Spring 2020 – CS Special Topics

1. **CS 494* – Cruz – Big Data Mining**
2. **CS 494* – Ganchinho de Pina – Principles of Concurrent Programming**
3. **CS 494* – Kanich – Secure Web Applications Development**
    - **Class will become CS 484, MOM still required for Sp'20**
4. **CS 494 * – Kash – Economics and Computation**
5. **CS 494* – Lillis – Algorithms in Practice**
    - **This class/section is for CS undergrads only**
6. **CS 494* – Polakis – Network Security**
    - **Class will become CS 469, MOM still required for Sp'20**
7. **CS 594 – DasGupta – Randomized Techniques for Designing Algorithms and Proving Lower Bounds**
8. **CS 594 – Gjomemo/Venkatesan – Program Analysis, Vulnerability Detection, and Secure Software Design**
9. **CS 594 – Sun – Algorithm Methods for Big Data**
10. **CS 594 – Wu – High-performance Data Systems**

*CS Undergraduate students must submit a modification of major (MOM) to use the class as a technical elective.

Back to List

Spring 2020 – CS Special Topics

CS 494– Big Data Mining

- Instructor: Isabel Cruz
- Meeting time: R 5-7pm
- CRN: 43287 – undergraduate
  43288 – graduate

Course Description:
This course is about *data mining* for very large amounts of data, that is, data that does not fit in main memory, called *big data*. Topics may include: similarity search, data-stream processing and algorithms, advanced technology for search engines, association rules, algorithms for clustering big recommendation systems, mining of very large graphs, dimensionality reduction, and machine learning algorithms. The class meets once per week for 3 hours.

Course Work:
There will be 3-5 homeworks/programming assignments, including Gradiance Automated Homeworks. There will be quizzes *or* two midterms.

Prerequisites: **CS 251 and STAT 381; or IE 342 or ECE 341. CS 480 is highly recommended. Graduate students need permission from the instructor.**

The course is primarily for UNDERGRADUATES, hence Graduates need authorization from me. This is because graduates have the possibility of taking part of this material in CS 5XX courses, whereas for undergrads this is their only possibility to learn this material.

Spring 2020 – CS Special Topics

CS 494– Principles of Concurrent Programming

- Instructor: Luis Ganhinho de Pina
- Meeting time: TR 2-3:15pm
- CRN: 42279 – undergraduate
  42290 – graduate

Course Objectives:
This course will be focused on the foundations and basic principles of concurrent programming: What machines need to provide to ensure that concurrent programs behave as expected. The course will start with an idealized model of computation, and move on to real models such as the standard Java memory model or C++11 memory model.

The foundational approach to concurrent programming will then be supplemented with reasoning about the performance of concurrent data-structures used in a realistic environment, and how decisions made at the architecture level impact the design of high-performant concurrent data-structures. This will involve revisiting the idealized model of computation introduced earlier, with a more pragmatic approach that fits modern (and future) architectures.

Our hope is that, at the end of the course, students will have a basic understanding of both the foundations and the practice of concurrent programming.

Class Meeting: 75 minutes meetings twice per week

Method of Instruction: Primarily lecture based

Student Deliverables: There will be 4-6 homework assignments, in which the students will implement the concepts presented in class, or a small variation of those. There will be one in-class midterm. There will be one final exam.

Prerequisites: CS361 Computer Systems is a pre-requisite for both graduate and undergraduate students.

Back to List

Spring 2020 – CS Special Topics

CS 494– Secure Web Applications Development

- Instructor: Chris Kanich
- Meeting time: TR 12:30-1:45pm
- CRN: 42280- undergraduate
  42281- graduate

Course Description:
Web applications are simultaneously one of the most widely used and widely attacked forms of deployed code. At the same time, the concepts of computer security are best taught within a relatable context so that students can immediately apply their knowledge to relevant situations. The unique challenges inherent in building secure web applications made available to billions of potential users and attackers requires understanding how to use and integrate concepts from software engineering, systems programming, and computer security. This course integrates the concepts that underlie designing, deploying, attacking, and defending web applications to provide students with a foundational understanding of how to design and deploy scalable and secure web applications.

This class will teach students the concepts and techniques that enable web applications to maintain high performance in the face of numerous users and attackers. Students will learn and be able to apply software engineering concepts to manage the complexity of client-side and server-side software. Students will learn and be able to apply computer systems concepts to manage the scalability of the web application, and provide performant service to large numbers of simultaneous users. Students will learn and be able to apply computer security concepts to designing a web application which is robust to known and unknown attacks. Students will gain familiarity and facility with modern tools which enable creating applications that apply the aforementioned design, performance, and security concepts. Students will learn and be able to apply fundamental security concepts so that they can evaluate the security of future application designs in the face of potential future attacks.

Prospective 15 week schedule:
1. Secure web server and web site deployment best practices
2. JavaScript fundamentals
g. Web browser capabilities and its API
4. Managing trust with adversarial user input
5. Software engineering for single page applications
6. Software engineering for server-side application logic
7. System design for backing stores: SQL, NoSQL, KV;
8. Advanced attacks against security
9. Web privacy
10. Usable web security

Back to List

Spring 2020 – CS Special Topics

11. System design and engineering for high demand scenarios
12. Advanced design and deployment techniques
1g. DevOps, Incident Response & Red teaming
14. Recent advancement & review

Spring 2020 – CS Special Topics

CS 494– Economics and Computation

- Instructor: Ian Kash
- Meeting time: TR 12:30-1:45pm
- CRN: 42278- undergraduate
  42289- graduate

Course Description:
In this course, students will become broadly familiarized with techniques at the interface between economics and computer science. The goal is to equip students will the tools to analyze and optimize the markets (broadly construed) they are called to play a role in designing or using over the course of their career. Upon successful completion of this course, students should be able to (a) make effective decisions in the presence of other strategic decision makers and (b) understand how their algorithmic and other design decisions affect the incentives of market participants.

Tentative Schedule:

Week	Topic
1	Game Theory I (Normal Form Games)
2	Game Theory II (Congestion Games and the Price of Anarchy)
3	Game Theory III (Extensive Form Games)
4	Peer-to-Peer and Reputation Systems
5	Mechanism Design
6	Dynamic Mechanism Design (Scheduling)
7	Auction Theory (Internet Advertising)
8	Combinatorial Auctions (Spectrum Auctions)
9	Fair Division I (Cake Cutting, Rent Division)
10	Fair Division II (Divisible and Indivisible Goods)
11	Cloud Computing
12	Matching Markets (School Choice, Kidney Exchanges)
13	Economics of Platforms (Ridesharing)
14	Information Elicitation (Scoring Rules and Prediction Markets)
15	Catchup and Additional Topics
16	Final Exam

Assignments and Grading:
There will be 4-6 homework assignments (mostly written but 1-2 programming), an in class midterm and a final exam

Prerequisites: This course will be accessible to students with a range of backgrounds in computer science. Students from other disciplines near computer science and students who come at the interface from the other side (e.g. economics or the business school) should be able to benefit from the course as well, although some topics will have a strong computational or algorithmic component. CS 251 is a minimal requirement for CS students, although additional mathematical background such as CS 401, CS 411, or completion of some of the 9 required hours of mathematics courses would be beneficial. Other students by permission of instructor.

Back to List

Spring 2020 – CS Special Topics

CS 494– Algorithms in Practice

- Instructor: John Lillis
- Meeting time: MW 3-4:15pm
- CRN: 42287

Course Description:
This course is an applied follow-up to CS401. Emphasis is on design, implementation and presentation of non-trivial algorithms. Throughout the course, there will be group based problem solving exercises with problems of varying degrees of difficulty. The course is roughly divided into three components:

1. Review of fundamental data structures and algorithmic techniques with an emphasis on dynamic programming; "review" will be done mainly through numerous case studies.
2. Exact algorithms for NP-hard problems; we study "exponential, but better than brute force" direct solutions to NP-hard problems (e.g., by dynamic programming) and also, exact solutions via reduction to Satisfiability (SAT) or Integer Linear Programming (ILP), and then leveraging off-the-shelf SAT/ILP solvers.
3. Approximate and heuristic approaches to NP-hard problems including local search techniques. Local search techniques for classic problems such as the traveling salesman problem and graph partitioning will be included.

Students should expect the following:

● programming projects: these will sometimes be team projects and require a "report" component.
● occasional written homework
● presentations: students will give presentations ("white-boarding" style) on problems drawn from group exercises and projects. These may be "team presentations" as appropriate.
● exams (tentative): expect a take-home midterm and final (these are like short-turnaround written homework assignments).

Prerequisite: CS401

Back to List

Spring 2020 – CS Special Topics

CS 494– Network Security

- Instructor: Jason Polakis
- Meeting time: TR 3:30-4:45pm
- CRN: 42271 – undergraduate
  43270 – graduate

Course Description:
This course will cover the principles and practice of network security. We will provide an overview of cryptographic foundations, followed by a discussion of network security applications. Given the ubiquitous nature of network communications in modern computing, we will cover a wide range of systems and applications, and the security threats that arise.

Topics:
- Security Principles
- Network Security Infrastructure
- Protocol Security
- Network Security Applications
- Network Defenses
- Advanced Topics

Grading:
- Assignments: 40% - Exams: 40% - Reading homework & participation: 20%

Spring 2020 – CS Special Topics

CS 594– Randomized Techniques for Designing Algorithms and Proving Lower Bounds

- Instructor: Bhaskar DasGupta
- Meeting time: MW 3-4:30pm
- CRN: 33792

Course description:
Can you check if a proof is correct by checking only a small part of the proof? Can you check if two documents are identical without checking the entire documents? The answers for many questions of these types are in the affirmative if the algorithms are allowed to toss coins.
Over the last two decades, randomization techniques are extremely powerful tools in designing algorithms as well as in showing algorithms of certain kinds cannot exist. Randomized techniques are ubiquitous in algorithmic designs and analysis and in fact, as quantum mechanics dictates, nature is inherently probabilistic. However, a systematic approach to understanding various aspects of these techniques are crucial, as opposed to just designing ad-hoc randomized techniques narrowly tailor-made for a specific application.

In this course, we will discuss various types of randomized techniques with mathematically precise analysis and show how to apply these techniques in designing algorithms for well-known CS problems. In addition, we will also discuss limits of randomization methods, namely algorithms of certain types (e.g., resource-constrained) that cannot be achieved by any randomized methods.

The required pre-requisite is CS 401 or equivalent. Prior enrollment in CS 501 is encouraged but not required. There are no required textbooks, but standard graduate-level textbooks, such as the ones listed below, may be helpful:

- M. Mitzenmacher and E. Upfal, Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis, 2nd Edition, Cambridge University Press, 2017.
- R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, New York, NY, 1995.
- N. Alon and J. H. Spencer, The probabilistic method, Wiley-Interscience, 2000.

Back to List

SPRING 2020 – CS SPECIAL TOPICS

CS 594– Program Analysis, Vulnerability Detection, and Secure Software Design

- Instructors: Rigel Gjomemo and Venkat Venkatesan
- Meeting time: T 5-7:30pm
- CRN: 33649

Course Description:
Development of secure programs is an extremely difficult task, as witnessed by countless software vulnerabilities and exploits disclosed and witnessed every year. To detect such vulnerabilities, researchers have developed several approaches, including static analysis (symbolic execution and constraint solving), fuzzing, and transparent instrumentation of program code. A vulnerability is a "weakness" in a program that may be "exploited" to subvert the intended operations of that program.

This course will examine methods for vulnerability detection in programs by introducing various topics in the broad areas of software analysis and testing, compilers, and patching. Simultaneously, more foundational topics on information flow analysis and graph analytics will be covered. The course will also touch upon principles of secure programming.

Format and Schedule:
This class will meet once a week for three hours. Students will be assigned required readings about relevant literature for each class and the instructors will summarize the essence of the readings with a lecture. This will be followed by a critical discussion of the specific topic of discussion. Participants will be assigned tools created by both academia and industry to analyze and test programs to find vulnerabilities. They will work in teams in class projects related to the class topics.

Topics:
Overview of program representations
Overview of software analysis methods for vulnerability detection
Static analysis techniques
Dynamic analysis techniques
Overview of approaches and tools for software analysis
Automatic exploit generation
Human collaboration with automatic tools
Principles of secure software design
Class project discussion.

Evaluation:
There will be regularly assigned readings for each class. A critical written review of the papers (in a suggested format) will be assigned homework at the beginning of each class. (Reviews are due on Sundays before the presentation date). This will constitute about 30% of the grade. The remaining 70% of the grade will come from an evaluation of the student's class project. There will be no exams.

Back to List

Spring 2020 – CS Special Topics

Prerequisites:
Some background in computer security is required. Preparation that is equivalent to CS487 (or) CS587 (or) permission of the instructor.

Back to List

Spring 2020 – CS Special Topics

CS 594– Algorithm Methods for Big Data

- Instructor: Xiaorui Sun
- Meeting time: TR 11a-12:15pm
- CRN: 42282

Course Overview:
Recently there has been a lot of glorious hullabaloo about Big Data and how it is going to revolutionize the way of computation. As part of the general excitement, it has become clear that for truly massive datasets and digital objects of various sorts, even algorithms which run in linear time (linear in the size of the relevant dataset or object) may be much too slow. Traditional models and paradigms of sequential computation has become obsolete. Different computation scenarios ask for different trade-offs between solution quality, space usage, running time, and communication. These paradigms are formalized as sublinear, streaming, parallel, and other distributed algorithmic models of computation. New algorithm techniques within these models have been developed, and have played an important role in the modern data analysis. This course aims at timely dissemination of foundational algorithmic developments for big data analysis and exposing students to cutting edge research in this area. The course will involve deep theoretical analysis with the goal of developing practical algorithms for a variety of applications.

Prerequisites:
You should be familiar with the following topics:
1. Important: Big-O notation and basic analysis of algorithms. CS 401: Computer Algorithms I or equivalent will be assumed.
2. Important: You should be comfortable with proofs, basic discrete math, and probability. A solid undergraduate-level mathematics background is good preparation. The course CS 151 is a good sources. If you have questions about your mathematical readiness you should contact the instructor before enrolling.
3. Less important: Prior graduate level coursework in theoretical computer science will be helpful, but the course should be accessible to mathematically strong undergraduates as well. It may be helpful to have taken CS 501 (computer algorithms II), but this is not a requirement.
4. Less important: We will freely use standard tail bounds for sums of independent random variables (i.e. Chernoff bounds and Hoeffding bounds) so it may be helpful to come into the course familiar with these basic tools. Here is a quick summary of these results that has been helpfully prepared.
The course will not include any programming.

Method of instruction:
This course will be based on the following main components:
1. During the first half of the course, the instructor will present the computational models and algorithmic techniques for big data analysis.
2. During the second half of the course, the students will read and present research papers that are related to the course.

3. The students will do a project on a topic related to the course material. The students may do a solo project or work with one partner. A list of topics for the survey papers will be provided. Projects can be either implementation based or theoretical depending on interest.

Student Deliverables:
Students are expected to come to lecture and are encouraged to participate. Each student will prepare scribe notes for one or two lectures during the semester. All the lecture notes will be made available on the class web page. Each student will present at least one research paper to the class. Each student will complete a research project on a topic related to the course material, and give a brief presentation in class on his or her final project.

Back to List

Spring 2020 – CS Special Topics

CS 594– High-performance Data Systems

- Instructor: Xingbo Wu
- Meeting time: MW 4:30-5:45pm
- CRN: 34724

Course Description:
This course focuses on the design and implementation of low-level storage, indexing, and caching in high-performance data systems such as Key-value stores and NoSQL databases. Today's many-core CPUs, large memory/SSD, and fast network have made it possible to quickly store and process large volumes of data. However, established systems are often bottlenecked by software inefficiencies such as serialization caused by locking and excessive rewrites caused by sorting. We will discuss new mechanisms that can offer better performance and efficiency. Students will get familiar with advanced data structures, system programming interfaces, and design principles behind representative systems.

Condensed syllabus:

This course will cover the following topics:

1. Index structures: (hash tables and ordered index), concurrency control (locks, lock-free/lock-less, RCU), and mechanisms for persistency/consistency (logging & journaling, COW).
2. Hardware/OS interfaces: Network/storage I/O (polling/multiplexing, kernel-bypassing, DirectIO/DAX, multi-level caching), and in-kernel/in-hardware processing (in-kernel KV cache).
3. Distributed data system. consistent hashing, transaction processing, load balancing, and fault-tolerance (replication, erasure coding).

The instructor will give regular lectures (~10 weeks), followed by student paper presentations and discussions (~5 weeks). Students will be writing two paper reviews and complete two programming projects.

Grading:
10%: class participation
20%: paper reviewing
20%: paper presentation
50%: two programming projects