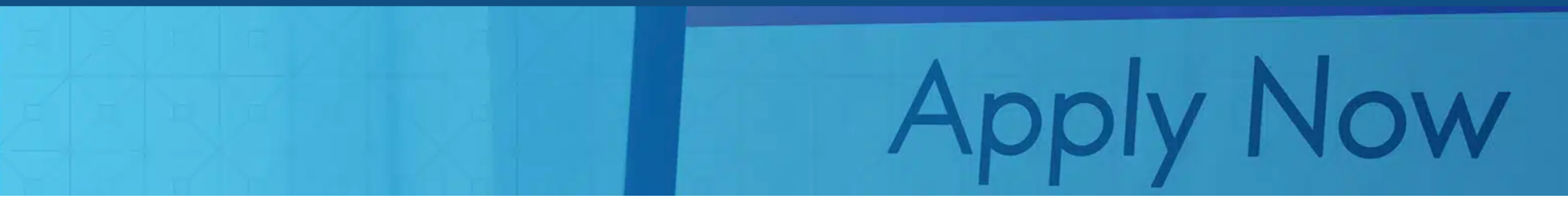
Sergey Lebedev (2015)
Currently, a Software Engineer at Google DeepMind (UK), working on JAX, a library for numerical computing at scale. Before this, worked at JetBrains on probabilistic models for epigenetics data, at Criteo on scalable machine learning infrastructure.

Alexander Golovnev (2016)
Currently, an Assistant Professor at Georgetown University (USA), working on computational complexity and cryptography. Before this, got his Ph.D. from NYU and was a postdoc at Yahoo Research and Harvard University.

Valentina Kiryushkina (2018)
Currently, a Team Leader at JetBrains (Germany), working on JetBrains Academy Plugin.

Yurii Rebryk (2019)
Currently, a Founder and CEO at GetFluently.app (USA). Before this, was a Machine Learning intern at Google, Nvidia, Lyft Level5, and Amazon Alexa.

Aleksandra Novikova (2022)
Currently, a Data Science M.Sc. student at EPFL (Switzerland). Before this, was an intern at JetBrains and Google.



Admission (2025)

1. **Compile your documents.** Prepare your documents, fill out the application form. Follow these recommendations on writing your resume and personal statement. If you have any questions, contact us via nup@jetbrains.com or join our WhatsApp and Telegram chats.
2. **Send your documents to the university's admissions office.** Email your documents to student.advisor4@nup.ac.cy. Submit your application by 11:59 pm UTC June 12, 2025.
3. **Take the CSAI Entrance Test.** The entrance test will take place two times: on April 27 and on June 15 (08:00 am UTC). It will consist of 10 math problems and 6 informatics problems, and you will have four hours without a break to complete them.
4. **Participate in an interview.** The applicants with the best entrance exam scores will be invited to participate in an interview, taking place between June 18 and 30, 2025. The interview is a 40-minute conversation with the program's organizers and teachers. We will discuss your motivation and previous experience, and we'll ask you to solve a few problems in the fields of mathematics and programming.
5. **Receive a decision about the scholarship.** Scholarship decisions will be made by July 1, 2025. If you receive one of the 15 scholarships, JetBrains will provide a grant that covers the tuition and accommodation fees, as well as a monthly stipend of €300.
6. **Collect documents for a visa (if necessary) and enroll.** You'll need to provide an assortment of documents to Neapolis University Pafos, including your academic qualifications, English Proficiency Certificate (if you can't receive an English Proficiency Certificate in your place of residence, please contact us so we can provide a code that you can use to take the Password test for free), Medical Certificate, payment confirmation, and additional documents for the Cypriot authorities.

Scholarship

Education is a significant investment in your future, and we work hard to make it an affordable reality for students worldwide. Each year, we offer 15 select applicants a grant that covers tuition fees and accommodation, and provide them with a monthly stipend of €300. The JetBrains scholarship covers:

	JetBrains	You
Tuition	✓	✗
Room	✓	✗
Medical Insurance	✓	✗
Visa fee (if applicable)	✓	✗
Pocket money (€300 per month)	✓	✗
Time and effort	✓	✓

Preparing

- The best way to prepare (for the studies and for the entrance test) is to join one of our free Online Clubs for School Students:

  - AI Club
    - instant automated checks
    - contests every two months
  - Coding Club
    - three-hour mashup of tasks every week
    - ask questions directly in Discord chat
  - Math Club
    - live lectures and discussions
    - weekly homework assignment of 15 problems

- You may want to brush up on the basics of the following topics: logarithms; trigonometry; combinatorics; limits; recursion and induction; algorithms.
- Finally, we encourage you to practice solving 2024 Entrance Test.

Accommodation




Our students reside at Paphos Gardens Apartments during their studies. This apartment complex is just a 10-minute walk from the university and offers a range of amenities, including two swimming pools, a cafeteria, a laundry facility, and other, like a table tennis.

Students are accommodated in two-bedroom apartments, with two students sharing each bedroom. The apartments feature a shared living room, a bathroom, a fully equipped kitchen for preparing meals, and either a balcony or a terrace, depending on the unit. Students also have the option to purchase discounted meal plans, with choices ranging from breakfast only to half board.

For those who need to stay over the summer, we provide accommodation in the university dormitory, conveniently located in the same building as the university itself. Here, students live in double rooms with private bathrooms. The dormitory also has several shared kitchens and laundry facilities on each floor, as well a gym and a swimming pool on the territory.

Cyprus

Cyprus is an island country located in the eastern Mediterranean Sea, known for its rich cultural heritage and history. Cyprus is a popular tourist destination, offering beautiful beaches, a warm climate, and diverse historical sites.

In recent years, it has also become an emerging hub for the tech and IT industry, attracting startups and international companies with its favorable business environment. With a sunny climate and a growing digital economy, Cyprus offers a unique blend of leisure and professional opportunities.

Prices for food and groceries in Paphos are similar to the prices in other cities of Cyprus. A meal in a mid-range restaurant costs around €15 to €25 per person, while a fast-food meal can cost around €8. Transportation costs in Paphos: a one-way ticket on public transport costing around €1.50. Other expenses, such as utilities, healthcare, and entertainment, can vary depending on your lifestyle and needs. You may want to compare the cost of living in Paphos with other cities using this online calculator.

Cyprus has a subtropical climate with hot and dry summers and mild winters, with average temperatures ranging from 25 to 35 degrees Celsius in summer and 10 to 20 degrees Celsius in winter. Rainfall is relatively low throughout the year.

In addition to their studies, our students engage in various activities, most of which they have organized themselves: Yoga with an instructor, Volleyball, Board games, Table tennis, Book club, Movie club. We also occasionally gather for picnics, historical walks, tennis tournaments, and celebrate holidays together.

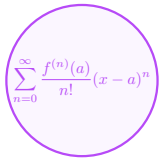
Let's stay in touch!

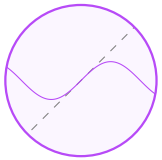
Landing
bit.ly/csai2023

Materials
bit.ly/csai-nup

Chat
t.me/csainup

Calculus 1: Foundations and Differential Calculus





1st semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

The first part of the course of calculus consists in studying the basic concepts and theorems of differential calculus and functions of one real variable. The course goes through the basic axiomatic of real numbers, number sequences, series (infinite sums), and functions of one real variable — all these subjects are classical for the initial course of calculus. However, students are invited to study some additional topics in depth. Among examples are the realization of real numbers in programming languages, basic notions of general topology on a line, and geometrical and combinatorial applications of calculus. The main goal of the first semester is to teach basic mathematical culture and important skills required in the consequent mathematical and programming courses. Examples of such skills are finding the extrema of functions using Fermat's principle, comparisons of asymptotics of functions, and approximate computation of values by the Taylor formula.

Alexander Shen

Alexander Smal

Prerequisites

- Mathematics at the level of the school curriculum

Where you will need it

- In all subsequent mathematical disciplines of an analytical nature
- To understand the courses: Calculus 2, 3, Probability Theory, Mathematical Statistics, Convex optimization, Fundamentals of Machine learning
- For the formation of a common mathematical culture

Syllabus

- Axiomatic of real numbers and concrete realizations
- Sets of real numbers. Supremum and infimum
- Number sequences, convergence
- The number e
- Series, summation
- Functions of one real variable. Limits, continuity, general topology of the real line
- Differentiation and derivatives. Rolle's, Fermat's, Cauchy's and Lagrange's theorems
- Higher derivatives. Mathematical analysis of functions
- Taylor's formula

You will learn

- To define basic mathematical notions in rigorous manner
- To evaluate qualitatively and quantitatively the asymptotical behavior of functions and sums
- To analyze local behavior of functions in various optimization problems
- To operate with abstract structures and implement them for solving naturally posed problems

Algebra 1: Linear Algebra

$$\frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

1st semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

This course addresses the underlying structures of computations: from modular arithmetic and complex numbers to polynomials and rationals function, vectors and matrices. The goal is to learn the language and master the abstractions used for working with computable objects (rings, fields, vector spaces, etc.), while also investigating the wide range of applications: from cryptography and coding theory to geometry and signal processing.

The course serves as an introduction to linear algebra, which is one of the most widely used areas of mathematics and a foundational tool in neural networks.

During the course the students will learn how to implement most of the notions discussed in it on a computer and how to use some already existing implementations, with their strong sides and limitations.

Capstone Project

The students will implement their own set of routines for linear algebra, up to the procedure for solving a system of integer linear equations.

Andrei Smolensky

Prerequisites

- Mathematics at the level of the school curriculum

Where you will need it

- In all subsequent mathematical courses
- For successful passing of various courses (for example, Algebra 2, Mathematical analysis 1, Algorithms 1, Machine learning)

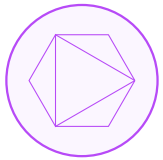
Syllabus

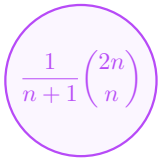
- Algebraic structures: ring, field, vector space, matrix
- Modular arithmetic: residue classes, congruences, Chinese remainder theorem
- Complex numbers: algebraic and geometric interpretation, trigonometric form, roots of unity
- Polynomials and rational functions: Bezout theorem, interpolation
- Linear algebra: linear independence, dimension, systems of linear equations, linear maps and their matrices

You will learn

- To use a variety of computational tools and notions
- To translate problems into the language of linear algebra and solve them
- To switch freely between different languages used for describing linear phenomena

Discrete Mathematics 1: Logic and Combinatorics





1st semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

The main goal of the course is to learn how to design and analyze discrete mathematical models in various areas of computer science. We will focus on formal proofs of correctness: will cover various proof strategies, will see how an intuitively correct proof may go wrong (which can lead to a failure of software or hardware in practice). Being able to distinguish between correct and incorrect proofs becomes more and more important in the era of large language models. The course is equipped with a large number of exercises, both theoretical and practical. Using theoretical exercises, the students will practice solving problems and writing down formal proofs. Using programming exercises, the students will see a rich variety of practical applications where discrete mathematical ideas arise and will practice implementing (as well as testing and debugging) efficient solutions.

In the first part of the course, we will learn the basics of logic, set theory, and combinatorics. We will practice using their tools to solve various real life problems.

Capstone Project

There will be two capstone projects in this course:
15-puzzle solver. By implementing a collection of methods for working with permutations, you will be able to implement a program that solves any instance of the well-known 15-puzzle in the blink of an eye.
Nonogram solver. By encoding an instance of Nonogram as a formula in CNF and using state-of-the-art SAT solvers, you will be able to implement an efficient Nonongram solver.

Alexander Kulikov

Ivan Mihajlin

Prerequisites

- Mathematics: proofs, functions (logarithm, polynomial, exponent)
- Programming: Python basics (input/output, loops, recursion)

Where you will need it

- For designing and analyzing efficient algorithms and programs
- For successful passing of the following courses: Algorithms, Probability Theory, Theoretical Computer Science, Mathematical Statistics, Fundamentals of Machine learning

Syllabus

- Proofs
  - Existence and optimality: constructive and nonconstructive proofs, proofs of nonexistence, proofs of optimality.
  - Proofs from computational point of view: certificates, proof systems, complexity classes, interactive proofs, zero-knowledge proofs.
  - Mathematical induction: strong induction, well-ordering principle, strengthened statements, nested statements.
- Logic
  - Predicates: normal forms, Post's theorem, quantifiers.
  - Satisfiability: SAT solvers, proofs systems, formal verification.
  - Circuits: computational models, lower and upper bounds.
- Sets
  - Cardinality: countable sets, diagonal argument.
  - Orders: orders and mathematical induction, Dilworth's theorem.
- Combinatorics
  - Enumeration: enumerating subsets, permutations, bracket sequences; branch and bound methods; dynamic programming; ILP solvers.
  - Permutations and combinations: binomial coefficients, estimates.
  - Recurrence relations: ways of solving recurrence relations, divide-and-conquer and backtracking algorithms.
  - Catalan numbers: three proofs of the formula, various occurrences of Catalan numbers.
  - Generating functions: basic rules of working with generating functions.

You will learn

- To write down short and rigorous proofs of mathematical statements
- To estimate the running time of algorithms and programs
- To apply discrete mathematics ideas in various areas of computer science
- To implement brute force search, coding and decoding, reductions to the satisfiability problem

Computer Science Basics with Python

1st semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

Our goal is to level up students' general knowledge in different areas of computer science and to teach them how to use a variety of tools to solve problems in Python.

In this course, we do not aim to learn the syntax of the Python language. We will use the language as a tool to explore different concepts, such as storing different formats of data in computer memory. During the course you will work with various tools for application development and deployment, learn libraries for working with data and developing web applications.

Special attention is paid to practice rather than theory. We do not aim to learn the syntax of the Python language, but students who do not have sufficient programming experience in this language will be able to learn it in additional classes.

Capstone Project

Students will build a client-server application, applying the tools and techniques learned during the course. The specifics of the project (e.g., its purpose or domain) are up to the students, but it must include key elements such as Docker for containerization, database integration, testing, and CI/CD pipelines. By the end of the course, students will have a working application that follows modern development practices.


Alexander Avdiushenko


Tatiana Berlenko

Prerequisites

- Mathematics and Computer Science at the level of the school curriculum

Where you will need it

- Creating and optimizing software, analyzing data, and improving system performance.
- Working as Software Developer, ML Engineer, Data Scientist, DevOps Engineer.

Syllabus

- Introduction to Software Engineering. Introduction to PyCharm and Python development.
- SSH. Working on a remote server. Networks. Docker. Docker compose
- Version control systems. CI/CD. Github actions
- Comparison of different programming languages.
- Testing. TDD. Logging. DevOps
- Introduction to DataBases. Sql and NoSQL databases.
- Client-server approach. REST API. Using Flask library.

You will learn

- How to set up a development environment and write Python applications using PyCharm.
- To work on remote servers using SSH.
- To use Docker and Docker Compose for containerization.
- To work with version control systems and implement CI/CD pipelines using GitHub Actions.
- To compare different programming languages and choose the right one for the task.
- To apply testing methodologies, such as TDD, and implement logging and DevOps practices.
- To design and work with both SQL and NoSQL databases.
- To build client-server applications and REST APIs using the Flask library.

Programming basics with C

#include

1st semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

This course introduces the Linux programming environment and the basics of C programming, which is essential for every programmer to understand computer and system programming. C language constructs are closely related to typical machine instructions, making it a fundamental language for tasks such as operating system development, embedded systems programming, and robot programming. Learning C programming techniques not only allows students to write efficient programs, but also deepens their understanding of how computers execute those programs. The Linux software environment is the industry standard for the development of any server and embedded systems. Linux defines basic system abstractions such as files, processes, sockets, and others, and contains a wealth of basic tools for developing and debugging programs. After taking this course, all of them will become good friends of future professional software developers.

Capstone Project

In this course, the students have a possibility to implement various projects:

- File Manager with Text User Interface
- Archivator/Dearchivator
- Sound Files Editor
- Command Line Image Manipulation Program
- Process Browser
- Text-based Chat Program
- ELF File Format Inspector
- Command Shell
- Memory Inspection Tool
- GIF Animation Creator

Kirill
Krinkin

Prerequisites

- High school level of informatics

Where you will need it

- To study Computer architecture
- To study Operating systems development
- To make projects with robots in SmartCity Lab and Industrial robot Lab

Syllabus

- Program compilation, linking, loading into memory, execution.
- Basic Syntax and Structure of C Programs
- Data Types, Variables, and Operators
- Control Structures (Conditional Statements and Loops)
- Functions and Recursion
- Arrays, Strings, and Pointers
- Structures, File I/O, and Preprocessor Directives
- Linux Shell and Command Line Interface
- Development Environment and tools
- Git and introduction to DevOps
- Networking for remote development
- Working with robots in SmartCity Lab
- Working with robots in Inductrial robot Lab

You will learn

- How to setup developmnent process in Linux environment
- How to create system level software
- To create and debug programs with C
- How to interact with Linux based embedded systems

Calculus 2: Integration and Functions of Many Variables

2nd semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

The second part of the course of calculus begins with the definition of indefinite integral (antiderivative) and the basic rules of computations for them. Shortly after the course switches to definite integral and its connection to anti-derivatives. The course makes an emphasis on geometrical understanding and applications of integrals, as well as approximate methods of numerical integration. Special attention is paid to improper integrals and its relation to series summation. The main part of the course is the differential calculus of functions of several real variables. The goal is to study functions of several variables for an extremum, to solve optimization problems (including conditional optimization), in particular, the gradient descent method. The study of functions of several variables is theoretically based on understanding open and compact subsets of $\mathbb{R}^n$. For this reason, there will be several lectures devoted to the language of general topology including the very important notion of a metric space. The course of the second semester ends with the implicit function theorem, and the notion of smooth submanifold of $\mathbb{R}^n$ which is becoming extremely important in modern machine learning.

Alexander Shen

Alexander Smal

Prerequisites

- Calculus 1
- Discrete Mathematics 1
- Algebra 1 and 2

Where you will need it

- In all subsequent mathematical disciplines of an analytical nature
- To understand the courses: Calculus 3, Probability Theory, Mathematical Statistics, Fundamentals of Machine learning
- To create a common mathematical culture

Syllabus

- Antiderivatives
- Definite integrals. Fundamental theorem of calculus
- Applications of the definite integral: area, volume, length. Curves of infinite length and other fractals
- Numerical integration: trapezoid formula, Simpson formula. Combinatorial applications
- Improper integrals
- Metric spaces, examples, equivalent metrics
- General topology of the space $\mathbb{R}^n$
- Continuous and differentiable functions of several variables. Partial derivatives. Gradient of a function
- Higher order derivatives and differentials
- Optimization problems in several variables. Training of neural networks
- Implicit function theorem
- Submanifolds and their tangent spaces. Lagrange multiplier theorem.

You will learn

- To use integration in practical problems
- To make general topological reasoning about existence of extrema of functions on metric spaces
- To evaluate qualitatively and quantitatively the behavior of functions of several variables
- To formalize common sense optimization problems in rigorous terms and solve them exactly or numerically

Algebra 2: Analytic Geometry

$$v - 2\frac{(u,v)}{(u,u)}u$$

2nd semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

The second part of the algebra course is dedicated to a deep dive into linear algebra, with a special focus on bilinear algebra, the study of geometry from an algebraic viewpoint. It covers the structure theory of operators, their determinants, eigenvalues and eigenvectors, and their importance for analysis, computational methods and machine learning.

A large section of the course is about scalar products and their generalizations, which provide a formalism for various quadratic phenomena and similarity measures. The study of the structure-preserving mappings (rotations, reflections and, more generally, orthogonal and unitary operators) builds a basis for numerous extremely important applications: singular value decomposition, low-rank approximations and principal component analysis, polar and QR decompositions, Procrustes problem, the least squares methods and beyond.

The course culminates in the introduction and study of quaternions, an extension of the complex numbers used, for example, in 3D graphics.

Capstone Project

- By using the spectral properties of the graph-associated matrices, the students will implement one of the two basic approaches to automated graph drawing.
- By using the tools from bilinear algebra, the students will implement the data points alignment routine.

Andrei Smolensky

Prerequisites

- Algebra 1
- Discrete Mathematics 1
- Mathematical Analysis 1

Where you will need it

- For successful passing of the following courses: Mathematical analysis 2, Algorithms 1, Mathematical statistics, Machine learning

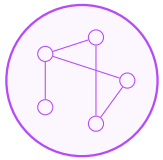
Syllabus

- Operators: determinants, eigenvalues and eigenvectors, canonical forms
- Bilinear algebra: bilinear and quadratic forms, diagonalization
- Analytic geometry: Euclidean and Hermitian spaces, projections, reflections, orthogonalization
- Normal operators: orthogonal and unitary matrices, Hermitian matrices, eigendecomposition
- Decompositions and applications: SVD, QR and polar decompositions, PCA, least squares, low-rank approximations

You will learn

- To calculate determinants and use them
- To interpret geometric problems in terms of linear algebra
- To apply the tools of bilinear algebra to data analysis

Discrete Mathematics 2: Probability and Graphs



2nd semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

The main goal of the course is to learn how to design and analyze discrete mathematical models in various areas of computer science. We will focus on formal proofs of correctness: will cover various proof strategies, will see how an intuitively correct proof may go wrong (which can lead to a failure of software or hardware in practice). Being able to distinguish between correct and incorrect proofs becomes more and more important in the era of large language models. The course is equipped with a large number of exercises, both theoretical and practical. Using theoretical exercises, the students will practice solving problems and writing down formal proofs. Using programming exercises, the students will see a rich variety of practical applications where discrete mathematical ideas arise and will practice implementing (as well as testing and debugging) efficient solutions.

In the second part of the course, we will learn the basics of discrete probability theory and graph theory.

Alexander Kulikov

Ivan Mihajlin

Prerequisites

- Proofs, logic, combinatorics (Discrete Mathematics 1)

Where you will need it

- For successful passing of the following courses: Algorithms, Probability Theory, Theoretical Computer Science, Mathematical Statistics, Fundamentals of Machine learning.

Syllabus

- Discrete probability theory
  - Events: distributions; probability theory and combinatorics; process tree; birthday paradox; computing probabilities recursively.
  - Independency: fractions; conditional probabilities; independent events; Bayes' formula; Lovász Local Lemma.
  - Random variables: independent random variables; distributions; expectations; geometric distribution; Poisson distribution.
  - Deviations: Markov inequality; variance; Chebyshev inequality; law of large numbers; central limit theorem; sampling method; Chernoff inequality.
  - Probabilistic method: tournament paradox; Ramsey numbers; sum-free sets; maximum satisfiability.
  - Probability in Computer Science: algorithms; verification; comparing files over a network; confidential computation; approximation algorithms; universal hashing; derandomization; universal hash function; error ammplification.
- Graph theory
  - Trees: DFS and BFS trees; minimum spanning tree; Cayley's formula; matrix-tree theorem.
  - Cycles: topological ordering; strongly connected components; Eulerian and Hamiltonian cycles; traveling salesman problem; de Bruijn graphs and genome assembly.
  - Flows: connectivity; flows; cuts; bipartite matchings; project selection; image segmentation.
  - Matchings: independent sets and coverings; bipartite graphs; vertex covers; stable matchings.
  - Colorings: greedy coloring; degrees and cliques; nonlocality of chromatic number; chromatic polynomial; coloring algorithms.
  - Planarity: Euler's formula; nonplanar graphs; crossing number; planar graph coloring; special layouts.

You will learn

- To compute probabilities of basic events as well as expectations and deviations of random variables
- To model various problems as graphs and to apply known graph tools for solving them

Algorithms 1: Basic Toolbox

2nd semester
6 ECTS, 3 hours per week
homework, final exam

Abstract

This course provides an introduction to basic techniques used for algorithm design. It covers the common algorithms, algorithmic paradigms, and data structures. The course emphasizes the relationship between algorithms and programming, and introduces basic performance measures and analysis techniques for these problems.

The course is equipped with a large number of exercises, both theoretical and practical. Using theoretical exercises, the students will practice solving problems and presenting their solutions. Using programming exercises, the students will practice implementing comlicated algorithms.

Pavel
Mavrin

Prerequisites

- Mathematics: proofs, functions (logarithm, polynomial, exponent)
- Programming: Python basics (input/output, loops, recursion)

Where you will need it

- For designing and analyzing efficient algorithms and programs
- For successful passing of the following courses: Theoretical Computer Science, Machine Learning

Syllabus

- Algorithms and complexity analysis
  - Algorithms, complexity, asymptotics.
  - Randomized algorithms.
  - Amortized analysis
- Sorting algorithms
  - Merge sort
  - Heap sort
  - Quick sort
  - Lower bound for sorting algorithms
- Basic technics
  - Greedy algorithms
  - Dynamic programming
  - Divide and conquer
- Data structures
  - Binary heap
  - Stack, queue
  - Segment tree
  - Binary search tree
  - Hash table

You will learn

- To estimate the running time of algorithms and programs
- To describe and use basic algorithms and data structures
- To recognize problems that can be solved using techniques learned in this cours
- To design new algorithms for similar problems

Programming Paradigms

2th semester
6 ECTS, 3 hours per week
assignments, midterm, final exam

Abstract

Unlock the diverse world of programming paradigms with our comprehensive course, which offers a deep dive into various programming approaches, including imperative, declarative, procedural, functional, and logic programming, along with object-oriented programming (OOP).

You'll gain hands-on experience with languages like `Python`, `Haskell`, `Prolog`, `Java`, and `C++`, exploring their unique features and applications.

This course is perfect for aspiring programmers and software developers looking to broaden their programming knowledge and skills across different paradigms.

Capstone Project

Interpreter for a small language. Students need to complete and extend implementation of an interpreter for a small language for two-dimensional geometry objects. An implementation in `Python` is mostly completed, and they should choose any other functional programming language (`Haskell`, `Lisp`, `Scala`, `SML`, `Racket`, ...) and rewrite the code in functional paradigm.

Prolog project Students will develop a project in the `Prolog` language that they come up with themselves or choose from a suggested list. It can be lowest common ancestor algorithm, optimal team builder or Sudoku game, everyone can choose a project that interests them and implement it in a logic programming.

Audio Processing System. Create a versatile audio processing application in `Java` or `Kotlin` that can handle various transformations on audio streams, such as volume adjustment, speed modification, noise removal, and channel manipulation (mono/stereo).


Alexander Avdiushenko


Tatiana Berlenko

Prerequisites

- Basic math concepts (Mathematical Analysis 1, Algebra, Discrete Mathematics 1)
- Programming: basic knowledge of some programming language: input-output, loops, recursion, classes, functions (Python, Programming Basics)

Where you will need it

- More specialized software engineering courses
- Working as Software Developer, ML Engineer, Data Scientist

Syllabus

- Programming Paradigms: An overview of imperative, declarative, and procedural programming
- Procedural Programming: Learn the basics with languages like `Python` and `C`, and apply your knowledge through practical exercises
- Functional Programming: functions, recursion, immutability, with practical examples in `Python` and `Haskell`
- Logic Programming: Introduction to `Prolog` and its comparison with imperative programming
- Object-Oriented Programming (OOP): Understand OOP principles, objects, classes, inheritance, and polymorphism with practical sessions in `Java`, `C++`, and `Python`
- Programming Patterns: Learn and practice various programming patterns and antipatterns

You will learn

- To write programs using different programming paradigms
- To apply functional programming concepts in `Python` and `Haskell`
- To understand logic programming with `Prolog`
- To design and implement object-oriented programs in `Java` and `Python`
- To recognize and apply programming patterns and antipatterns
- To showcase your understanding through project defense sessions, receiving feedback to refine your skills

Mathematical Analysis 3

3rd semester
6 ECTS, 3 hours per week
exam

Abstract

In the third part of the Calculus course, students study two big topics: measure theory (multidimensional integration), and functional series, including Fourier series and Fourier transform. Measure theory extends the study of integrals initiated in Calculus-2, and are needed primarily in the courses of Probability Theory, Statistics and Machine Learning. The basic mathematical notions studied in this part of course are the differential of a smooth map and the Jacobian determinant of the coordinate change. The most important result is the method for changing coordinates in a multiple integral — it allows to simplify many practical calculations. The second part of the course, the functional series and Fourier analysis. This part of the course is heavily based on linear algebra, in particular, the notion of the Hilbert space naturally extends that of a (finite dimensional) Euclidean space. Some practical applications of Fourier analysis will be discussed in the course.

Fedor
Bakharev

Prerequisites

- Mathematical analysis 1 and 2.
- Algebra 1.
- Discrete mathematics 1

Where you will need it

- In all subsequent mathematical disciplines of an analytical nature
- To understand the courses: Probability Theory, Mathematical Statistics, Machine Learning
- To create a common mathematical culture

Syllabus

- Abstract measure theory and Lebesgue measure in $\mathbb{R}^n$
- Munipulations with integral, sums, derivatives and limits
- Most important theorems from Measure theory
- Change of variables in a multiple integrals and applications
- Hilbert space, orthogonal systems. Functional spaces
- Fourier decomposition of a periodic function
- Fourier transform

You will learn

- To understand the basic constructions of the measure theory and multiple integration
- To know the values of basic integrals used in probability theory and the derivation of the corresponding formulae
- To understand the scalar product on functions and how to use it in optimization problems
- To understand the basic of the Fourier analysis and its applications

Computer Architecture

3rd semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

This course provides an in-depth exploration of the fundamental principles and design methodologies of modern computer systems.The discipline introduces students to the architecture of the computer at multiple levels. Starting with electrical signals and the physical structure of the computer's main components, the course material gradually dives into the mechanisms of the processor, memory, and communication between electronic components. It is based on the RISC-V instruction set architecture, providing a comprehensive understanding of assembly language programming. It covers the creation of specific hardware components and related assembly instructions, essential for the development and optimization of various applications, including those required for embodied AI.

Capstone Project

Traffic Lights Controller. You will develop a finite state machine that manages the timing and sequencing of red, yellow, and green lights for multiple directions, creating the circuit using Verilog and simulating it in a digital design environment. The final implementation may be synthesized and tested on an FPGA board to demonstrate real-world functionality.
QEMU Target for Custom Extension RISC-V Processor. You will modify the QEMU source code to implement new instructions and architectural features, creating a software model of your custom RISC-V variant. The final result will be a functional QEMU emulator capable of running and testing software written for your extended RISC-V architecture.

Kirill
Krinkin

Prerequisites

- High school level of physics
- Programming basics with C

Where you will need it

- To study Operating systems development
- To work with modern hardware
- To develop own processing units
- To work in the area of embodied AI and Cyber-physical systems
- To make projects with robots in SmartCity Lab and Industrial robot Lab

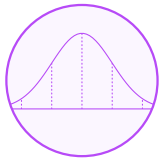
Syllabus

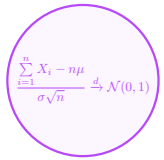
- Combinational and Sequential logic.
- RISC-V instruction set architecture introduction: base and optional extensions
- ALU, memory organization
- Processor instructions encoding and their types
- Single-cycle processor: data path, control unit
- Multi-cycle processor: data path, control unit
- Pipelined processor, Pipeline hazards
- Hardware design with Verilog
- RISC-V Privileged Architecture
- ELF structure, application binary inteface

You will learn

- How to design custom processors and other hardware
- How to programm RISC-V with assembly language
- To make programms on the lowest level
- To support hardware on operating system level

Probability Theory





3rd semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

The course is dedicated to the study of randomness in the continuous setting, which requires, as compared to the discrete case, a new language of probability densities and cumulative distributions functions. We will study the notion of a random variable, the properties and structure of various distributions and where they appear, and their important characteristics such as expected value and variance.

Probability theory lays the foundations of mathematical statistics, thus we will also study basic results which find their application there, such as the law of large numbers and the central limit theorem, and several probabilistic notions of convergence.

A part of the course will be dedicated to the study of random processes, including Markov chains (used, among many other things, for random sampling and as models for reinforcement learning), Gaussian processes (used in Bayesian inference) and martingales (used, for example, in the analysis of performance and convergence of online and reinforcement learning).

During the course we will study how various random phenomena can be modeled on a computer, including the efficient sampling from probability distributions.

Andrei
Smolensky

Prerequisites

- Mathematical analysis 2
- Discrete mathematics 2

Where you will need it

- For implementing any system including a non-trivial use of randomness
- For successful passing of the following courses: Mathematical statistics, Machine learning

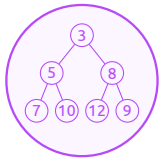
Syllabus

- PDF and CDF, important continuous distributions
- Markov, Chebyshev and Jensen inequalities
- Convergence: almost sure, in probability, in distribution; LLN and CLT
- Random processes
- Bayesian inference

You will learn

- To understand and analyze continuous probability distributions and their properties
- To apply the law of large numbers and the central limit theorem
- To apply concepts from information theory, such as entropy and mutual information, to measure uncertainty and information content.
- To explore and implement basic applications of continuous probability in machine learning

Algorithms 2: Data Structures



3rd semester
6 ECTS, 3 hours per week
homeworks, midterm, final exam

Abstract

This course provides an in-depth understanding of some of the fundamental algorithms and data structures from various areas of computer science. We will cover some advanced algorithmic ideas (both classical and recent), and the theory behind it (theorems, proofs).

The course is equipped with a large number of exercises, both theoretical and practical. Using theoretical exercises, the students will practice solving problems and presenting their solutions. Using programming exercises, the students will practice implementing complicated algorithms.

Pavel
Mavrin

Prerequisites

- Mathematics: proofs, functions (logarithm, polynomial, exponent)
- Programming: Python basics (input/output, loops, recursion)

Where you will need it

- For designing and analyzing efficient algorithms and programs
- For successful passing of the following courses: Theoretical Computer Science, Machine Learning.

Syllabus

- Graph algorithms
  - Binary lifting method
  - Depth-first search
  - Biconnectivity. Strong connectivity
  - 2-SAT problem
  - Minimal spanning tree
  - Shortest paths
  - Bipartite matchings
  - Graphs games analysis
- String algorithms
  - Substring search problem
  - Aho-Corasick algorithm
- Math algorithms
  - Number theory algorithms
  - Cryptography algorithms
  - Fast Fourier transformation

You will learn

- To describe and use fundamental algorithms
- To explain the mathematical concepts needed for the analysis of the behavior of algorithms
- To recognize problems that can be solved using techniques learned in this course
- To design new algorithms for similar problems

Fundamentals of Machine Learning

3th semester
6 ECTS, 3 hours per week
assignments, midterm, final exam

Abstract

The aim of the course is to demystify AI. This comprehensive course delves into both basic and advanced concepts of machine learning and deep learning. Starting with Python, Pandas and fundamental machine learning techniques, students will explore statistical learning models, linear and logistic regression, decision trees, and ensemble methods.

The journey continues into the realm of deep learning, covering multilayer neural networks, stochastic gradient descent, backpropagation, optimization techniques, and more. A little bit more advanced topics such as recurrent, convolutional neural networks, and general pre-trained transformers (GPT) will also be addressed.

Practical coursework, including programming assignments and hands-on projects, will be primarily based on Python and PyTorch.

Capstone Project

Two-layer neural network. In this course, you will initially write a simple two-layer fully connected neural network using only Python and NumPy. This will allow you to thoroughly understand the backpropagation algorithm and the stochastic gradient descent.

GPT-2 from scratch. Afterwards, using PyTorch, you will implement the GPT-2 decoder from OpenAI, based on the attention mechanism in the transformer architecture. Once trained, this will enable you to generate character-by-character texts of your favorite author, Linux kernel-like code, or plausible mathematical theorems in LaTeX.

Alexander
Avdiushenko

Prerequisites

- Vectors and matrices, SVD (Algebra)
- Bayes' theorem (Discrete Mathematics)
- Mean and variance (Probability theory)
- Differentiation (Mathematical Analysis 1), gradient descent (Mathematical Analysis 2)
- Programming: basic knowledge of the programming language `Python` (input-output, loops, recursion, classes, functions)

Where you will need it

- More specialized machine learning courses
- Working as Data Scientist, ML Engineer, ML Researcher

Syllabus

- Machine Learning task examples, quality evaluation
- Matrix differentiation, chain rule
- Linear methods of classification and regression
- Stochastic gradient descent
- Decision trees, ensembles, gradient boosting
- Intro to neural networks, Backpropagation, Multi-Layer Perceptron (MLP)
- Intro to language modelling: bigrams
- Activations, BatchNorm
- WaveNet, convolutions
- Recurrent Neural Networks (RNN), attention, transformers

You will learn

- To navigate the vast field of machine learning
- To understand fundamental principles of the various methods
- To formalize tasks in the model learning language
- The advantages and disadvantages of neural networks
- To implement basic models in the programming language `Python`
- Problems that arise when training models

Human Computer Interaction

4th semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

UX/UI designers are the heroes of a profound transformation. Their work turned personal computers into today's wildly successful mobile devices, enabling users to communicate and collaborate in remarkable ways. The desktop applications that once served the needs of professionals have increasingly given way to powerful social and AI tools that deliver compelling user experiences to global communities. Researchers created the interdisciplinary design science of Human-Computer Interaction by applying the methods of experimental psychology to the powerful tools of computer science. They then integrated lessons from educational and industrial psychologists, instructional and graphic designers, technical writers, experts in human factors or ergonomics, and growing teams of anthropologists and sociologists.

This course takes a holistic approach to Human Computer Interaction and follows a human centered design approach to understanding, designing and evaluating digital experiences. We will learn how to design intuitive systems and interfaces that are elegant, effective, and ethical. Fundamental aspects of human physiology and psychology are introduced, and key features of interaction and common interaction styles are delineated. Interaction design, cognition, emotional and persuasive design, theory and practice of interface design and AI design, user experience, prototyping, evaluation of user interfaces and emerging HCI themes all form part of this course.

Avgousta Kyriakidou

Prerequisites

- None required.

Where you will need it

- This course equips you with cutting-edge skills, seamlessly integrating AI tools into the design process of various systems, while also learning the essentials of designing for AI.
- Having skills in HCI not only enhances your ability to create user-friendly, effective, and ethical products but also positions you as a valuable asset in the job market, capable of addressing the complex challenges of designing technology that meets human needs.

Syllabus

- Introduction to HCI, Interaction design, User experience design. Purpose and use.Design in the age of AI.
- Understanding and conceptualizing interaction.
- HCI Design and cognition – evaluation based on cognitive modeling. Persuasive design and emotional interaction.
- The Design Process; User Centered Design; 4 Pillars of Design, Design Rules, Guidelines and Standards.
- User research, Task analysis; Scenarios and user personas. How to use AI tools for research and ideation.
- Ideation and Prototyping. How to use AI tools for prototyping and testing.
- Input methods: Menu selection, Form Fill-In and Dialogue Boxes. Graphic design: Typography and colour theory; Design Patterns and Metaphors.
- Principles and best practices of designing for the small screen (mob applications)
- How to design the future: Essential factors for AI-powered products
- Evaluation methods, usability, accessibility, trust and ethics
- Coursework project feedback sessions, Final oral coursework demonstrations

You will learn

- Key techniques from HCI, interaction design, and UX design used for understanding, designing, and evaluating interactive products, services, and experiences.
- Psychological foundations of HCI and interaction design, including cognition, emotional design, and persuasive design principles.
- Challenges in designing user experiences, covering the main components of UX, essential features of the design process, and their application across various system types.
- Integrating AI into the design process, learning to work harmoniously with AI by utilizing tools for in-depth UX research, including market analysis, persona development, data processing, AI-powered prototyping, visual design, and UX writing to enhance problem-solving capabilities.
- Human-centered design in the AI era.

Algorithm Engineering

3th semester
6 ECTS, 3 hours per week
project work, midterm, final exam

Abstract

This course invites you to immerse yourself in the world of algorithm engineering through a dynamic, project-based approach and software engineering way. In a flipped classroom setting, you'll collaborate with your teammates to design and develop a custom framework featuring a suite of algorithms capable of processing diverse data types such as images, text, graphs, and audio.

You'll also have the opportunity to create forks of well-known open-source libraries, applying your knowledge to real-world challenges.

Through engaging case studies, we'll explore how to effectively analyze requirements, design robust algorithms, and implement them as high-quality code. You'll learn to optimize and test your solutions, ultimately building reliable software libraries that can be deployed and monitored in production environments. This course is about more than just learning—it's about applying your skills to create impactful, real-world solutions.

Dmitry
Botov

Prerequisites

- Data structures, Algorithms and complexity analysis (Algorithms 1)
- Object-Oriented Programming (OOP), Programming Patterns (Programming Paradigms)
- Software Engineering Basics and Programming: basic knowledge of some programming language: (Python, Programming Basics)

Where you will need it

- In Advanced Algorithm Design and Development: To create efficient and scalable algorithms in various technical domains.
- For Building and Maintaining Software Systems: To integrate algorithms within larger software systems, ensuring their reliability and maintainability.
- In Data Science and Machine Learning Projects: To design, implement, and optimize machine learning algorithms and data processing pipelines.
- Collaborative Engineering Projects: To work effectively in Agile teams, contributing to the development of complex algorithmic solutions.
- For Contributing to Open Source and Industry Frameworks: To develop and enhance algorithmic frameworks used in open-source projects and industry applications.

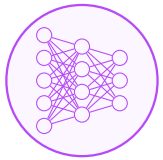
Syllabus

- Introduction to Algorithm Engineering: Methods and Techniques in Systems Engineering
- Software Engineering Principles for Algorithms
- Requirements Engineering for Algorithm Development
- Analysis of Algorithmic Frameworks (e.g., Machine Learning and Data Processing Frameworks)
- Design of Algorithm Frameworks
- Algorithm Evaluation: Metrics and Methods
- Test-Driven Development (TDD) for Algorithm Frameworks
- Deployment of Algorithms: Ensuring Reliability and Maintainability
- Optimization Techniques: System Performance Monitoring and Management
- Final Project Presentation

You will learn

- To understand the fundamentals of requirements engineering, algorithm frameworks, and design principles
- To develop algorithms collaboratively within an engineering team using Agile methodology
- To apply various software engineering and DevOps tools and practices
- To evaluate, optimize, and refine algorithm implementations
- To create and deploy custom algorithm frameworks, including contributions to open-source projects

Advanced Machine Learning



4th semester
6 ECTS, 3 hours per week
assignments, midterm, final exam

Abstract

Explore the depths of semi- and unsupervised learning, mastering clustering and the Expectation-Minimization algorithm through hands-on practice.

Learn reinforcement learning and know the difference between AlphaGo and AlphaZero, explore generative models like VAEs and GANs. Delve into large language models (LLMs) with a focus on tokenization, embeddings, and the new paradigm of prompt-based programming. Address LLM challenges, from hallucinations to multi-agent systems, and understand the economics of optimization.

This course is designed for those looking to elevate their machine learning skills and stay at the forefront of technological advancements.

Capstone Project

AlphaZero-like players. AlphaZero, developed by DeepMind, represents a groundbreaking approach in the field of AI, capable of mastering complex games like Chess, Shogi, and Go through self-play and deep reinforcement learning.
This project aims to introduce you to the core principles behind AlphaZero's architecture and learning processes by applying these concepts to simpler board games like TicTacToe, Connect Four, and Dots and Boxes.
Build Your Own Jarvis. Design and implement a personalized AI assistant that can understand and respond to voice commands on your laptop. By assembling various parts such as voice recognition, handling dialogue with LLM, voice generation and making it all work together seamless via scripting in Python. Your main challenge will focus on optimizing for performance and determining which features to prioritize for speed.

Alexander
Avdiushenko

Prerequisites

- Fundamentals of Machine Learning (ML)
- Bayes' theorem (Discrete Mathematics)
- Mean and variance (Probability theory)
- Programming: basic knowledge of the programming language Python (input-output, loops, recursion, classes, functions)

Where you will need it

- Even more specialized machine learning courses
- Working as Data Scientist, ML Engineer, ML Researcher

Syllabus

- Clustering and EM Algorithm: Theory and practical applications
- Reinforcement Learning: From foundational principles to advanced applications like AlphaZero
- Generative Models: VAEs, GANs, CLIP, flows, and diffusion models with practical sessions
- Large Language Models: Tokenization, embeddings, and advanced prompting techniques
- LLM Challenges: Hallucinations, multi-agent systems, and optimization economics

You will learn

- To navigate the vast field of machine learning
- To gain hands-on experience with clustering and generative models
- To understand and tackle LLM-specific challenges
- Enhance your knowledge in both theoretical and practical aspects of AI

Mathematical statistics

4th semester
6 ECTS, 3 hours per week
homework, midterm, exam

Abstract

Mathematical statistics is a branch of mathematics that deals with the description and analysis of data for the construction of probabilistic models of random phenomena that gave rise to these data. The described intermediate position between real observations and abstract probabilistic models makes mathematical statistics the main tool for solving applied problems, in which uncertainty is interpreted as randomness. In this course, we will get acquainted with both classical results in this area, and with more modern computational methods.

Practical classes will be conducted on Python, on which we will apply the acquired knowledge to real data and investigate the limits of applicability of the studied methods.

Andrei
Smolensky

Prerequisites

- Probability Theory: fundamentals, random variables, limit theorems
- Mathematical analysis 3: measure theory, integration
- Algebra: linear algebra, orthogonality
- Programming: basic knowledge of the programming language Python

Where you will need it

- To understand the course Machine Learning
- At work in any position related to with forecasting and analytics
- In life for a more critical and meaningful perception of information

Syllabus

- Fundamentals of mathematical statistics: descriptive statistics, parameter estimation, confidence intervals, hypothesis testing, goodness of fit and homogeneity tests, linear regression
- Monte Carlo method, resampling methods
- Fundamentals of the Bayesian approach, Bayesian classification

You will learn

- visualize data for preliminary analysis
- evaluate unknown parameters of the model under training and determine the quality of the assessments received
- formulate problems in terms of statistical hypotheses and test them
- use bootstrap for parameter estimation and hypothesis testing
- work with linear models, determine their quality and use them for forecasting
- use the packages `scipy.stats`, `pandas`, `matplotlib`, `sklearn`, etc. to solve the problems described above on real data

Robotics and Computer Vision

5th semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

The project-oriented course “Robotics and Computer Vision” aims to explore the environment techniques and tools for programming robots, encompassing embedded programming, middleware such as ROS2, and a variety of libraries that assist in creating robot control algorithms. It also aims to apply knowledge in programming, machine learning, computer vision to mobile robotics. Students will have a brief introduction to control theory, signal processing, sensor fusion. Over the semester, students are expected to complete a project related to implementing autonomous robot behavior (project development is the main form of study of this course). Participants will be able to work with real robot hardware in the new SmartCityLab.

Kirill
Krinkin

Prerequisites

- Programming basics with C
- Computer Architecture
- Linear Algebra
- Operating Systems

Where you will need it

- To work with modern robotics equipment
- To be able to create embodied autonomous agents
- To make projects with robots in SmartCity Lab and Industrial robot Lab

Syllabus

- Introduction into Robotica
- Robot Operating System
- Introduction into Embedded Programming
- Control theory fundamentals
- Computer vision and applications with OpenCV
- Structure from Motion Algorithms
- Simultaneous Localization and Mapping
- Navigation for Mobile Robots
- GPU Programming fundamentals
- TPU programming fundamentals

You will learn

- To design robot from different perspectives (hardware, system and applied software)
- To implement autonomous robot behaviors
- To implement multi-robot distributed systems
- To create algorithms for robot perception, cognition and planning

Agile Scrum for AI Development

5th semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

Scrum stands out as the predominant agile framework widely employed in diverse sectors, spanning from web development and computer games to robotics and Artificial Intelligence. Its versatility extends to the management and control of complex and contemporary development projects using a range of iterative techniques. This course delves deeply into the details of Scrum, encompassing both theoretical foundations and practical applications, while also addressing potential limitations. Students will experience applying SCRUM in real-life AI development projects and thus acquire the essential knowledge to adeptly implement Scrum in real-world development scenarios. The course thoroughly explores team roles, activities, backlogs, sprints, meetings, and other integral elements of Scrum, providing detailed explanations, and practical applications as well as enhancing students' collaboration and critical analysis skills.

Avgousta Kyriakidou

Prerequisites

- None required

Where you will need it

- Mastery of Scrum practices enhances career prospects in AI and software development, particularly in environments focused on continuous delivery and rapid iterations.
- The course's focus on managing uncertainties and balancing speed with quality in Agile environments will be invaluable for students in real-world scenarios where they must adapt quickly to changes and unforeseen challenges.
- The course will strengthen students' collaboration, problem-solving and decision-making skills, essential for both technical and leadership roles.

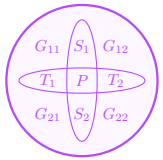
Syllabus

- Introduction to Agile methodology in the age of AI. Overview of agile principles, comparison with traditional project management approaches, need for adaptability in AI projects
- Understanding the AI development lifecycle, key considerations at each stage in an agile context.
- Introduction to the SCRUM framework, artifacts, values, roles, meetings and more, adapting SCRUM for AI.
- Forming Agile AI teams–Scrum Roles and responsibilities, cross-functional, self-organising, team dynamics and collaboration.
- Agile requirements engineering–Product backlog in AI projects, Product Vision and Product Backlog, User Stories, MOSCOW and Definition of Done. Breaking down Epic user stories, planning poker estimation, MVP, Release planning. Sprint Planning for AI, Breaking down user stories into tasks and into sprint-sized increments, The Sprint goal, Sprint Backlog Adapting sprint planning for evolving AI requirements.
- Conducting AI Sprints, Implementing iterative model development, Agile practices for coding, testing, and integration, Intro to XP.
- Lego4Scrum Hackathon – Industry well known based technique to apply Scrum in practice.
- Coursework project feedback sessions, Final oral coursework demonstrations.

You will learn

- The core principles of Agile methodology and the Scrum framework, including roles, artifacts, and meetings

Advanced Graph Theory



5-8 semesters, elective
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

This course delves into the advanced concepts of graph theory, building on the foundations laid in Discrete Mathematics courses. Emphasis will be placed more on theoretical understanding. If time permits, additional advanced topics like maximum flow algorithms, graph expanders, and Yeo and Kotzig's theorems will be discussed upon request.

Nikolai Chukhin

Prerequisites

- Proofs, logic, combinatorics (Discrete Mathematics 1)
- Fundamentals of Graph Theory (Discrete Mathematics 2)

Where you will need it

- Essential for theoretical research in mathematics and computer science
- Crucial for developing algorithms and solving complex problems in topics related to graph theory
- Provides foundational knowledge for postgraduate studies in mathematics and theoretical computer science

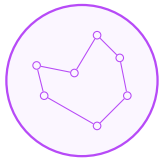
Syllabus

- Paths and Cycles. Theorems: Ore's, Dirac's, Pósa's, Tutte's and Chvátal's Closure
- Matchings. Theorems: Gallai's, Berge's and Kőnig's, Tutte's, Petersen's for Cubic Graphs, Plummer's for Regular Graphs, Petersen's 2-Factor, Lovász's and Berge's Formula for Graph Deficiency
- Connectivity. Graph Block Decomposition, Theorems: Menger's, Dirac's, Mader's, Halin's for Triangles
- Colorings. Theorems: Vizing's, Brooks', Borodin's, Gallai's for k-Critical Graphs, Dirac's, Vizing's, Gupta's
- Planar Graphs. Cyclic Traversal of Boundaries, Articulation Points of Face Boundaries, Graph Triangulation, Theorems: Kuratowski's, Wagner's, Golovina-Yaglom on 3-Coloring Triangulations and Dual Graphs, Tait's Colorings, Thomassen's Theorem on 5-List-Coloring Planar Graphs
- Directed Graphs. Strongly Connected Components, Existence of Hamiltonian Cycle in Directed Graphs, Hamiltonian Path and Cycle in Strongly Connected Tournaments, Theorems: Rédei's on Hamiltonian Paths in Tournaments, Chvátal-Lovász, Gallai-Milgram, Dilworth's, Roy-Gallai, Galvin's, Alon-Tarsi
- Networks and Flows. Menger's Theorem, Maximum Flow in Arbitrary Networks, Dinic's algorithm
- Extremal Graph Theory. Ramsey Numbers for Trees and Complete Graphs, Edge Count in Graphs without $K_{m,n}$ and Theorems of Turán for Graphs without K_n, Bondy-Simonovits for Even Cycles and Induced Ramsey for Bipartite and General Graphs
- Spanning Trees. Number of Spanning Trees, Prüfer Code, Theorems: Cayley's, Bowler-Carmesin on Covering and Packing, Nash-Williams

You will learn

- Analyze and solve advanced graph theoretical problems
- Conduct rigorous proofs and logical reasoning in graph theory

Algorithms for NP-hard Problems



elective (5–8 semesters)
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

Many computational problems arising in practice are NP-hard meaning that it is unlikely that they can be solved efficiently. In this course, we consider three popular approaches for dealing with such problems from a theoretical point of view (as we are interested in provable guarantees):

- Exact algorithms: If we cannot solve a problem in polynomial time, can we at least solve the problem faster than by the brute force approach?
- Approximation algorithms: Can we find a solution that is provably not much worse than an optimum one?
- Parameterized algorithms: Can we design algorithms that are efficient when a certain parameter of the input is small enough?

Alexander Kulikov

Prerequisites

- Discrete probability (Discrete Mathematics 2)
- Basic algorithms (Algorithms 1 and 2)

Where you will need it

- For designing algorithms for hard problems.
- For doing research in Theoretical Computer Science.

Syllabus

- Course overview. Algorithms for the Vertex Cover problem: $O^*(1.619^n)$ exact algorithm, 2-approximation algorithm based on inclusion-wise maximal matching, parameterized algorithms.
- Kernels for the Vertex Cover problem.
- Kernel of size $2k$ for the Vertex Cover problem. Exact algorithms for the Hamiltonian Path problem: $O^*(2^n)$ time and exponential space using dynamic programming, $O^*(2^n)$ and polynomial space using the inclusion-exclusion formula.
- $O^*(2^n)$ time and exponential space algorithm for the Hamiltonian Path problem using FFT. Color coding technique for the k-path problem. Multivariate polynomials over finite fields.
- Algebraic algorithms for the k-path problem.
- Approximation algorithms for TSP: inapproximability of the general case, 1.5-approximation for the Metric TSP, $2/3$-approximation for MAX-ATSP.
- 4- and $2\frac{2}{3}$-approximation for the Shortest Common Superstring problem.
- Approximation scheme for Euclidean TSP
- $O^*(2^{\frac{\omega n}{3}})$ exact algorithms for MAX-CUT and MAX-2-SAT via fast matrix multiplication
- 0.878-approximation algorithm for MAX-CUT and MAX-2-SAT via semidefinite programming
- Exact algorithms for the graph coloring problem: $O^*(1.5^n)$ randomized algorithm for 3-coloring via a reduction to 2-SAT, $O^*(3^n)$ for chromatic number using dynamic programming, $O^*(2^n)$ for chromatic number using inclusion-exclusion
- Exact algorithms for 3-SAT: $O^*(2^{2n/3})$ algorithm for Unique 3-SAT based on branching w.r.t. a random permutation, $O^*(1.5^n)$ algorithm based on local search, $O^*((4/3)^n)$ algorithm based on random walks

You will learn

- To design approximation, exact, and parameterized algorithms.
- To distinguish between easy and hard computational problems.
- To implement algorithms for NP-hard problems.

Complexity Theory

elective (5–8 semesters)
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

In this class, we will try to understand the idea of computation itself. We will start by learning the necessary language and showing that there are problems that we can not solve even with unlimited resources. Then we will dive into the intricate world of the complexity classes.

Ivan
Mihajlin

Prerequisites

- Discrete probability (Discrete Mathematics 2)
- Basic algorithms (Algorithms 1 and 2)

Where you will need it

- For distingushing between computationally hard and easy problems
- For doing research in Theoretical Computer Science
- For getting a million dollar prize for solving the P vs NP problem

Syllabus

- Computability and uncomputability
- P vs NP
- Polinomial hierarchy
- Space complexity
- Randomized computation
- Complexity of boolean circuits
- Can an untrustworthy magician with infinite knowledge be useful?

You will learn

- The language of modern theoretical computer science
- To understand the complexity of problems through different lenses
- To navigate the web of complexity classes
- Cool staff

AI-Enhanced Cybersecurity

7th semester
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

This course addresses the fundamental concepts and principles of cybersecurity. It aims to provide students with the knowledge and skills necessary to protect and defend information systems, ensuring their integrity, authentication, confidentiality, and availability. Students will learn about cyber threats and attacks, defense mechanisms, and the legal and ethical issues related to cybersecurity.

Lefteris
Zacharioudakis

Prerequisites

- Basic programming skills

Where you will need it

- In subsequent courses on advanced cybersecurity topics.
- For professional roles in cybersecurity and information assurance.

Syllabus

- Introduction to Information Security, security concepts, security strategy.
- Attacks and Threats, types of malicious attacks, advanced persistent threats.
- Denial of Service Attacks
- Intrusion Detection and Prevention Systems, Scanning and Analysis Tools, Honeypots, Honeynets, and Padded Cell Systems.
- Access Control Fundamentals, Protecting Remote Connections, Firewalls.
- Basic Cryptography – Symmetric, Foundations of Cryptology, Cipher Methods, Cryptographic Algorithms, Cryptographic Tools.
- Basic Cryptography – Asymmetric Cryptography.
- Network Security Protocols, SSL, TLS, HTTPS.
- Risk Management, Risk Identification, Risk Assessment, Risk Control.
- Incident Response.
- Introduction to Forensics.

You will learn

- To identify and analyze cyber threats and vulnerabilities.
- To design and implement appropriate security measures.
- To understand the ethical and legal implications of cybersecurity.
- To apply critical thinking and problem-solving skills in real-world scenarios.

Modern Cryptography

5–8 semester (elective)
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

Modern communication is based on cryptography. Everyone needs to keep the information private not only statically but also when it is transferred over communication channels. Each party needs to authenticate the other parties. People prefer to avoid disclosing information about them unless it is absolutely necessary.

How does public-key (and private-key) cryptography work? What do we need to construct a secure protocol? To answer these questions is the purpose of this course.

The crux of the matter is that we cannot base modern cryptography on the traditional complexity-theoretic assumptions (no, **P** $\neq$ **NP** is by far not sufficient). As we all know, practical implementations frequently fall into trapholes; deep understanding of the underlying notions is required to avoid such fuckups. Many people heard about RSA albeit not everyone knows how to use it correctly. Therefore, we study various cryptographic primitives (such as one-way functions, trapdoor functions, bit commitment schemes) and construct useful protocols that use them (digital signatures, public-key encryption, blockchains, etc).

Edward Hirsch

Prerequisites

- Discrete probability, basic randomized algorithms (Discrete Mathematics 2)

Where you will need it

- To get a job in any field requiring the design of authentication, encryption, blockchains protocols.
- To become a computer science professional.

Syllabus

- Protocols and primitives: what Alice and Bob can do against Charlie?
- Background: all material in algorithms and complexity that you missed by chance.
- One-way functions.
- Trapdoor functions.
- Oblivious transfer.
- Cryptographic hash-functions.
- Security basics: indistinguishability and semantic security.
- Public-key encryption.
- Key agreement.
- Private-key encryption.
- Digital signatures.
- Bit commitment (bid in a sealed envelope).
- Secure distributed function evaluation.
- Blockchains.

You will learn

- How to construct secure cryptographic protocols.
- How secure is modern cryptography (spoiler: not much).
- The real meaning of a lot of buzzwords :-)

Introduction to File Systems

5th or 7th semester
6 ECTS, 3 hours per week
homeworks, project, final exam

Abstract

The first goal of this course is to explain how to present a physical storage device (an HDD or an SSD) as a hierarchy of directories and files. However, we will not be content with just any file system implementation. We want file systems that:

1. are reliable and can protect themselves from hardware errors, kernel crashes and other unexpected interruptions,
2. are performant and can make use of modern high-speed devices.

While exploring these two requirements we will come up with techniques that are useful in any scalable database or networking system:

1. Disk-friendly implementations of file lists and indices will lead us to B-trees and LSM trees which are the basis of almost any database system.
2. Flash-friendly IO will lead us to constructing IO queues in such a way as to hide the request latency and to avoid effects like head-of-line blocking. Such queues are the key component of network protocols like HTTP 2 and QUIC.
3. Protecting against unexpected interruptions and crashes will lead us to journaling and designing idempotent FS operations. These are key aspects of database systems and distributed systems that run over unreliable networks (read: the whole of the Internet).
4. If we have enough time we will consider how to scale a storage system and make it distributed. This will lead us to PAXOS and model checking with TLA+ and TLC.

Capstone project

Students will implement a network protocol for an object storage. The protocol will be tuned to maximize the throughput in high-latency networks, and minimize the RAM required on the server side. The project will also touch on the topic of chaos engineering for testing network protocols.

Artem
Anisimov

Prerequisites

- Programming Basics with C
- Computer Architecture

Where you will need it

- To use databases efficiently.
- To implement databases and other storage systems.
- To design and implement Web services and networked applications.
- To design and implement reliable and highly available systems.

Syllabus

- POSIX and Windows APIs for accessing file systems.
- Virtual File System layer in the Linux kernel.
- On-disk layout of Ext4 and NTFS.
- B-trees, LSM-trees, Bloom filters and “the power of 2 choices”.
- Journaling and request idempotency.
- Queue management, avoidance of long tails and head-of-line blocking.
- Erasure codes and RAID.
- Distributed systems and consensus algorithms.

You will learn

- To analyse the performance of applications that work with the storage and the network.
- To use the storage and the network efficiently.
- To implement systems that are resilient in face of network outages, crashes and faulty hardware.
- To use the `Golang` programming language as well as to use the `C` language more efficiently.

Information Theory

$-\sum p(x) \log p(x)$

5–8 semester (elective)
6 ECTS, 3 hours per week
homework, midterm, final exam

Abstract

The course is dedicated to studying approaches to defining the concept of "amount of information". The sequence of topics in this course follow Kolmogorov's classic paper "Three approaches to the quantitative definition of information" (1965).

The course covers three approaches to defining the concept of "amount of information": combinatorial (Hartley's information), probabilistic (Shannon's entropy), and algorithmic (Kolmogorov complexity). The course includes many examples of application of information theory methods in various areas of computer science: cryptography, communication complexity, coding theory, finite automata theory, computational complexity theory, machine learning, etc.

Alexander Smal

Prerequisites

- Proofs, logic, combinatorics (Discrete Mathematics 1)
- Discrete probability theory (Discrete Mathematics 2)

Where you will need it

- For designing and analyzing algorithms.
- For data analysis and machine learning.

Syllabus

- Combinatorial approach: Hartley's formula
- Probabilistic approach: Shannon's entropy
- Information theory in coding theory
- Information theory in cryptography
- Communication complexity and formula complexity
- Algorithmic approach: Kolmogorov complexity
- Applications of Kolmogorov complexity

You will learn

- To compute the amount of information in an answer to a question, in a random variable, in a text, etc.
- To use information-theoretic method in various areas including data analysis and machine learning.

Practical Formal Methods

5–8 semesters (elective)
6 ECTS, November 18–22
homework, research project

Abstract

This short course provides a brief but comprehensive introduction to practical applications of formal methods for specifying and designing software systems.

Formal Methods are an area of Computer Science that studies construction of secure and highly reliable software and hardware. Formal methods are concerned with specifications that are precise and are stated in languages endowed with a formal syntax and semantics (i.e., precise meaning). Formality helps the specification process in at least two ways: (a) it delivers unambiguous, high-quality specifications, and (b) it enables automated tool support. As we will see, formal specification techniques allow for using state-of-the-art tools for automated validation and verification that help software developers make sense of the code, looking for errors in requirements, models, designs, and implementations.

We will study a collection of techniques for formal software development, spanning the whole development process: from high-level semantics that models a system's behavior to verifying that its implementation does what is intended. This study will be done through the use of actual tools supporting these techniques.

Ilya
Sergey

Prerequisites

- Mathematical logic basics (Discrete Mathematics 1: Logic and Combinatorics)
- Basic data structures and algorithms (Algorithms 1: Basic Toolbox)

Where you will need it

- For analysing and understanding complex computations with many moving parts, such as concurrent and distributed algorithms, reactive systems, and communication protocols
- For being able to rigorously state and prove correctness of algorithms and data structures

Syllabus

- State machines and their safety properties
- System specifications using temporal logic of actions
- Fairness and liveness, refinement
- A tool: TLA+
- Basics of Satisfiability Modulo Theories
- A tool: Z3 SMT Solver
- Introduction to deductive program verification and Hoare Logic
- A tool: Dafny Program Verifier

You will learn

- The principes behind machine-assisted formal reasoning
- To express a variety of correctness properties of systems, data structures, and algorithms
- To employ formal verification for ensuring the absence of bugs and vulnerabilities in correctness-critical software

ML System Design

elective (5–8 semesters)
6 ECTS, 3 hours per week
project work, midterm, final exam

Abstract

This course is a hands-on, project-oriented journey into the world of Machine Learning Systems Design, specifically crafted to guide you through the complete lifecycle of developing a real-world machine learning system from start to production deployment.

Throughout the course, you and your teammates will engage in every aspect of building a production-ready ML system— from problem identification, data engineering, and model development, to deployment, monitoring, and maintenance. By the end of the course, you'll not only have a deep understanding of the technical skills required but also the experience of having created a fully operational ML system, ready to deliver value in a production environment.

This course is ideal for those who are eager to move beyond theoretical knowledge and want to develop the practical, collaborative skills needed to succeed in today's fast-paced AI and ML industries.

Dmitry
Botov

Prerequisites

- Fundamentals of Machine Learning, Advanced Machine Learning
- Software Engineering

Where you will need it

- In Advanced Machine Learning Projects: To design, develop, and deploy scalable and reliable machine learning systems that can handle real-world data and applications.
- For Building and Optimizing ML Pipelines: To create end-to-end pipelines for data processing, model training, and deployment, ensuring efficiency and reproducibility.
- In Production-Level AI Solutions: To transition models from research to production, integrating them seamlessly into business operations and products.
- For Ensuring Ethical and Safe AI Deployment: To address ethical considerations, ensuring that your machine learning models are fair, unbiased, and safe in their application.

Syllabus

- Introduction to Machine Learning Systems Design. Production vs. Research
- Fundamentals of ML Systems Design. Lifecycle of an ML System.
- Design Doc for ML System
- Gathering Datasets and Data Annotation
- Data Engineering: data preprocessing, pipeline development, and data versioning.
- Exploratory Data Analysis (EDA) and Visualization
- Experiment Design: validation schemas, metrics, baseline solution
- Model Development: training pipeline and managing experiment versions
- Feature Engineering
- Model Evaluation
- Monitoring and Reliability
- Model Serving and Inference Optimization
- ML Safety and Ethical Machine Learning
- ML Service Integration and Model Deployment
- ML Infrastructure and Maintenance

You will learn

- To develop strong skills in conducting experiments, validating models, and optimizing performance through advanced feature engineering, model evaluation, and error analysis
- To design machine learning systems, from problem identification and dataset gathering to model deployment and integration into production environments.
- To monitor and maintain ML systems in production, detect and address model drift, and apply ethical considerations to create fair, unbiased, and reliable machine learning models.